jQuery

Pour Web Designers et Web Developpers

Mulkay Alain

1 Introduction

1.1 Présentation du cours

Le cours jQuery est composé de plusieurs modules :

- 1 module de 3 jours consacré aux bases de jQuery et judicieusement appelé …jQuery Base
- 1 module de 4 jours orienté Designer (jQuery Design) reprenant la base plus toutes une série de méthodes orientées design
- 1 module de 2 jours consacré à quelques spécificités plus élaborées, module appelé jQuery avancé, ces spécificités sont :
    - Les événements vus d'une manière approfondie (l'objet event, la propagation événementielle, inhibition du traitement par défaut d'un évènement)
    - AJAX
    - La bibliothèque jQuery UI
- 1 module de 4 jours (jQuery Developper) reprenant plus en détails la matière de jQuery avancé avec en plus :
    - Création d'un plug-in

L'appartenance d'un chapitre à la matière de jQuery avancé sera précisée par l'ajout de ''(jQuery avancé)'' derrière le titre du chapitre

1.2 Objectif du cours

A l'issue du cours jQuery module de base , vous aurez acquis une 1ère expérience de l'interactivité entre vos projets Web et la bibliothèque jQuery.

Vous serez capable, entre autres :

- de charger (load) et d'installer la bibliothèque jQuery pour pouvoir l'utiliser au sein de vos pages Web
- d'intégrer vos nouveaux scripts de manière optimale.
- de manipuler le contenu de vos documents
- de valider des données de formulaire
- de créer des effets d'animation
- d'intégrer des plug-ins tout fait

A l'issue du cours jQuery avancé, vous aurez approfondi l'utilisation de la bibliothèque jQuery.

Vous serez capable, entre autres :

- d'implémenter une gestion évènementielle avancée
- de réaliser des requêtes au serveur et de manipuler différents format de réponse en utilisant les techniques AJAX
- de mettre en place une interface utilisateur avancé basé sur la bibliothèque jQuery UI
- de créer un plug-in – (jQuery Developper)

1.3 Prérequis

1.3.1 Module de base

- Avoir une bonne maîtrise de l' (X) HTML
- Avoir une bonne maîtrise des CSS
- Connaître le principe du DOM
- Connaître les bases de JavaScript

1.3.2 Module avancé

- Avoir une bonne maîtrise de l' (X) HTML
- Avoir une bonne maîtrise des CSS
- Connaître le principe du DOM
- Connaître les bases de JavaScript
- Avoir une expérience en jQuery

  Ou avoir suivi le module de base

1.4 Définition de jQuery

1.4.1 Wikipedia

jQuery est une bibliothèque JavaScript libre qui porte sur l'interaction entre JavaScript (comprenant AJAX) et HTML, et a pour but de simplifier des commandes communes de JavaScript.

1.4.2 Mais encore. (Version simplissime)

Ensemble de fonctions JavaScript, enregistrées dans un fichier .js externe permettant d'effectuer toute une série d'action via une écriture de code beaucoup plus compacte

2 Charger et installer la bibliothèque jQuery

2.1 Introduction

Il existe 2 méthodes pour pouvoir profiter des multiples fonctions de jQuery au sein de vos pages web

- La 1ère consiste à pointer directement, donc sans la copier dans le sous-dossier de votre site web, vers la bibliothèque jQuery mise à votre disposition par le site Google
- La 2ème consiste à copier après l'avoir chargée ("loader"), la bibliothèque jQuery dans un sous-dossier de votre site web et de pointer ensuite vers ce fichier au moyen de l'élément html script défini par les balises <script …..> </script>

L'idéal est de combiner ces 2 méthodes

2.2 1ère partie : Lien vers la bibliothèque jQuery de Google

Vous trouverez le lien vers la bibliothèque hébergé par Google à l'adresse qui suit :

https://developers.google.com/speed/libraries/devguide?hl=fr#jquery

2.2.1 Inclure la bibliothèque jQuery

- Copiez l'élément script suivant juste avant la balise de fermeture </body>
- Dans l'exemple ci-dessous, 1.9.1 est le n° de version de la bibliothèque jQuery , ce n° de version est à adapter en fonction de la bibliothèque utilisée.

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" >

</script>
```

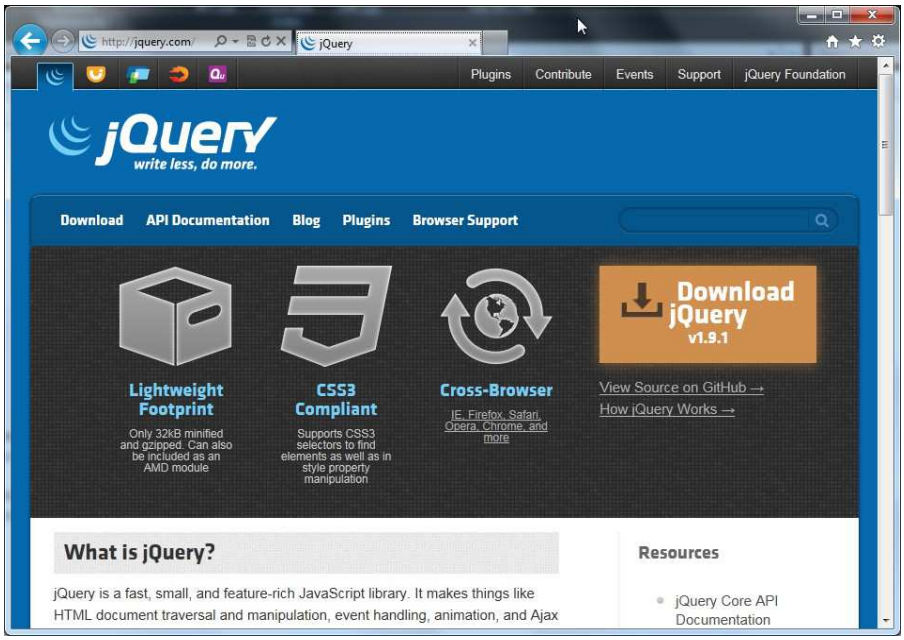
2.2.1 L'élément script au sein d'une page .html

