DUT MMI – IUT de Marne-la-Vallée
19/02/2019
M2202 – Algorithmique et programmation Javascript

Cours 3
La bibliothèque jQuery

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

iut Marne-la-Vallée

Philippe Gambette

Plan

Sources

- Cours de Jean-Loup Guillaume
http://jlguillaume.free.fr/www/documents/teaching/ntw1213/LI385_C5_Jquery.pdf

- Cours de programmation web avancée de Thierry Hamon
https://perso.limsi.fr/hamon/PWA-20122013/Cours/JQuery.pdf

- *jQuery : écrivez moins pour faire plus !*, de tit_toinou
http://openclassrooms.com/courses/jquery-ecrivez-moins-pour-faire-plus

- *jQuery, Le guide complet*, de Guillaume Allain et Timothy Stubbs

- *Javascript & Ajax pour les nuls*, d'Andy Harris

- Documentation de jQuery : http://api.jquery.com/

Introduction

jQuery = bibliothèque Javascript
pour simplifier le développement de sites web interactifs

Présenté pour la première fois en janvier 2006 par son créateur John Resig.
Première version stable en août 2006.
Téléchargeable sur http://jquery.com/download/

Javascript et jQuery en pratique :

- code exécuté par le navigateur (« côté client »)
- débuggage possible avec F12 ou clic droit « examiner/inspecter l'élément »
- messages d'avertissement et d'erreur visible dans la console du navigateur
- pour écrire dans la console en Javascript : `console.log("blabla")`
- pour afficher dans le navigateur une fenêtre bloquante avec un message en Javascript : `alert("blabla")`

Avantages

- Puissante
- Légère (94 Ko), avec utilisation possible du fichier sur les serveurs de Google : https://developers.google.com/speed/libraries/devguide#jquery
- Multiplateforme : évite les problèmes de compatibilité entre navigateurs
- Gratuite et open source
- Facilite la sélection d'éléments d'une page web
- Facilite l'AJAX
- S'écrit en dehors du code HTML

Principe de jQuery

1. Sélectionner une partie du document.

2. Agir dessus

Objet jQuery = ensemble de noeuds du DOM (Document Object Model)

→ ensemble de balises du document

→ les objets jQuery se créent avec la fonction `jQuery()` abrégée en `$()` :

- prend en entrée une chaîne de caractères contenant un «sélecteur»
- renvoie en sortie un objet jQuery

Exemple :

1. `$("div")` renvoie un objet contenant tous les "`div`" du document.

2. `$("div").hide()` cache tous les "`div`" du document.

En pratique

- Insérer le lien vers la bibliothèque jQuery pour la charger

```
<SCRIPT TYPE="text/javascript"
SRC="https://ajax.googleapis.com/ajax/libs/jquery/2
.1.3/jquery.min.js"></SCRIPT>
```

- Attendre le chargement de la page :

```
<SCRIPT TYPE="text/javascript">
$(document).ready(function(){
   // vos instructions Javascript/jQuery ici
})
</SCRIPT>
```

- **`jQuery()`** s'abrège en **`$()`**

→ attention aux conflits avec d'autres librairies :
http://learn.jquery.com/using-jquery-core/avoid-conflicts-other-libraries/

- **`$(document).ready(`**`function(){…}`**`)`** s'abrège en **`$(`**`function(){…}`**`)`**

Sélection d'éléments

Possibilité de sélectionner :

- par type de bloc
- par identifiant
- par classe
- par nom d'attribut
- en combinant les critères
- en faisant référence aux positions relatives dans le DOM
- en ne récupérant qu'un seul élément parmi les objets sélectionnés
- en filtrant parmi les objets sélectionnés

Similaire à sélection CSS !

Point Javascript :

Utilisation d'un **préfixe "js-"** pour distinguer les classes et identifiants utiles pour **jQuery/Javascript** de celles et ceux utiles pour la mise en forme en **CSS**.

Voir page https://css-tricks.com/stop-using-css-selectors-non-css/ et les commentaires en particulier.

