Université Montpellier-II
IUT de Béziers
Département SRC

UNIVERSITÉ MONTPELLIER 2
SCIENCES ET TECHNIQUES

Programmation Web avancée : jQuery et Ajax

Responsable : Chouki TIBERMACINE
Bureau : Direction des études
Tél. : 04 67 11 18 21
Mél. : Chouki.Tibermacine@iutbeziers.fr
Web : http://www.lirmm.fr/~tibermacin/ens/pwa/

2011-2012

Plan du module

- Cours 1 : Manipulation de documents HTML avec jQuery
- Cours 2 : Gestion des événements et animations avec jQuery
- Cours 3 : Fonctions avancées de jQuery
- Cours 4 : Style architectural d'applications Web Ajax

Cours 1

Manipulation de documents HTML avec jQuery

Plan du cours

- Introduction à la librairie JavaScript jQuery
- Utilisation d'accesseurs aux éléments HTML et CSS
- Manipulation de la structure de documents HTML

Plan du cours

- Introduction à la librairie JavaScript jQuery
- Utilisation d'accesseurs aux éléments HTML et CSS
- Manipulation de la structure de documents HTML

Défauts du langage JavaScript utilisé seul

- Écriture de code trop verbeuse pour des traitements récurrents
- Manque de langage de requête déclaratif (simplifier le code)
- Exemple : récupérer tous les divs dans un document

```
document.getElementsByTagName("div")
```

- Son équivalent en jQuery :

```
$("div")
```

- Et si on veut filtrer cet ensemble pour récupérer les éléments qui sont d'une certaine classe CSS ?
  - En JavaScript : définir une boucle, puis tester la valeur de l'attribut class, ...
  - En jQuery : `$("div.maClasse")`

C'est quoi jQuery ?

- Une **librairie JavaScript** gratuite (pas un nouveau langage) qui fonctionne de façon **unifiée sur tous les navigateurs**

- Un ensemble d'objets et de fonctions permettant :
  - D'accéder, modifier ou supprimer des éléments HTML et leurs propriétés CSS en utilisant des expressions agrégées
  - Gérer les évènements de façon indépendante des navigateurs
  - Produire des animations de façon très simple
  - Naviguer et interroger (d'où le nom jQuery) un document HTML de façon déclarative

- Équivalent de xPath et xQuery dans les technologies XML (mais implémenté au dessus d'un langage de script)

Comment utiliser jQuery ?

- Déclarer dans un document HTML dans une balise <script> le chemin où se trouve le fichier (.js) contenant la librairie
  Même technique pour intégrer n'importe quelle librairie JavaScript (même si vous définissez votre propre librairie)

- Deux solutions sont envisageables : (chacune ses avantages)
  - Télécharger la librairie sur votre disque
  - La référencer en utilisant une URL :
    http://code.jquery.com/jquery-xxx.js (où xxx est la version)

- Exemple : la librairie se trouve dans le répertoire « js »

```
<script type="text/javascript"
        src="js/jQuery.js">
</script>
```

La fonction jQuery() ou $()

- La fonction principale et le point de départ de votre code jQuery
- Quatre façons différentes d'utiliser cette fonction :
  - Lui passer en argument une chaîne contenant un sélecteur CSS
    - Elle retourne l'ensemble des éléments qui correspondent à ce sélecteur
    - Ceci sera détaillé dans un prochain cours
  - Lui passer en argument un élément HTML, un document ou l'objet Window
    - La fonction va envelopper cet objet et le transformer en objet jQuery (qui est retourné) sur lequel vous pouvez invoquer les fonctions jQuery (au lieu d'utiliser les fonctions DOM)

Façons d'utiliser la fonction jQuery() ou $() -suite-

- ...
  - Lui passer en argument une chaîne contenant une balise HTML
    - Elle retourne un objet jQuery enveloppant le nouvel élément HTML créé (qui n'est pas encore inséré dans le document !!!)
    - Exemple :

```
var img = $("<img/>",{ src:url, alt:desc });
```