```
<html lang="fr">
   <head>
      <meta  charset=utf-8"/>
      <title>index</title>
      <link  ... href="css/main.css" >
      <link ... href="css/printer.css" >
   </head>
   <body>
      ...
    <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" >
    </script>
   </body>
</html>
```

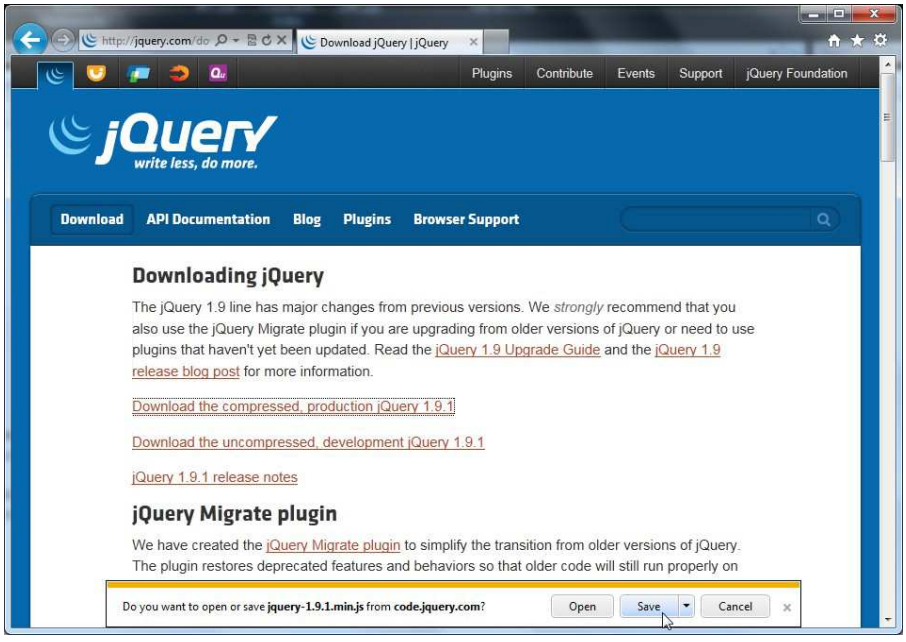
2.3 2ème Partie : Copie de la bibliothèque jQuery

2.3.1 Charger la bibliothèque jQuery

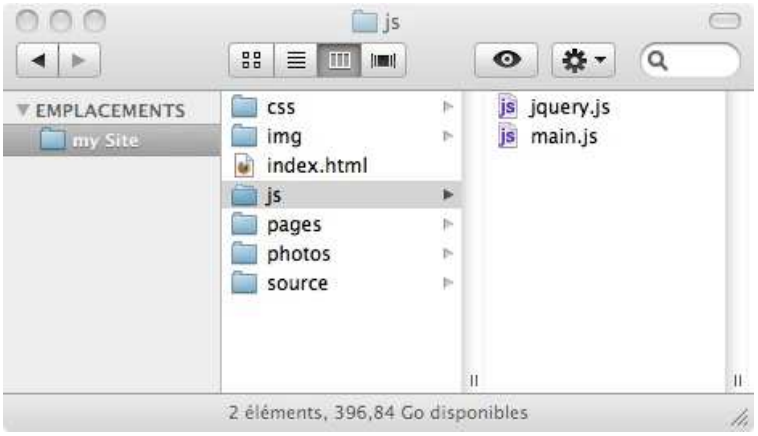
- Tapez dans votre navigateur favori, l'u.r.l. suivante : http://jquery.com/



- Dans la fenêtre qui vient de s'ouvrir, cliquez sur "Download jQuery " (ici pour l'exemple, la version 1.9.1)
- Dans la nouvelle page, cliquez sur *Download the compressed, production jQuery v x.x.x (ici v 1.9.1)*
- Enregistrez le fichier *jQuery-1.9.1.min.js* dans le sous-dossier contenant les fichiers JavaScripts de votre site (ici le dossier js)
- Pour ne pas devoir réadapter la doc tous les mois, j'ai renommé le fichier *jQuery-1.9.1.min.js* en *jQuery.js.*



Pour les besoins du cours, nous utiliserons un dossier site (mySite) dont l'arborescence est reprise ci-dessous.



2.3.2 Inclure la bibliothèque jQuery

- Créez un élément script juste avant la balise de fermeture </body>
- Copier dans l'élément script le code JavaScript suivant :

```
<script>
   window.jQuery || document.write('<script src="js/jquery.js" > <\/script>');
</script>
```

- Le chemin relatif de votre fichier jquery.js se fait par rapport à votre page .html
- Rajoutez un 2ème élément script contenant lui l'emplacement de votre fichier .js (ici main.js), ce fichier contiendra votre propre code JavaScript. Cet élément script doit venir impérativement APRES l'élément script chargeant la bibliothèque jQuery.

2.3.3 L'élément script au sein de la page index .html

```
<html lang="fr">
   <head>
      <meta  charset=utf-8"/>
      <title>index</title>
      <link  ... href="css/main.css" >
      <link ... href="css/printer.css" >
   </head>
   <body>
      ...
     <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" >
     </script>
     <script>
      window.jQuery || document.write('<script src="js/jquery.js" ><\/script>');
     </script>
     <script src="js/main.js"> </script>
   </body>
</html>
```

2.3.4 L'élément script au sein des autres pages .html situées dans le sous-dossier pages