Sélection d'éléments par type de bloc, identifiant, classe

Pour renvoyer toutes les balises : `$("*")`

Pour renvoyer tous les blocs `<div>` de la page : `$("div")`

`$("div").length` *donne le nombre de div dans la page.*

```
// pour sélectionner <span id="js-test">JL</span> :
$("#js-test")

// pour sélectionner <ul class="js-test">JL</ul> :
$(".js-test")

// pour sélectionner <div toto="titi">JL</div> :
$("[toto='titi']")
```

Point Javascript :

En Javascript, ce qui est écrit après `//` (sur la même ligne) est ignoré par le navigateur : on dit que c'est un **commentaire**. Pour faire un commentaire sur plusieurs lignes : `/* plusieurs lignes... */`

En Javascript, on écrit les blocs de texte **entre des guillemets doubles** ou bien **entre des guillemets simples**. On les appelle des **chaînes de caractères**.

Exemples :
```
alert("Aujourd'hui il fait beau") ;
alert('La page dit "bonjour"');
```

Sélection d’éléments par combinaison de critères

```
// tous les blocs div de classe js-main
$("div.js-main")

// tous les tableaux d'identifiant js-data
$("table#js-data")

// objets d'id "js-content" ou de classe "js-menu"
// attention à la position des guillemets
$("#js-content, .js-menu")
```

Sélection d'éléments filtrée

```
// Recherche de p contenant des objets avec classe
// js-header pour rendre visibles ces objets
$("p").find(".js-header").show();

// similaire à $(selecteur, contexte)
$(".js-header", $("p")).show();
```

Point Javascript :

En Javascript, les **fonctions** (comme `hide` ou `show`) permettent de **réaliser des actions**. Quand on souhaite **utiliser une fonction**, on écrit :

- ce sur quoi on veut l'appliquer (par exemple un sélecteur jQuery)
- suivi d'un point
- suivi du nom de la fonction
- suivi de parenthèses.

Entre les parenthèses, on écrit parfois des paramètres, séparés par des virgules (voir exemples diapo 37 pour la fonction `show`).

Cette fonction peut renvoyer une valeur. Dans ce cas on peut la **stocker** dans une **variable**. Par exemple, après `var num = parseInt("3")`, la variable `num` contient la valeur entière `3`.

Sélection d'éléments filtrée par numéro d'élément

```
// récupération de tous les éléments
// + extraction du troisième

$("div").get(2)

$("div")[2] // équivalent

// récupération d'un seul élément (le troisième)

$("div").eq(2)

// en partant de la fin

$("div").eq(-2)
```

Sélection d'éléments fondée sur la structure du DOM

Possibilité d'atteindre :

- les enfants (>) ;
- tous les descendants (espace) ;
- le (+) ou les (~) frères suivants.

```
<ul>
<li>item 1</li>
<li>item 2</li>
<li class="trois">item 3
<ol><li>3.1</li></ol></li>
<li>item 4
<ol><li>4.1</li></ol></li>
<li>item 5</li>
</ul>
```

```
// cache 4 et 5 :
$('li.trois ~ li').hide();

// cache 4 :
$('li.trois + li').hide();

// cache les <ol> :
$("ul ol").hide();

// ne cache rien :
$("ul > ol").hide();
```

Sélection d'éléments fondée sur la structure du DOM

Possibilité de sélectionner de manière plus précise :

- frère, enfants, parents
- utilisation de fonctions

```
// frère suivant
[sélecteur].next(expr)

// frère précédent
[sélecteur].prev(expr)

// frères
[sélecteur].siblings(expr)

// enfants
[sélecteur].children(expr)

// enfants y compris blocs de texte et commentaires
[sélecteur].contents(expr)

// parent
[sélecteur].parent(expr)
```

Autres sélecteurs

```
// premier paragraphe

p:first

// dernier élément de liste

li:last

// quatrième lien

a:nth(3) ou a:eq(3)

// paragraphes pairs ou impairs

p:even or p:odd

// tous les liens à partir (greater than)
// du quatrième ou avant (lower than)

a:gt(3) or a:lt(4)

// liens qui contiennent le mot click

a:contains('click')
```

Sélecteurs de visibilité

```
// si l'élément est visible

$("div:visible")

// sinon

$("div:hidden")
```