  - Lui passer en argument une fonction :
    - Cette fonction sera exécutée lorsque le chargement du document est fini (le DOM est prêt)
    - C'est la façon la plus utilisée pour écrire du code jQuery
    - Exemple : -c'est ce qu'on va faire en TP-

```
jQuery(document).ready(function() {
   // Placer tout votre code ici
});
```

Requêtes jQuery

- On peut passer en argument de la fonction jQuery() une chaîne représentant un sélecteur CSS

- Exemples de sélecteurs :

```
div          // tous les éléments <div> du document
#prenom      // l'élément qui a un id="prenom"
.contenu     // tous les éléments avec class="contenu"
```

- jQuery supporte des sélecteurs plus sophistiqués (détaillés dans un prochain cours)

Résultats de requêtes jQuery

- Le résultat retourné par la fonction jQuery(...) ou $(...)
  est un objet jQuery

- Cet objet ressemble à un tableau : il a une propriété length
  et les éléments sélectionnés peuvent être accédés par un indice
  Vous pouvez cependant invoquer : size() et get(<indice>)

- Exemple :

```
$("body").length //retourne 1-un seul body par document
$("body")[0]     //équivalent à : document.body
```

Parcours des résultats de requêtes jQuery

- Utiliser la fonction each() sur l'objet jQuery retourné
- Cela permet d'écrire du code plus abrégé : mieux qu'une boucle pour parcourir tous les éléments sélectionnés
- Il faut passer à each() une fonction (callback) qui va être invoquée pour chaque élément dans l'objet qui contient les résultat
- Le mot clé this référence l'élément en question
- La fonction (callback) peut déclarer comme premier argument l'indice de l'élément et comme 2ème argument l'élément (this)

Parcours des résultats de requêtes jQuery - suite -

- Cette fonction retourne l'objet jQuery résultat pour permettre le chaînage d'invocations de fonctions
- Pour simuler un break, faites un : return false;
- Exemple :

```
$("div").each(function() {// Pour tous les <div>
 if($(this).is(":hidden"))
  return; // Ignorer les div cachés (ex : display=none)
 $(this).onclick = function() {...};
});
```

Plan du cours

- Introduction à la librairie JavaScript jQuery
- Utilisation d'accesseurs aux éléments HTML et CSS
- Manipulation de la structure de documents HTML

Généralités sur les accesseurs dans jQuery

- Il n'y a pas une paire de setter et getter, mais plutôt une seule fonction : getter si invoquée sans arguments et setter sinon.
- Lorsqu'un setter est invoqué sur un objet jQuery contenant plusieurs éléments, ce setter changera les valeurs de tous les éléments
- Un getter s'applique sur le premier élément seulement (chaîner avec map() pour récupérer les valeurs de tous les éléments)
- Un getter ne retourne pas l'objet jQuery sur lequel il s'applique. Il est donc utilisé à la fin d'une chaîne d'invocations de fonctions

Généralités sur les accesseurs dans jQuery - suite -

- Un setter reçoit parfois comme argument un objet. Dans ce cas, cet objet précise des propriétés (nom de la propriété et sa valeur) qui sont utilisées par le setter pour mettre à jour l'élément ou les éléments sélectionnés

- Un setter reçoit souvent comme argument une fonction comme valeur. Dans ce cas, cette fonction est invoquée pour calculer la valeur utilisée par le setter

Accesseurs d'attributs HTML

- La fonction attr() est l'accesseur (getter/setter) pour les attributs HTML
- La fonction removeAttr() permet de supprimer un attribut d'un ou de plusieurs éléments HTML sélectionnés

- Exemples :

```
// récupérer l'attribut action de <form> (getter)
$("form").attr("action");
// Fixer la valeur de l'attribut src d'un élément
// ayant comme id icon (setter)
$("#icon").attr("src","icon.gif");
// Fixer les valeurs de 4 attributs à la fois
$("#banner").attr({src:"banner.gif",alt:"Publicite",
                   width:720,height:64});
```