```
<html lang="fr">
   <head>
      <meta  charset=utf-8"/>
      <title>index</title>
      <link  ... href="css/main.css" >
      <link ... href="css/printer.css" >
   </head>
   <body>
      ...
     <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js" >
     </script>
     <script>
        window.jQuery || document.write('<script src="../js/jquery.js" ><\/script>');
     </script>
      <script src="../js/main.js" > </script>
   </body>
</html>
```

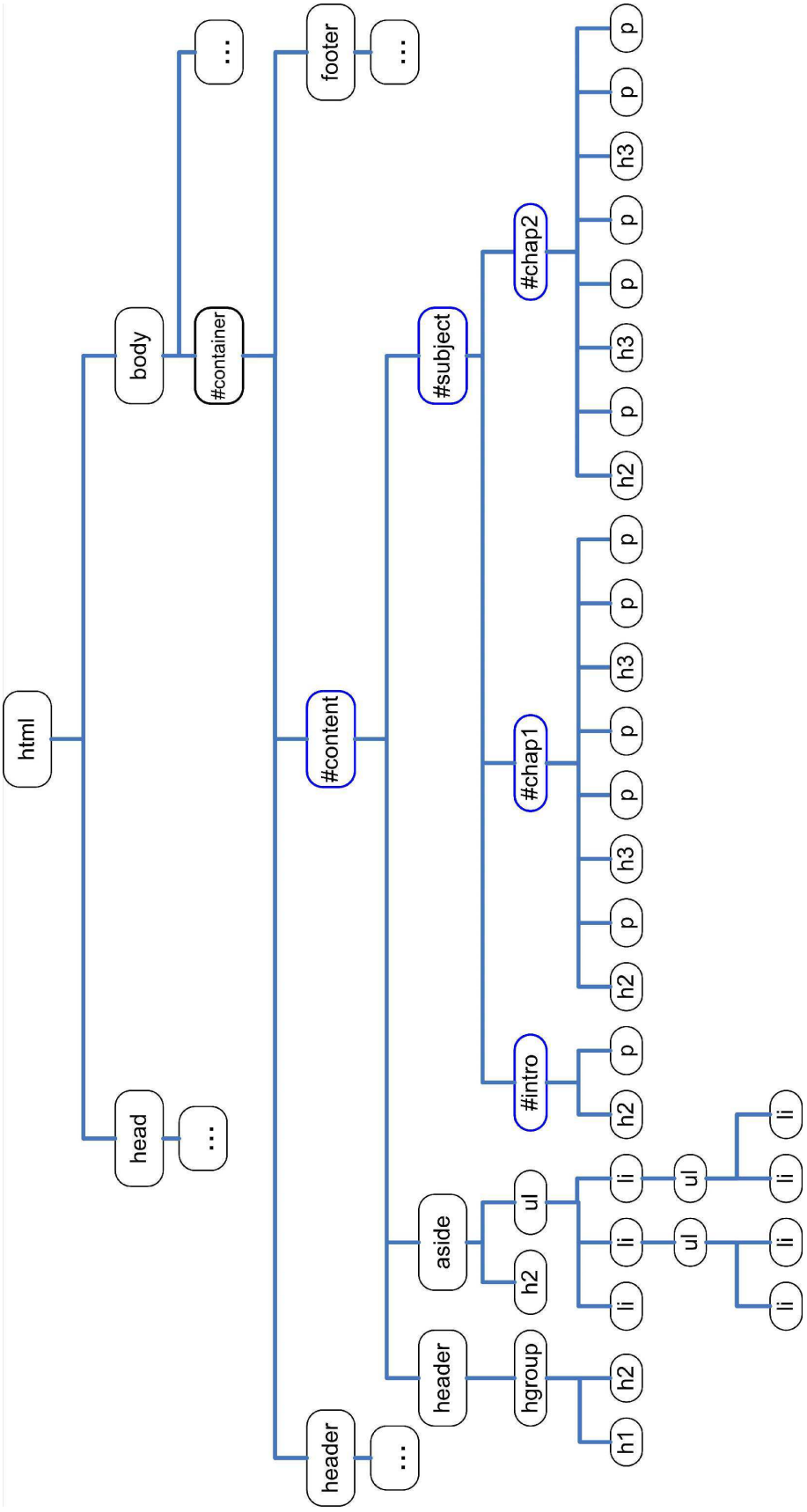
3 Les composants de jQuery

3.1 Extrait d'une page html

Voici le code html qui va nous servir pour illustrer les différentes méthodes de jQuery

```
<html>
  <head>
  </head>
  <body>
     …
     <section id="content">
        …
        <aside>
           <h2>Table des Matières</h2>
            <ul>
              <li><a href="#intro">Intro</a> </li>
              <li><a href="#chap1">Chap.1</a>
                <ul>
                  <li>1A</li>
                  <li>1B</li>
                </ul>
              </li>
              <li><a href="#chap2">Chap.2</a>
                <ul>
                   <li>2A</li>
                   <li>2B</li>
                </ul>
               </li>
             </ul>
        </aside>
        <section id="subject" >
           <div id="intro" >
            <h2> Introduction </h2>
            <p> Lorem ipsum Intro... </p>
           </div><!--     End #intro -->
           <div id="chap1" >
             <h2> Chapitre 1 </h2>
             <p> Lorem ipsum Intro Chap 1... </p>
             <h3>§ 1A</h3>
             <p> Lorem ipsum Intro § 1A... </p>
             <p> Lorem ipsum Contenu 1A...  </p>
             <h3>§ 1B</h3>
             <p> Lorem ipsum Intro § 1B... </p>
             <p> Lorem ipsum Contenu 1B...  </p>
           </div><!--     End #chap1 -->
           <div id="chap2" >
             <h2> Chapitre 2 </h2>
             <p> Lorem ipsum Intro Chap 2... </p>
             <h3>§ 2A</h3>
             <p> Lorem ipsum Intro § 2A... </p>
             <p> Lorem ipsum Contenu 2A...  </p>
             <h3>§ 2B</h3>
             <p> Lorem ipsum Intro § 2B... </p>
             <p> Lorem ipsum Contenu 2B...  </p>
           </div><!--     End #chap2   -->
        </section><!--    End Section subject -->
     </section><!--   End Section content       -->
     …
   </body>
</html>
```

3.2 Représentation de l'extrait sous forme de DOM



3.3 Enfin du Code jQuery

3.3.1 Exemple Type