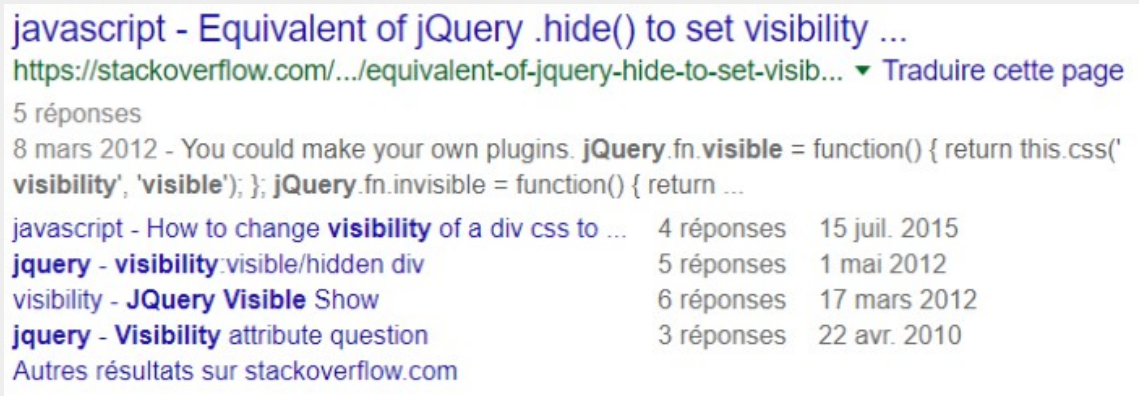
Point Javascript :

Suggestion de recherches internet :

jquery visibility

jquery visibility selector

site:jquery.com visibility



Si résultats sur StackOverflow : lire en diagonale les titres des sujets

Sélecteurs de formulaire

```
// sélectionner les cases à cocher
$("input:checkbox")

// sélectionner les boutons radio
$("input:radio")

// sélectionner les boutons
$(":button")

// sélectionner les champs texte
$(":text")

$("input:checked")

$("input:selected")

$("input:enabled")

$("input:disabled")
```

Formulaires : accès aux valeurs

Accès aux valeurs avec `.val()` sur les objets sélectionnés

Code HTML :

```
<select name="valeur">
<option value="1">1</option>
<option value="2" selected="selected">2</option>
<option value="3">3</option>
</select>
```

Code jQuery :

```
$("select option:selected").val()
```

Formulaires : modifieurs

Modification des valeurs avec `.val(nouvelleValeur)`

```
// obtenir la valeur de la première checkbox cochée

$("input:checkbox:checked").val();

// modifier la valeur d'un champ text de nom txt

$(":text[name='txt']").val("Hello");

// sélectionne une valeur d'un select d'identifiant
//lst

$("#lst").val("NS");
```

Modification des propriétés avec `.prop(propriété,valeur)`

```
// décocher les cases à cocher cochées

$("input:checkbox:checked").prop('checked',false);
```

Modifier le contenu HTML

***[sélecteur]*.html('[contenu]')** : remplacement du contenu d'un élément (les balises sont considérées comme des balises)

***[sélecteur]*.text('[contenu]')** : remplacement du contenu d'un élément en considérant le tout comme du texte (les caractères < et > des balises sont remplacés par les entités XML (> et <)

***[sélecteur]*.after('[contenu]')** : insertion du contenu après l'élément sélectionné

***[sélecteur]*.before('[contenu]')** : insertion du contenu avant l'élément sélectionné

***[sélecteur]*.append('[contenu]')** : insertion du contenu dans l'élément sélectionné à la suite des éléments existants

***[sélecteur]*.prepend('[contenu]')** : insertion du contenu dans l'élément sélectionné avant les éléments existants

Modifier le contenu HTML

***[sélecteur]*.wrap('<balise></balise>')** : insertion des balises passées en paramètre **de part et d'autre** de l'élément

***[sélecteur]*.wrapInner('<balise></balise>')** : insertion des balises passées en paramètre **de part et d'autre des enfants** de l'élément

***[sélecteur]*.unwrap()** : suppression de la balise parent

Possibilité de combiner les modifications les unes à la suite des autres :

```
$("div").html("Hello jQuery").wrapInner('<b></b>')
```

Attention à la lisibilité !

