FORMATION À JQUERY

Thomas Morin

décembre 2012

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, Autocompleter, 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 ) ;
        }); // ferme la fonction anonyme et la méthode
})(jQuery);
```

JSON : La notation d’objet en javaScript

Une écriture synthétique

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

Le même chemin qu'en CSS

Syntaxe :

```
var $capture = $("sélecteur")
```

Élément, identifiant et classe de la CSS2

```
var $h1 = $("h1") ;
var $top = $("#top") ;
```

Attributs de la CSS3

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

Opérateurs

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

Pseudo-éléments

```
var $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") ;
```

Format Json

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

Information sur le style

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

TRAVAUX PRATIQUES 1

Attributs avec la méthode .attr()

Paires de clés et de valeurs

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

Format Json

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

Obtenir les attributs d'un nœud

```
var href = $lien.attr("href ") ;
```

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") ;
```

Information sur la classe

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

Manipuler les classes ou injecter du style ?

Contenus texte et html

Insérer du contenu texte

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

Insérer du contenu html

```
$a.html("<strong> Revenir au sommaire </strong>")
```

Récupérer les contenus texte et html

```
html = $a.html() ;
    // Retourne "<strong> Revenir au sommaire </strong>"
texte = $a.text() ;
    // Retourne "Revenir au sommaire"
```

TRAVAUX PRATIQUES 2

Plusieurs nœuds peuvent composer l'objet

```
<ul>
        <li>…</li>
        <li>…</li>
</ul>
<ul>
        <li>…</li>
</ul>
```

Sélection des items

```
var $li = $("li") ;
alert($li.length) ;  // Propriété length : nombre d'item
alert($li.text()) ;   // Fusionne les contenus texte
alert($li.html()) ;  // Contenu html du premier item
```

Parcourir les nœuds 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
```

Parcourir les nœuds *autours* d'un objet jQuery

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

Des méthodes pour parcourir

Syntaxe

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

Descendre

find(), children()

Remonter

parent(), closest()

En latéral

next(), prev(), siblings()

Manipuler

filter(), not(), add()

Des méthode pour modifier

Syntaxe

$parent.append($nouvelEnfant) ;

Descendre

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

Remonter

wrap(), unWrap(), wrapInner()

En latéral

after(), before(), insertAfter(), 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
$tBody.children("tr")
        // Tri selon l'identifiant
       .sort(function(ligneA,ligneB) { // Fonction de tri :
            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 "click"

```
$("a").click(
    // Fonction anonyme en paramètre :
    function() {
            // Nœud courant : celui qu'on a cliqué
            var $noeudClique = $(this) ;
            // Instruction
            var title = $noeudClique.attr("title");
            alert(title) ;
    } // ferme la fonction anonyme
) // ferme la le méthode click
```

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

Exemple avec la position du curseur

```
// Quand on déplace la souris
$body.mousemove(function(event) {
        // Event est l’évènement passé en parametre
        var curseurX = event.pageX ;
        var curseurY = event.pageY ;
        // Affiche les coordonnées de la souris
        $p.html(curseurX + ", " + curseurY ) ;
})
```

Utile également pour la méthode .hover()

Valeur de retour

L’action est annulée lorsque la fonction renvoie « faux »

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

Utile également pour la méthode .click()

Déclencher l’évènement

Sans fonction passée en paramètre, les méthodes déclenchent l’événement automatiquement :

```
// Soumet un formulaire :
$("form").submit() ;

// Clique sur le premier bouton trouvé
$("button").eq(0).click() ;
```

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 :

```
$animated.stop(
    clearQueue, // Passé à true, vide la pile d'exécution
    goToEnd // Passé à true, force l'animation à se terminer
)
```

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

Échange 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: "auteur1" },
        // 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 :
        "authentification.asp",
        // Paramètres :
        {login: "mon-login", password: "mon-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...
        }
);
```

Paramètres de la fonction de retour

data (1er paramètre) :

Contenu du fichier chargé sous forme de chaîne de caractère

error (2nd paramètre) :

Indique la réussite du chargement

Exemple

```
$("#success").load("/not-here.php",function(data, status) {
    // Si le chargement échoue status vaut "error"
    if (status == "error") alert("Échec !") ;
    // Data est une chaîne de caractère
    else alert(data)  ;
 });
```

TRAVAUX PRATIQUES 6

Des extensions de l'objet jQuery ($)

Fonctions et propriétés de jQuery

```
$.mesProprietes = {texte:"a",nombre:22}
$.maFonction = function(parametres) {
    // ...Instructions...
}
// Exemples : $.fx.off, $.trim()
```

Méthodes des objets jQuery

```
$.fn.maMethode = function (parametres) {
    // ...Instructions...
})
// Exemple : $.fn.prepend()
```

Grouper les extensions avec $.extend()

Fonctions et propriétés de jQuery :

```
$.extend({
        maFonction :  function(parametres) {...},
        mesPropriétés :  {...}
});
// ⇔ $.maFonction = function(parametres) {…}
```

Méthodes des objets jQuery :

```
$.fn.extend({
        maMethode1 : function(parametres) {...},
        maMethode2 : function(parametres) {...}
});
// ⇔ $.fn.maMethode1 = function(parametres) {…}
```

Construire des méthodes chainées

Déclaration de la fonction

```
$.fn.maMethode = function {
    // mot réservé « this », c'est un objet jQuery !
    var $lesNoeuds = this ;
    // Instructions, par exemple :
    alert ($lesNoeuds.length) ;
    // La méthode retourne l'objet auquel elle s'applique
    return $lesNoeuds ; // ⇔ return this ;
}
```

Utilisation chaînée

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

Parcourir les nœuds de l'objet jQuery

Méthode .each() pour parcourir un à un les nœuds auxquels la méthode s'applique :

```
$.fn.maMethode = function {
    var $lesNoeuds = this ;
    // boucle .each() sur les nœuds
     $lesNoeuds.each(function(i){
            var $noeudCourant = $(this) ;
            //...instructions...
    })
    return $lesNoeuds ;
}
```

Configuration avec $.extend(conf,options)

Les paramètres passés à la fonction écrasent les paramètres par défaut

```
$.fn.maMethode = function(options) {
    // Configuration dans un objet jSon
    var confDefaut = {
        vitesse : 2000 ,
        callback : function(a) { ... }
    }
    conf = $.extend(confDefaut ,options) ;
    // ... Instructions en utilisant conf.vitesse, conf.callback
    alert(conf.vitesse) ;
}
```

Canevas « classique » des méthodes

```
$.fn.maMethode = function(options) {
    // Groupe de nœuds auxquels la méthode s'applique
    var $this = $(this) ;
    // Configuration par défaut :
    var confDefaut = {…} ;
    // Les paramètres écrasent la configuration :
    var conf = $.extend(confDefaut,options) ;
    // Instructions
    $this.each(function(){ .. }) ;
    // Retourne le groupe de nœuds
    Return $this ;
}
```

Canevas « compact » des méthodes

```
$.fn.maMethode = function(options) {
    // Configuration dans un objet jSon écrit à la volée :
    var conf = $.extend({
            vitesse : 2000 ,
            callback : function(a) { ... }
    },options) ;
    // Les instructions et le retour sont chaînés :
    return this.each(function(){
            var noeudCourant = $(this) ;
            // ... Instructions
    })
}
```

Concevoir sa fonction comme un plugin

Une fonction principale :

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

Des paramètres par défaut :

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

Des fonctions utilisée 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)

jCaroussel :

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