```
$(document).ready( function (){
        $(' #tdm').click( function () {
                $('#tdm').slideUp('slow');
        })
})
```

3.3.2 La fonction jQuery

- jQuery ne contient qu'une seule fonction pouvant recevoir un paramètre

  $(‘ paramètre’)

  Ce paramètre peut être :

  - un sélecteur => $('div p')
  - un élément DOM => $(this)
  - un array composé d'élément DOM => $(document.getElementByTag('p'))
  - un objet jQuery
  - du code HTML destiné à être injecté dans la page => $('<h1> Nouveau Titre</h1>).appendTo('#content')

- Cette fonction retourne un objet jQuery qui n'est en réalité qu'un ensemble, un set d'éléments html (une collection d'éléments DOM pour être très très précis) définit par le sélecteur passé en paramètre

  $(‘ sélecteur ’) => objet jQuery

- L'objet jQuery renvoyé, même s'il y ressemble fort n'est pas un tableau javascript (un array () d'éléments DOM)

  $(‘ p’) => objet jQuery =>

- Le résultat étant un objet, celui-ci possède donc des méthodes et des propriétés

3.3.3 Les méthodes jQuery

- méthodes d'accès de parcours du DOM

  ex: $('sélecteur').children()

- méthodes de manipulation du DOM

  ex: $('sélecteur').addClass()

- méthodes événementielles

  ex: $('sélecteur').click()

- méthodes d'animations

  ex: $('sélecteur').show()

- méthodes AJAX

  ex: $('sélecteur').load()

- le reste …

  ex: $('sélecteur').get(n)

3.3.4 Les propriétés jQuery

- propriétés globales

  ex: $.browser

- propriétés objet jQuery

  ex: $('sélecteur').length

4 Les Sélecteurs jQuery

4.1 Les sélecteurs CSS

Nous retrouvons comme paramètre de notre fct jQuery , exactement les mêmes sélecteurs que ceux utilisés pour les css 1, 2, 2.1 et 3

Type de sélecteurs	Exemple	Eléments HTML ‘’capturés » par le sélecteur
*	*	Tous les éléments
#id	#firstname	L’élément dont l’identifiant (id) ="firstname"
.class	.intro	Tous les éléments de class="intro"
élément	p	Tous les éléments <p>
élément#id	div#header	L’élément dont l’identifiant (id) = "header"
élément.class	div.chapter	Tous les éléments de class="chapter"
élément , élément	div,p	Tous les éléments et tous les éléments <p>
élément élément	div p	Tous les éléments <p> descendants d’éléments
élément > élément	div>p	Tous les éléments <p> dont le parent est un élément
élément + élément	div+p	Tous les éléments <p> situé directement après élément (ne sont pas inclus dans l’élément)
élément~élément	div~p	Tous les éléments <p> frères des éléments et qui suivent celles-ci
élément :first-child	li:first-child	Sélectionne tous les qui sont les 1er éléments de leur liste
élément :last-child	li:last-child	Sélectionne tous les qui sont les derniers éléments de leur liste
élément :only-child	a:only-child	Sélectionne tous les <a> qui sont les seuls éléments de leurs parents
élément :nth-child(n)	p:nth-child(2)	Sélectionne tous les <p> qui sont les seconds enfants de leurs parents. !!!!!!!Le parent peut avoir d'autres enfants que des <p>
élément :nth-child(odd)	tr:nth-child(odd)	Sélectionne toutes les lignes impaires d’une table. !!!!! commence à 1
élément :nth-child(even)	tr:nth-child(even)	Sélectionne toutes les lignes paires d’une table !!!!! commence à 1
élément :nth-child(Xn+Y)	p:nth-child(3n+1)	Sélectionne les éléments <p> qui sont les 1er ,4ème, 7ème etc.. enfants de leurs parents (n=0, 1, 2, 3,4 etc.)
élément :not(sélecteur)	div:not(.Contenu)	Sélectionne toutes les qui ne sont pas de class="Contenu"
élément :empty	p:empty	Sélectionne tous les <p> qui n'ont pas d'enfants (dont les balises n’encadrent rien)

4.2 Les sélecteurs d'attributs

Type de sélecteurs	Exemple	Eléments HTML ‘’capturés par le sélecteur
[attribute]	[target]	Tous les éléments avec un attribut de type target
[attribute=value]	[title=flower]	Tous les éléments avec un attribut title = "flower"
[attribute!=value]	[title!=flower]	Tous les éléments dont l'attribut title est différent de "flower"
[attribute^=value]	[title^="flo"]	Tous les éléments où l’attribut title commence par le prefixe "flo"
[attribute$=value]	[title$="flo"]	Tous les éléments où l’attribut title termine par le suffixe "flo"
[attribute*=value]	[title*="flo"]	Tous les éléments où l’attribut title contient au minimum la chaine de caractère "flo"
[attribute~=language]	[title~=flower]	Tous les éléments dont l'attribut title contient le mot "flower" (entouré d'espace)
[attribute|=language]	[lang|=en]	Tous les éléments où l’attribut lang vaut "en", même s’il est suivi ou précédé d’un tiret (-), comme "en-us"

4.3 Les sélecteurs de formulaires

Type de sélecteurs	Eléménts de formulaire ‘’capturés’’ par le sélecteur
:button	Renvoie un set d’élément <button> et <input type= ‘’button’’>
:checkbox	Renvoie un set d’élément type checkbox
:checked	Renvoie un set de checkbox et de boutons radio cochés
:enabled	Renvoie un set d’élément de formulaire activés
:disabled	Renvoie un set d’élément de formulaire désactivés
:file	Renvoie un set d’élément <input type=’’file’’>
:image	Renvoie un set d’élément <input type=’’image’’>
:input	Renvoie un set d’élément <input> <select> <textarea> et<button>
:password	Renvoie un set d’élément <input type=’’password’’>
:radio	Renvoie un set d’élément type boutons radio
:reset	Renvoie un set d’élément <input type=’’reset’’>
:selected	Renvoie un set d’élément <options> sélectionnés
:submit	Renvoie un set d’élément <input type=’’submit’’>