Exemples d'accesseurs aux attributs HTML - suite -

```
// Faire en sorte que tous les liens se chargent
// dans une nouvelle fenêtre
$("a").attr("target","_blank");
// Calculer la valeur de l'attribut target :
// Charger les pages locales dans la même fenêtre
// et les autres pages dans une nouvelle fenêtre
$("a").attr("target",function(){
  if(this.host==location.host) return "_self"
  else return "_blank"
});
// On peut passer en paramètre des fonctions ainsi :
$("a").attr({target:function(){...}});
// Faire en sorte que tous les liens se chargent
// dans la même fenêtre
$("a").removeAttr("target");
```

Cas particuliers de l'utilisation de la fonction attr()

- Si la fonction attr() est invoquée avec comme premier argument "css", "html", "text", ... , d'autres fonctions sont invoquées de façon implicite

- Exemple :

```
attr("css",{backgroundColor:"gray"})
->
css({backgroundColor:"gray"})
```

- Ces fonctions seront abordées dans les prochains transparents

Accesseurs d'attributs CSS

- La fonction css() ressemble beaucoup à la fonction attr(), mais elle fonctionne exclusivement avec les attributs CSS
- Quand elle joue le rôle de setter, cette fonction ajoute un style CSS à l'attribut « *style* » de l'élément HTML sélectionné
- La valeur retournée par cette fonction (cas de getter) peut provenir du document HTML ou d'une feuille de style associée
- Il est impossible d'accéder à des styles CSS composés (font ou margin, par exemple), mais à des styles individuels (font-weight, margin-top, par exemple)

Valeurs d'attributs CSS manipulées par css()

- La fonction css() peut prendre en argument par exemple "background-color" ou "backgroundColor"
- Dans le cas de getter, cette fonction retourne les valeurs numériques sous forme de chaînes (les unités incluses)
- Dans le cas de setter, cette fonction convertit les valeurs numériques en chaînes et ajoute, la cas échéant, les unités (*px* pour pixels, par exemple)

Exemples d'accesseurs d'attributs CSS

```
// Récupérer le font-weight du premier <h1>
$("h1").css("font-weight");
// Une autre façon de faire la même chose
$("h1").css("fontWeight");
// ERREUR : style composé
$("h1").css("font");
// Fixer le style sur tous les <h1>
$("h1").css("font-variant","smallcaps");
// Ceci marche pour un style composé (SETTER)
$("div.note").css("border","solid black 2px");
// Fixer les valeurs de plusieurs attributs CSS
$("h1").css({backgroundColor:"black",textColor:"white",
             fontVariant:"small-caps",
             padding:"10px 2px 4px 20px",
             border:"dotted black 4px"});
```

Accesseurs de classes CSS

- L'accès à la liste des classes CSS associées à un élément HTML peut se faire grâce à la propriété *class*
- La fonction addClass() permet d'ajouter une classe CSS à un élément HTML sélectionné
- La fonction removeClass() permet de supprimer une classe
- La fonction toggleClass() permet d'ajouter une classe quand celle-ci n'est pas définie dans l'élément HTML sélectionné et la supprime sinon
- La fonction hasClass() permet de tester la présence d'une classe

Exemples d'accesseurs de classes CSS

```
// Ajouter une classe CSS à tous les éléments <h1>
$("h1").addClass("brillant");
// Ajouter deux classes aux éléments <p> se trouvant
// après les éléments <h1>
$("h1+p").addClass("brillant premierpara");
// Ajouter une classe différente (calculée)
// à des éléments HTML différents
$("section").addClass(function(n) {
  return "section" + n;
});
```

Exemples d'accesseurs de classes CSS - suite -

```
// Supprimer une classe de tous les <p>
$("p").removeClass("brillant");
// Plusieurs classes peuvent être supprimées
$("p").removeClass("brillant premierpara");
// Supprimer toutes les classes des <div>
$("div").removeClass();
// Ajouter ou supprimer une classe en fonction
// de sa présence
$("tr:odd").toggleClass("ligneimpair");
// Est-ce qu'il y a un <p> qui a une certaine classe ?
$("p").hasClass("premierpara")
```