Possibilité de récupérer du contenu d'un autre objet pour le passer en paramètre :

```
$("div.a").html($("div.c").html());
```

→ met le contenu du `div.c` dans le `div.a`

Récupérer / modifier les attributs d'une balise

***[sélecteur]*.attr('attri')** : permet de récupérer la valeur de l'attribut `attri` du premier élément sélectionné par le sélecteur, récupère **undefined** si l'attribut `attri` n'est pas défini pour cet élément.

***[sélecteur]*.attr('attri','val')** : permet d'attribuer la valeur `val` à l'attribut `attri` du premier élément sélectionné par le sélecteur.

Code HTML :

```
<img src="logoUPEM.png" alt="logo.png">
```

Code jQuery :

```
// alerte qui affiche : logo
alert($("img").attr('alt'));
// changement de l'attribut alt : logo UPEM
$("img").attr('alt','logo UPEM');
```

Récupérer ou modifier les propriétés CSS

Récupération de la valeur de l'attribut CSS d'un élément :
***[sélecteur]*.css('color')** renvoie la couleur de l'élément

Attribution d'une valeur à l'attribut CSS d'un élément :
***[sélecteur]*.css('color','red')** attribue la couleur rouge

Attribution d'une valeur à l'attribut CSS des éléments de classe CSS «id» en fonction de leur valeur actuelle à l'aide d'une fonction :

```
var tailleActuelle = parseInt($('.js-bla').css("font-size"));
$('.js-bla').css("font-size",function(){
   return tailleActuelle+10;
});
```

augmente de 10 points la taille de police de caractères des éléments de classe « js-bla ».

Attribution de valeurs à un ensemble d'attributs CSS d'un élément :
***[sélecteur]*.css({'border':'1px solid black', 'color':'red'})** attribue la couleur rouge à l'élément et lui ajoute une bordure noire.

Récupérer ou modifier la classe CSS

***[sélecteur]*.addClass('cla')**

Ajoute la classe CSS `cla` à l'élément.

***[sélecteur]*.removeClass('cla')**

Retire la classe CSS `cla` à l'élément.

***[sélecteur]*.toggleClass('cla')**

Ajoute la classe CSS `cla` à l'élément s'il ne l'a pas, la lui retire sinon.

***[sélecteur]*.hasClass('cla')**

Est vrai si l'élément a la classe CSS `cla`, faux sinon.

→ Renvoie `true` si l'élément a la classe CSS `cla`, `false` sinon.

Répéter un traitement avec each

Appelle une fonction pour chaque élément sélectionné :

- `$(this)` : élément courant
- `i` : index de l'élément courant

```
$("table tr")
   .each(function(i){
     // On teste si i est pair,
     // c'est-à-dire si i modulo 2 est égal à 0 :
     if (i % 2 == 0)
       $(this).css("background-color","#FFDDDD");
   }
);
```

Événements https://api.jquery.com/category/events/

Premier événement :

`$(document).ready(…)` : quand le DOM est prêt

≠ `onLoad` en Javascript : quand tous les éléments sont chargés (images...)

Autres événements :

```
blur, focus, load, resize, scroll, unload,
beforeunload, click, dblclick, mousedown, mouseup,
mousemove, mouseover, mouseout, mouseenter,
mouseleave, change, select, submit, keydown,
keypress, keyup, error
```

Associer des événements (jQuery≥1.7)

```
// associer une fonction à un événement
$("div").on("click", function() {
    $(this).text("code HTML : "+$(this).html())
});

// arrêter d'exécuter l'événement
$("div").on("click", function() {
    $(this).text("code HTML : "+$(this).html());
    $("div").off("click")
});

// exécuter une seule fois (pour chaque objet)
$("div").one("click", function() {
    $(this).text("code HTML : "+$(this).html())
});
```

Point Javascript :

le **+** placé entre deux chaînes de caractères permet de **concaténer**, c'est-à-dire de coller l'une après l'autre deux chaînes de caractères.