:text	Renvoie un set d’élément <input type=’’text’’>

4.4 Les sélecteurs personnalisés

Type de sélecteurs	Exemple	Eléménts de formulaire ''capturés'' par le sélecteur
élément :first	p:first	Sélectionne le 1er <p> de la page
élément :last	p:last	Sélectionne le dernier <p> de la page
élément élément :first	div p:first	Sélectionne les 1er <p> des dont ils sont descendants
élément :odd	li:odd	Sélectionne tous les impairs de toutes les listes de la page !!!!! commence à 0
élément :even	.header:even	Sélectionne les éléments pairs de classe= ''header''. !!!!! commence à 0
élément :eq(n)	.header:eq(4)	Sélectionne le 5ème élément de classe= ''header''.
élément :gt(n)	.header :gt(4)	Sélectionne tous les éléments de classe= ''header''qui suivent le 5ème
élément :lt(n)	.header :lt(4)	Sélectionne les 5 premiers élément de classe= ''header''.
:contains(texte)	li:contains(Trois)	Sélectionne tous les qui contiennent le texte "Trois" !!!!!! Tient compte des Majuscules
:parent	li:parent	Sélectionne tous les qui ont des nœuds enfants y compris du texte
:has(élément)	.intro:has('li')	Sélectionne tous les éléments de classe intro qui ont des comme descendants
:header	div:header	Sélectionne tous les éléments de type <h1> <h2> <h3> <h4> <h5> <h6> qui sont inclus dans une
:animated	#subject:animated	Sélectionne tous les éléments en cours d'animation qui sont les descendants d'un élément d'Id = "subject"
:hidden		Sélectionne tous les éléments qui sont "cachés" : les éléments dont display : none les éléments <input type="hidden" > les éléments dont width et heigth = 0 les éléments dont un ancêtre correspond à un des 3 cas précédant
:visible		Sélectionne tous les éléments qui ne correspondent pas aux cas précédents (:hidden)

5 Les méthodes de jQuery

5.1 Remarque :

Les lignes du tableau qui sont grisées contiennent des méthodes obsolètes à ne plus urtilisées. Elles sont signalées pour pouvoir les remplacer si vous les rencontrées dans du code que vous devez updater

5.2 Les méthodes de parcours du DOM

Ces méthodes appliquées à l'obet jQuery vont permettre de créer un nouvel objet jQuery (un nouveau set) dont les éléments correspondent au sélecteur. Le terme ''set'' dans le tableau ci-dessous désigne l'ensemble des éléments de l'objet jQuery auquel est appliqué la méthode .

set.méthode(sélecteur) => nouvel objet jQuery

Méthode	Renvoie un objet jQuery contenant
.filter (sélecteur)	les éléments sélectionnés qui correspondant au sélecteur
.filter (fct(){})	les éléments sélectionnés pour lequel la fct renvoie true
.eq (n)	l'élément sélectionné qui se trouve à l'indice n du set (commence à 0)
.first()	le 1er élément du set
.last()	le dernier élément du set
.slice (n, [m])	Les éléments sélectionnés qui se trouvent dans la plage d'indice n, m (commence à 0)
.not (sélecteur)	les éléments sélectionnés qui ne correspondent pas au sélecteur
.has(sélecteur)	Les éléments qui ont "sélecteurs" comme descendant
.find (sélecteur)	Les éléments descendants qui correspondent au sélecteur
.children ([sélecteur])	Les éléments enfants (avec filtrage facultatif par le sélecteur)
.parents ([sélecteur])	Tous les ancêtres
.parentsUntil ([sél..])	Tous les ancêtres jusqu'au sélecteur non compris
.parent ([sélecteur])	Le parent de chaque élément sélectionné
.closest (sélecteur)	Le premier élément qui correspond au sélecteur, en partant de l'élément sélectionné et en remontant l'arborescence du DOM
.offsetParent ()	Le parent le plus proche ayant une CSS de position
.siblings ([sélecteur])	Tous les frères
.next ([sélecteur])	Le frère qui vient immédiatement après chaque élément sélectionné
.nextAll ([sélecteur])	Tous les frères qui suivent chaque élément sélectionné
.nextUntil (sélecteur)	Tous les frères qui suivent chaque élément sélectionné jusqu'au sélecteur non compris
.prev ([sélecteur])	Le frère qui précède immédiatement chaque élément sélectionné

.prevAll ([sélecteur])	Tous les frères qui précèdent chaque élément sélectionné
.prevUntil ([sélecteur])	Tous les frères qui précèdent chaque élément sélectionné jusqu'au sélecteur non compris
.add (sélecteur)	les éléments sélectionnés plus les éléments correspondant au sélecteur
.is (sélecteur)	Renvoie "true" si le sélecteur est compris dans l'objet jQuery
.end ()	Voir exemple
.andSelf ()	Voir exemple
.each(fct(){})	Itère (boucle) sur le set défini par le sélecteur et exécute la fonction fct sur chaque élément renvoyé
.map (fct(){})	Applique une fct à tous les éléments du set et renvoie un nouvel objet jQuery avec les modifications apportées par la fct
.contents ()	Les éléments enfants, y compris les éléments textes

Exemples :

andSelf()

$('td :contains(Hamlet)').nextAll().addClass('selected')

ajoute la classe .selected à tous les éléments frères de la cellule contenant le texte 'Hamlet '

$('td :contains(Hamlet)').nextAll().andSelf().addClass('selected')

ajoute en plus la classe .selected à la cellule contenant le texte 'Hamlet '

end()

$('ul.first').find('.one).css('color','red').end().find(.two).css('color','green')

récupère les éléments de class="one" dans les de class='first' et mets leurs fontes en rouge , la méthode .end() permet de retrouver le set avant filtrage par find(), c'est à dire les de class='first' pour y sélectionner les éléments de class="two" et mettre leurs fontes en vert

map()