Accesseurs de valeurs dans des formulaires HTML

- La fonction val() est utilisée pour accéder en lecture et en écriture aux valeurs des éléments dans des formulaires HTML
- Elle permet entre autres d'accéder à l'état de sélection des cases à cocher, des boutons radio ou des éléments <select>

Exemples d'accesseurs de valeurs dans des formulaires

```
// Obtenir la valeur du champs ayant l'id prenom
$("#prenom").val()
// Obtenir une seule valeur d'un <select>
$("#dept").val()
// Obtenir un tableau de valeurs d'un <select multiple>
$("select#sports").val()
// Obtenir la valeur d'un bouton radio sélectionné
$("input:radio[name=civilite]:checked").val()
// Préciser la valeur d'un champs de texte
$("#email").val("Mot de passe invalide");
// Remettre les valeurs par défaut dans tous les champs
$("input:text").val(function() {
  return this.defaultValue;
});
```

Accesseurs de contenus d'éléments HTML

- Les fonctions text() et html() permettent d'accéder en lecture et en écriture au contenu texte ou HTML d'un élément sélectionné
- La fonction text(), invoquée, sans arguments retourne le contenu texte de tous les nœuds descendants des éléments sélectionnés
- La fonction html(), invoquée sans arguments, retourne le contenu HTML du premier élément uniquement
- Si ces fonctions sont invoquées avec une chaine en argument, elle remplace le contenu d'un élément par cette chaine
- Il est possible d'appeler ces fonctions en leur passant en argument une fonction

Accesseurs de contenus d'éléments HTML

```
// Obtenir le titre du document
var t = $("head title").text();

// Obtenir l'HTML du premier élément <h1>
var hdr = $("h1").html();

// Donner à chaque entête un numéro de section
$("h1").text(function(n,current) {
  return "N°" + (n+1) + current;
});
```

Accesseurs de positions d'éléments HTML

- La fonction offset() permet d'obtenir la position d'un élément HTML dans un document
- Elle retourne un objet ayant les propriétés *left* et *top*, qui stockent les coordonnées x et y de l'élément HTML
- Un objet ayant de telles propriétés peut être passé en paramètre (dans ce cas offset() joue le rôle de setter)
- Exemple :

```
var elt = $("#monElem");
var pos = elt.offset(); // récupérer la position
pos.top += 100;
elt.offset(pos); // Fixer une nouvelle position
```

Accesseurs de positions et dimensions d'éléments HTML

- La fonction position() retourne la position d'un élément HTML par rapport à son parent (getter seulement)
- La fonction offsetParent() retourne le parent d'un élément
- Les fonctions suivantes retournent les dimensions d'un élément : width(), height(), innerWidth(), innerHeight(), outerWidth() et outerHeight()
- Elles prennent en considération les padding, border et margin de l'élément
- Les fonctions width() et height() sont des setters aussi
- Les valeurs passées peuvent être des nombres (px) ou des chaines

Accesseurs de défilement d'éléments HTML

- Les fonctions scrollTop() et scrollLeft() retournent les positions de la barre de défilement
- Elles peuvent jouer le rôle de setters aussi
- Exemple : défiler la fenêtre de n pages

```
function page(n) {
  var w = $(window); // envelopper l'objet window
  var hauteurPage = w.height();
  var posActuelle = w.scrollTop();
  w.scrollTop(posActuelle + (n*hauteurPage));
}
```

Plan du cours

- Introduction à la librairie JavaScript jQuery
- Utilisation d'accesseurs aux éléments HTML et CSS
- Manipulation de la structure de documents HTML

Insérer et remplacer des éléments

- Il existe plusieurs fonctions jQuery qui permettent d'insérer du contenu à l'intérieur, avant, après ou à la place d'un élément ou de plusieurs éléments HTML
- Ce contenu peut être du texte, de l'HTML, des objets jQuery, des éléments ou des nœuds de type texte
- Ces fonctions retournent l'objet jQuery sur lequel elles ont été invoquées
- Elles peuvent recevoir en paramètre une fonction qui calcule le contenu à insérer ou qui va remplacer du contenu