Associer des événements dans le futur (jQuery≥1.7)

```
// attacher un événement même dans le futur

$(document).on("click", "div",
    function() {
    $(this).text("test" + $(this).html());
    }
)
```

→ On met trois paramètres en entrée de la fonction `on`

Répétition d'événements

Le même événement peut être créé plusieurs fois :

tous les événements seront exécutés.

```
<a href="">clic</a>

<script>

  $("a").click(function(event) {
    alert(event.type);
  });

  $("a").click(function(event) {
    alert(event.pageX + ", " + event.pageY);
  });

</script>
```

Attributs de l'objet `event`

`type` : nom de l'événement exécuté

`target` : objet qui a exécuté l'événement

`currentTarget` : = this

`pageX` et `pageY` : position de la souris

Autres : https://api.jquery.com/category/events/event-object/

```
altKey, bubbles, button, cancelable, charCode,
clientX, clientY, ctrlKey, currentTarget, data,
detail, eventPhase, metaKey, offsetX, offsetY,
originalTarget, pageX, pageY, relatedTarget,
screenX, screenY, shiftKey, target, view, which
```

Attributs de l'objet `event`

Exemple avec événement lié au déplacement de la souris, actif sur tout le document :

```
<div id="log"></div>

<script>

$(document).on('mousemove',function(e){
  $("#log").text(e.pageX + ", " + e.pageY);
});

</script>
```

Mise en pratique :

1) ajouter un élément d'id «log» dans la page
2) insérer le code ci-dessus
3) où se trouve le point (0,0) ?
4) selon le même principe, affichez les codes clavier correspondant aux touches pressées

Déclenchement automatique d'un événement

Fonction `trigger`

```
<button>#1</button>

<button>#2</button>

<div>

  <span>0</span> clics.

</div>

<div>

  <span>0</span> clics.

</div>
```

```
$("button:first").click(
  function () {
    update($("span:first"));
  });

$("button:last").click(
  function () {
    $("button:first")
      .trigger('click');
    update($("span:last"));
});

function update(j) {
  var n = parseInt(j.text(), 10);
  j.text(n + 1);
}
```

Blocage du comportement par défaut

Fonction `triggerHandler` pour ne pas exécuter le comportement par défaut

```
<button id="old">
  trigger
</button>

<button id="new">
  triggerH
</button>

<input type="text"
value="Focus"/>
```

```
$("#old").click(function(){
  $("input").trigger("focus");
});

$("#new").click(function(){
  $("input").
    triggerHandler("focus");
});

$("input").focus(function(){
  $("<p>Focus</p>").
    appendTo("body");
});
```

```
// empêcher le comportement par défaut
$('#close').click(function(e){
    e.preventDefault();
});

// similaire à :
$('#close').click(function(){
  return false;
});
```

Propagation des événements

Exemple : menu déroulant multi-niveaux.

Un clic se propage sur tous les objets associés :

• si on clique sur `<li> Niveau 3 : item 2</li>` alors on clique aussi sur le `<li>` du niveau 2 et celui du niveau 1.

• on a donc trois alertes.

• la propagation est « ascendante ».

```
$(document).ready(function() {
  $("li").click(function () {
    alert($(this).html());
  });
});
```

```
<ul>
<li> Niveau 1 : item 1</li>
<li> Niveau 1 : item 2
  <ul><li> Niveau 2 : item 1</li>
      <li> Niveau 2 : item 2
        <ul><li> Niveau 3 : item 1</li>
        <li> Niveau 3 : item 2</li></ul>
  </li></ul>
</li>
</ul>
```

Blocage de la propagation des événements

On peut stopper la propagation des événements :

- `stopPropagation();`
- ou faire `return false;` (attention cela peut bloquer d'autres choses)

```
$(document).ready(function () {
  $("li").click(function (e) {
    alert($(this).html());
    e.stopPropagation();
  });
});
```

Voir aussi :

- `isPropagationStopped`
- `stopImmediatePropagation`
- `isImmediatePropagationStopped`