```
$('.checkbox').map(function () {
        return this.checked = true;
})
```

coche (check) toutes les cases à cocher d'un formulaire

5.3 Les méthodes d'événements

Méthode (E = $('sél. '))	Description
$('document').ready (*fct*)	*fct* sera invoqué lorsque le DOM et les CSS seront intégralement chargés
E.on(event,fct)	Nouvelle méthode remplaçant bind et live à partir de jQuery 1.7
E.off(event,fct)	Nouvelle méthode remplaçant unbind et die à partir de jQuery 1.7
E.bind(event,fct)	*fct* sera invoqué quand l'événement *event* sera envoyé à E
E.unbind(event)	Retire le gestionnaire *event* de E
E.live(event,fct)	idem bind mais ajoute *event* également aux éléments correspondants à E rajoutés après que la méthode live ait été instanciée
E.die(event)	idem unbind mais pour .live ()
E.one(event,fct)	*fct* sera invoqué 1 seule fois quand l'événement *event* sera envoyé à E
E.trigger(event)	Simule l'exécution de *event*
E.load(fct)	*fct* sera invoqué lorsque E (img) sera entièrement chargé
$(window).unload(fct)	*fct* sera invoqué lorsque l'utilisateur quittera la page contenant ce code
E.error(*fct*)	*fct* sera invoqué lorsque E déclenchera une erreur
E.blur(*fct*)	*fct* sera invoqué lorsque E perdra le focus
E.change(*fct*)	*fct* sera invoqué lorsque E perdra le focus et que sa valeur aura changée
E.click(*fct*)	*fct* sera invoqué lorsque d'un clic sur E
E.dblclick(*fct*)	*fct* sera invoqué lorsque d'un double-clic sur E
E.focus(*fct*)	*fct* sera invoqué lorsque E recevra le focus
E.keydown(*fct*)	*fct* sera invoqué lorsque une touche sera enfoncée alors que E a le focus
E.keypress(*fct*)	*fct* sera invoqué lorsque une touche sera touchée alors que E a le focus
E.keyup(*fct*)	*fct* sera invoqué lorsque une touche sera relachée alors que E a le focus
E.load(*fct*)	*fct* sera invoqué lorsque le chargement de E sera terminé (img ou body)
E.mousedown(*fct*)	*fct* sera invoqué lorsque le btn de la souris sera enfoncée au-dessus de E
E.mousemove(*fct*)	*fct* sera invoqué lorsque le pointeur de la souris survolera E
E.mouseout(*fct*)	*fct* sera invoqué lorsque le pointeur de la souris quittera E
E.mouseover(*fct*)	*fct* sera invoqué lorsque le pointeur de la souris entrera dans E
E.mouseup(*fct*)	*fct* sera invoqué lorsque le btn de la souris sera relaché au-dessus de E
E.resize(*fct*)	*fct* sera invoqué lorsque E sera redimensionné

E.scroll(*fc*t)	*fct* sera invoqué lorsque la position de défilement de E changera
E.select(*fc*t)	*fct* sera invoqué lorsque le texte dans E sera sélectionné
F.submit(*fc*t)	*fct* sera invoqué lorsque le formulaire F sera envoyé
E.hover(*fc*t1, *fc*t2)	*fct* 1 sera invoqué lorsque la souris entrera dans E et *fct2* lorsque la souris en sortira
E.toggle(*fc*t1, *fc*t2,…)	*fct1* sera invoqué lorsque du 1er clic de la souris sur E, *fct2* lors du 2ème et ainsi de suite

Exemples :

On, bind et live:

```
$('li').bind ('click,function(){
        $('tr:odd').addClass('impair');
});
```

Ajoute la class ="impair" à toutes les lignes (tr) impaires des tables lorsque l'on clique sur un élément li => si un nouvel élément li est rajouté dynamiquement dans la page, celui –ci ne sera pas cliquable

Equivalent avec la méthode *on* : l'élément li doit être défini seul (pas défini par rapport à un ancêtre ou un parent)

```
$('li').on ('click,function(){
        $('tr:odd').addClass('impair');
});
```

Pour que les éléments li à venir deviennent cliquables et déclenchent la même fonction , nous utiliserons la méthode *live*

```
$('li').live ('click,function(){
        $('tr:odd').addClass('impair');
});
```

Equivalent avec la méthode *on*: l'élément li doit être défini par rapport à un ancêtre ou un parent

```
$('ul ').on ('click','li',function(){
        $('tr:odd').addClass('impair');
});
```

5.4 Les méthodes de manipulation du DOM

Méthode	Description
.attr('X')	récupère la valeur de l'attribut X
.attr('X', 'txt')	affecte la valeur 'txt' à l'attribut X
.attr('X', fct)	affecte à l'attribut X le résultat de là fonction *fct*
.attr(map)	fixe les valeurs des attributs sous la forme : {'nom1' :'valeur1','nom2' :'valeur2,etc…}
.removeAttr('X')	supprime l'attribut X
.addClass('C')	ajoute la classe C au set d'élément
.removeClass('C')	enlève la classe C au set d'élément
.toggleClass('C')	enlève la classe C si elle est présente sinon l'ajoute au set d'élément
.hasClass('C')	retourne *true* si l'un des éléments possède la classe C
.css('X')	récupère la valeur de l'attribut css X
.css('X', 'txt')	affecte la valeur 'txt' à l'attribut css X
.css(map)	fixe les valeurs des attributs sous la forme : {'nom1' :'valeur1','nom2' :'valeur2,etc…}
.height()	renvoie la hauteur du 1er élément sélectionné
.innerHeight()	renvoie la hauteur (élément + padding) du 1er élément sélectionné
.outerHeight([bool])	renvoie la hauteur (élément + padding + border) du 1er élément sélectionné. Si bool = true margin est inclus dans le résultat
.height('n')	attribue la hauteur 'n' à tous les éléments sélectionnés
.width()	renvoie la largeur du 1er élément sélectionné
.innerWidth()	renvoie la largeur (élément + padding) du 1er élément sélectionné
.outerWidth([bool])	renvoie la largeur (élément + padding + border) du 1er élément sélectionné. Si bool = true margin est inclus dans le résultat
.width('n')	attribue la largeur 'n' à tous les éléments sélectionnés
.offset()	Renvoie un objet avec les attributs top et left du 1er élément sélectionné (coord calculées par rapport au document)
.offset(cordinates)	attribue les coordonnées top et left à tous les éléments sélectionnés
.position()	Renvoie un objet avec les attributs top et left du 1er élément sélectionné (coord calculées par rapport à l'élément parent)
.html()	récupère le contenu html du 1er élément sélectionné
.html('contenu')	remplace le contenu html de chaque élément sélectionné par 'contenu'
.text()	récupère le contenu texte de tous les élément sélectionnés dans une seule chaine de caractère
.text('nouveau texte')	remplace le contenu texte de chaque élément sélectionné par 'nouveau

	texte’
.val()	récupère la valeur de l’attribut *value* du 1er élément sélectionné
.val(‘txt’)	remplace la valeur de l’attribut *value* de chaque élément sélectionné par ‘txt’
.append(‘contenu’)	insère ‘contenu’ à la fin de chaque élément sélectionné
.appendTo(‘sélecteur’)	insère les éléments sélectionnés à la fin de chaque élément correspondant au sélecteur
.prepend(‘contenu’)	insère ‘contenu’ au début de chaque élément sélectionné
.prependTo(‘sélecteur’)	insère les élément sélectionnés au début chaque élément correspondant au sélecteur
.after(‘contenu’)	insère ‘contenu’ après chaque élément sélectionné
.insertAfter(‘sélecteur’)	insère les éléments sélectionnés après chaque élément correspondant au sélecteur
.before(‘contenu’)	insère ‘contenu’ avant chaque élément sélectionné
.insertBefore(‘sélecteur’)	insère les éléments sélectionnés avant chaque élément correspondant au sélecteur
.wrap(‘HTML’)	enveloppe chaque élément sélectionné avec le code HTML
.wrapAll(‘ HTML ’)	enveloppe tous les éléments sélectionnés avec le code HTML
.wrapInner(‘ HTML ’)	enveloppe le contenu de chaque élément sélectionné avec le code HTML
.unwrap()	Supprime le parent de chaque chaque élément sélectionné
.replaceWith(‘contenu’)	remplace les éléments sélectionnés par ‘contenu’
.replaceAll(‘sélecteur’)	remplace les éléments correspondant au sélecteur par les éléments sélectionnés
.empty()	retire les éléments enfants de chaque élément sélectionné
.remove([sélecteur])	retire du DOM les éléments sélectionnés (avec filtrage facultatif par le sélecteur)
.clone ([true false])	effectue une copie de tous les éléments sélectionnés en incluant facultativement les gestionnaires d’événements

5.5 Les méthodes d'effets

Méthode(E = $('sél. ')	Description
E.show([t],[fct])	Affiche E après un temps t (optionnel) ensuite exécute *fct* (opt.)
E.hide([t],[fct])	Cache E après un temps t (optionnel) ensuite exécute *fct* (opt.)
E.toggle([t],[fct]) ou (bool)	1.Alterne .show() et .hide() . 2. Si bool = true => affiche , si bool = false => cache
E.slideDown([t],[fct])	La hauteur de E varie de 0 à h (hauteur normale de E) durant un temps t (optionnel) ensuite exécute *fct* (opt.)
E.slideUp([t],[fct])	La hauteur de E varie de h à 0 durant un temps t (optionnel) ensuite exécute *fct* (opt.)
E.slideToggle([t],[fct])	Alterne .slideDown() et .slideUp()
E.fadeIn([t],[fct])	L'opacité de E varie de 0 à N (opacité normale de E) durant un temps t (optionnel) ensuite exécute *fct* (opt.)
E.fadeOut([t],[fct])	L'opacité de E varie de N à 0 durant un temps t (optionnel) ensuite exécute *fct* (opt.)
E.fadeTo(t,X[,fct])	L'opacité de E varie de N à X durant un temps t (obligatoire) ensuite exécute *fct* (opt.)
E.animate()	Voir exemple

5.6 Les autres méthodes

Ces méthodes ne renvoient pas d'objet jQuery

Méthode	Description
$('sélecteur').size()	Renvoie le nombre d'éléments défini par le sélecteur
$('sélecteur').get(n)	Renvoie l'élément qui se trouve à l'index n+1 dans le set défini par le sélecteur
$('sélecteur').index([E])	Renvoie l'index de E dans le set défini par le sélecteur (-1 si pas trouvé)
$.each(collection,fct)	Idem, la collection pouvant être une map ou un Array , dans ce cas la fct prent 2 paramètres : l'index et la valeur de la case de l'array
$.grep (array,fct,bool)	Exécute une fonction de filtre sur chaque case de l'array et renvoie un nouvel array après filtrage, si bool =true, l'ordre est inversé
$.trim(string)	Renvoie le string après avoir retiré les espaces au début et à la fin du string

6 jQuery et Ajax (jQuery avancé)

6.1 Qu'est ce qu'AJAX ?

6.1.1 Wikipedia

AJAX est un acronyme signifiant Asynchronous JavaScript and XML (« XML et JavaScript asynchrone ») et désignant une solution informatique libre pour le développement d'applications Web.

6.1.2 Mais encore .

AJAX n 'est ni un langage de programmation, ni une nouvelle technique mais bien un terme « accrocheur » pour désigner l'utilisation combinée de vieilles technologies Web pour obtenir le transfert de données entre le navigateur et le serveur sans que la page Web soit entièrement rechargée et sans bloquer la navigation dans la page.

En effet, avant l'utilisation de ces techniques, l'utilisateur qui venait d'envoyer une requête vers le serveur (envoi d'un formulaire d'inscription via un bouton « submit » par exemple) ne pouvait pas se déplacer dans sa page Web avant que la page Web, réponse à sa requête soit complètement chargée (via un script PHP par ex.) dans son navigateur

6.1.3 Les fameuses ˝vieilles˝ technologies Web utilisées par AJAX

- XHTML ou HTML pour la structure du contenu de la page Web.
- CSS pour la mise en forme du contenu de la page Web.
- JavaScript pour les transformations dynamiques de la page et l'interactivité avec l'utilisateur.
- Le DOM pour la lecture et /ou l'écriture de données dans une page HTML ET dans un fichier XML
- L'objet XMLHttpRequest comme mécanisme d'échange de données entre le navigateur et le serveur de manière synchrone ou asynchrone.
- XML, JSON ou simplement .txt comme format de fichiers contenant les données échangées.

Cependant, malgré toutes ces énumérations, vous « faites » déjà de l'AJAX à partir du moment où vous utilisez l'objet XMLHttpRequest de manière asynchrone.

6.2 Les formats de données échangées avec le serveur

6.2.1 Texte (paire)

```
name:"Mulkay";firstName:"Alain";address:"Grand Place de Quenast,13
Bte 6";zip:"1430";city:"Rebecq";country:"Belgium"
```

6.2.2 Extrait Code HTML

```
<table class="table" id="resultTable">
        <caption class="caption">Members List</caption>
        <thead
                <tr><th>Name</th><th>FirstName</th<th>Card N°</th></tr>
        </thead>
        <tfoot>
                <tr><td colspan="3" ><a href="#">back to the
top</a></td></tr>
        </tfoot>
        <tbody>
        <tr><td>Mulkay</td><td>Alain</td><td>4923</td></tr>
        <tr><td>Mulkay</td><td>Pierre</td><td>4800</td></tr>
        <tr><td>Durand</td><td>Denis</td>< td>4809</td></tr>
        <tr><td>Dubois</td><td>Julien</td>< td>4802</td></tr>
        </tbody>
</table>
```

6.2.3 XML

```
<people>
        <member>
                <name>Mulkay</name>
                <firstName>Alain<firstName>
                <address>
                        <street>Grand Place de Quenast,13 Bte 6</street>
                        <zip>1430</zip>
                        <city>Rebecq</city>
                        <country>Belgium</country>
                </address>
        </member>
        <member>
                <name>Mulkay</name>
                <firstName>Morgane<firstName>
                <address>
                        <street>Rue Roi Albert, 44</street>
                        <zip>5060</zip>
                        <city>Sambreville</city>
                        <country>Belgium</country>
                </address>
        </member>
</people>
```

6.2.4 JSON