Exemples d'insertion et remplacement d'éléments

```
// Ajouter du contenu à la fin d'un élément #log
$("#log").append("<br/>"+message);
// Ajouter du contenu au début de chaque <h1>
$("h1").prepend("N°");
// Insérer une ligne horiz. avant et après chaque <h1>
$("h1").before("<hr/>");
$("h1").after("<hr/>");
// Remplacer <hr/> par <br/>
$("hr").replaceWith("<br/>");
// Remplacer <h2> par <h1> en préservant le contenu
$("h2").each(function() {
  var h2 = $(this);
  h2.replaceWith("<h1>"+h2.html()+"</h1>");
});
```

Insérer et remplacer des éléments : fonctions alternatives

Opération effectuée	$(cible) .fonction(contenu)	$(contenu) .fonction(cible)
- Insérer du contenu à la fin de la cible	append()	appendTo()
- Insérer sur contenu au début de la cible	prepend()	prependTo()
- Insérer du contenu après la cible	after()	insertAfter()
- Insérer du contenu avant la cible	before()	insertBefore()
- Remplacer la cible par le contenu	replaceWith()	replaceAll()

Mêmes exemples avec les nouvelles fonctions

```
// Ajouter du contenu à la fin d'un élément #log
$("<br/>"+message).appendTo("#log");

// Ajouter du contenu au début de chaque <h1>
$(document.createTextNode("N°")).prependTo("h1");

// Insérer une ligne horiz. avant et après chaque <h1>
$("<hr/>").insertBefore("h1");
$("<hr/>").insertAfter("h1");

// Remplacer <hr/> par <br/>
$("<br/>").replaceAll("hr");
```

Copier des éléments

- Si vous insérez des éléments qui se trouvent déjà dans le document HTML, ceux-ci sont alors déplacés, mais pas copiés
- La fonction clone() permet de copier un ou plusieurs éléments sélectionnés, ainsi que leurs descendants (dans un objet jQuery, retourné par cette fonction)
- Ensuite, il faut utiliser les fonctions précédentes pour insérer ce qui a été copié dans le document
- Il faut noter ici, que clone() ne copie pas normalement les gestionnaires d'événements (placer *true* en paramètre si copie intégrale souhaitée)

Exemple de copie d'éléments

```
// Insérer un <div> avec un entête
$(document.body).append("<div id='listeLiens'>
<h1>Liste de liens</h1></div>");

// Copier tous les liens du document
// et les insérer à la fin du div précédent
$("a").clone().appendTo("#listeLiens")

// Insérer des retours à la ligne après chaque lien
$("#listeLiens > a").after("<br/>");
```

Supprimer des éléments

- La fonction empty() permet de supprimer tous les nœuds fils des éléments sélectionnés (sans altérer les éléments eux-mêmes)
- La fonction remove() permet de supprimer l'élément et tout son contenu du document
- Si cette fonction est invoquée avec un paramètre. Celui-ci est considéré comme sélecteur. Seuls les éléments qui correspondent au sélecteur sont supprimés

Supprimer des éléments - suite -

- La fonction filter() permet de supprimer des éléments de l'ensemble des éléments sélectionnées, sans les supprimer du document

- La fonction detach() fonctionne comme remove(), mais elle ne supprime pas les gestionnaires d'événement

Références bibliographiques

- Bear Bibeault et Yehuda Katz. « **jQuery in Action** ». Second Edition. Manning Publications Co. Juin 2010.
- David Flanagan. « ***jQuery – Pocket Reference*** ». First Edition. Éditions O'Reilly. Décembre 2010.
- Anthony T. Holdener III. « **Ajax: The Definitive Guide – Interactive Applications for the Web** ». Éditions O'Reilly. Janvier 2008.
- Documentation en ligne (très pratique) : **http://api.jquery.com/**

Questions