Effets https://api.jquery.com/category/effects/

Apparition et disparition

```
// montrer un élément
$("div").show();

// montrer un élément lentement (slow=600ms)
$("div").show("slow");

// cacher un élément rapidement (fast=200ms)
$("div").hide("fast");

// inverser (montrer ou cacher) en une durée fixée
$("div").toggle(100);
```

Exemples :

```
$("div").hide("slow", function() {
    $("div").show("slow")});

$("a").click(function() {
  $("div").show("fast", function() {
    $(this).html("show div");});
});
```

Effet personnalisé

`.animate`(options, durée, transition, complete, ...) :

- options : ensemble de propriétés CSS.
- durée : durée en millisecondes
- transition : comment se déroule l'animation (linéaire ou pas).
- complete : callback exécuté après la fin de l'animation.
- … https://api.jquery.com/animate/

```
// réduction de la largeur à 90%,
// ajout d'une bordure bleue de largeur 5px et
// changement d'opacité. Le tout en 1s.

$("div").animate({
  width: "10%",
  opacity: 0.5,
  borderWidth: "5px"},
  1000);
```

Enchainement d'animations

Par défaut les animations sont effectuées l'une à la suite de l'autre.

Modifiable en utilisant `queue:false`.

```
// enchainement des animations

// modification du style, puis de la largeur
// et enfin de l'opacité
$("div")
.animate({border: "5px solid blue"},2000)
.animate({width: "20%"},2000)
.animate({opacity: 0.5},2000);

// animations simultanées

$("div")
.animate({border: "5px solid blue"},{queue:false,
         duration:100})
.animate({width: "20%"}, {queue:false, duration:2000})
.animate({opacity: 0.5},2000);
```

Charger du contenu depuis une page web

Pour mettre du contenu dans un objet :

• une possibilité plus simple qu'avec les fonctions génériques AJAX

• possibilité de ne charger qu'une partie du fichier (même si tout le fichier est récupéré dans ce cas, puis traité pour en extraire la partie voulue)

```
// version sans arguments :
// appelle fichier.html et met le contenu dans div

$("div").load("fichier.html");

// version avec arguments : appelle fichier.php
// en transmettant nom=philippe en POST

$("div#content").load("fichier.php",
{"nom":"philippe"});

// version ne récupérant que l'objet d'id monid

$("div").load('test.html #monid');
```

http://api.jquery.com/load/

Charger du contenu depuis une API

Un exemple d'API “maison” : API qui, à partir de l'identifiant d'un film dans Allociné, renvoie l'URL de l'affiche du film.

- à tester sur
http://igm.univ-mlv.fr/~gambette/ENSIUT/monique.pantel/apiAllocine.php?id=225

- renvoie :


```
{"id":225,"urlAffiche":"http:\/\/fr.web.img3.acsta.net\/c_
215_290\/medias\/nmedia\/18\/73\/62\/07\/19209359.jpg"}
```


Utilisation en jQuery :

```
$.get("http://igm.univ-
mlv.fr/~gambette/ENSIUT/monique.pantel/apiAllocine.php?
id=225").done(function(data){
   console.log(data.id+" "+data.urlAffiche);
   // affiche le numéro 225 et l'adresse de l'affiche
   // dans la console
})
```

Deux options pour organiser son code

1) Mettre le code jQuery
dans le code HTML

Fichier `index.html`

```
<html>
<head><title>Mon site</title>
</head>
<SCRIPT TYPE="text/javascript"
SRC="https://ajax.googleapis.com/
ajax/libs/jquery/2.1.3/jquery.min
.js"></SCRIPT>
<SCRIPT TYPE="text/javascript">
$(document).ready(function(){
   alert($("title").html());
})
</SCRIPT>
</html>
```

2) Mettre le code jQuery
dans un fichier `code.js` séparé

Fichier `index.html`

```
<html>
<head><title>Mon site</title>
</head>
<SCRIPT TYPE="text/javascript"
SRC="https://ajax.googleapis.com/
ajax/libs/jquery/2.1.3/jquery.min
.js"></SCRIPT>
<SCRIPT TYPE="text/javascript"
SRC="./js/code.js"></SCRIPT>
</html>
```

Fichier `code.js` dans le dossier `js`

```
$(document).ready(function(){
   alert($("title").html());
})
```