```
[
	{
		name:”Mulkay”,
		firstName:”Alain”,
		address :{
			street:”Grand Place de Quenast,13 Bte 6”,
			zip:”1430”,
			city:”Rebecq” ,
			country:”Belgium”
		}
	} ,
	{
		name:”Mulkay”,
		firstName:”Morgane”,
		address :{
			street:”Rue Roi Albert,44”,
			zip:”5060”,
			city:”Sambreville” ,
			country:”Belgium”
		}
	}
]
```

6.3 Les méthodes AJAX

Méthode(E = $('sél. ')	Description
url	adresse du fichier (php, asp, etc..) vers lequel est envoyé la requête
data	Données envoyées vers le serveur
fct	Fonction Callback exécutée si la requête a réussie
dataType	le type de données attendu en retour de la requête (html, xml, json, text, script)
$.get(url[,data][,fct][,dataType])	récupère des données du serveur après envoi optionnel de données (data) de format 'dataType' au serveur via une requête Get
$.post(url[,data][,fct][,dataType])	récupère des données du serveur après envoi optionnel de données (data) de format 'dataType' au serveur via une requête Post
$.load(url[,data][,fct])	récupère des données du serveur au format html après envoi optionnel de données (data) au serveur
$.getJSON(url[,data][,fct])	récupère des données du serveur au format json après envoi optionnel de données (data) au serveur
$.getScript(url[,fct])	récupère du serveur un script Javascript
$.ajax(settings)	Méthode de bas niveau permettant de paramétrer l'objet AJAX dans les moindres détail via (settings) voir exemple
$.ajaxSetup(settings)	Définit des paramètres par défaut pour la méthode de bas niveau
.ajaxComplete(fct)	la fct s'exécutera lorsque la requête Ajax sera complètement terminée
.ajaxError(fct)	la fct s'exécutera lorsque la requête Ajax sera complètement terminée et s'il y a eu une erreur lors de son exécution
.ajaxSend(fct)	la fct s'exécutera lorsque la requête Ajax sera envoyée au serveur
.ajaxStart(fct)	la fct s'exécutera lorsque la requête Ajax commencera à s'exécuter
.ajaxStop(fct)	la fct s'exécutera lorsque toute les requêtes Ajax seront terminées
.ajaxSuccess(fct)	la fct s'exécutera lorsque la requête Ajax sera complètement terminée avec succès
.serialize()	Encode le contenu d'un formulaire sous forme d'un string au format URL-encoded notation
.serializeArray()	Encode le contenu d'un formulaire sous forme d'un Array de couples nom:valeur

Exemples:

Chargement d'un extrait de code html

```
$('#result').load('../source/content.html')
```

insert le code html du fichier 'content.html' dans l'élément d'id ="result"

Chargement d'un fichier XML (fichier repris au § 6.2. 3)

Récupération du nom, du prénom et du code postal de chaque personnes contenue dans le fichier JSON et injection des résultats dans une table d'id='resultTable' (décrite au § 6.2.2)

```
$.get("address.xml",function (response) {
    $(response).find("member").each(function(){
        $("#resultTable tbody").prepend('<tr> td>' + $(this).children().eq(0).text() + '</td><td>' +
            $(this).children().eq(1).text() + '</td><td>' +$(this).children().eq(3).
            children().eq(2).text()+ '</td></tr>');
        })
});
```

Chargement d'un fichier JSON (fichier repris au § 6.2.4)

Récupération du nom, du prénom et du code postal de chaque personnes contenue dans le fichier JSON et injection des résultats dans une table d'id='resultTable' (décrite au § 6.2.2)

```
$.getJSON('address.json', function(json){
    $.each(json, function(index, content){
        $('#resultTable tbody').prepend('<tr><td>' + content.name + '</td>'+ '<td>' +
          content.firstName + '</td><td>' + content.adress.zip + '</td>'</tr>');
     })
});
```

Méthode AJAX permettant de paramétrer un maximum d'élément

```
$.ajax ({
        url: '../source/content.html',
        async: true,
        beforeSend: function () {   },
        cache: true,
        complete: function () {   },
        contentType: application/x-www-form-urlencoded,          (type MIME)
        data: 'name=MULKAY&firstname=Alain'      (map ou string)
        dataType: 'xml'                  (xml, json, html, text, script)
        error: function () {   },
        password: 'azerty'               (mot de passe pour authentification si la requête l'exige)
        username: 'mulkay.a'             (nom pour authentification si la requête l'exige)
        success: function () {   },
        type: 'GET'
});
```

7 jQuery UI (jQuery avancé)

8 Création d'un Plug-in jQuery (jQuery avancé)

8.1 Introduction

Au même titre que la prestigieuse librairie jQuery-UI ,nous pouvons nous-même créer notre propre plug-in jQuery. Ce n'est pas très compliqué mais si nous voulons le partager avec la communauté jQuery, quelques règles doivent être respectées.

Le plug-in sera composé d'une seule fonction publique contenant tout le code.Cette fonction ser incluse dans une fonction anonyme permettant le fonctionnement du plug-in ,même si la variable $ est ''désactivée''.

La fonction aura un nom représentatif de ce qu'elle permet de faire , ici pour des raisons de facilité nous l'appellerons *pluginJQ*

Le fichier .js contenant l'entièreté du code javascript sera sauvé sous la forme « jQuery.nom_du_plugin.js » => dans notre exemple il se nommera donc jQuery.pluginJP.js

8.2 Création

8.2.1 Fct container

```
(function ($) {

}) (jQuery);
```

8.2.2 Déclaration de la fct publique de notre plug-in

```
(function ($) {
        //definition du plug-in
        $.fn.pluginJQ = function (param) {

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

8.2.3 Notre plug-in doit renvoyer un objet jQuery

```
(function ($) {
        //definition du plug-in
        $.fn.pluginJQ = function (param) {

                //futur code du plug-in

                return $(this) ;
        }) ;
}) (jQuery);
```

8.2.4 Gestion des paramètres par défaut

```
(function ($) {
        //definition du plug-in
        $.fn.pluginJQ = function (param) {
                var defaults = {
```

```
                        param1 : ''valeur1'',
                        param2 : ''aleur2'',
                        ……..
                        paramN : ''valeurN''
                }

                //futur code du plug-in

                return $(this) ;
        }) ;
}) (jQuery);
```

8.2.5 Gestion des paramètres optionnels

```
(function ($) {
        //definition du plug-in
        $.fn.pluginJQ = function (param) {
                var defaults = {
                        param1 : ''valeur1'',
                        param2 : ''aleur2'',
                        ……..
                        paramN : ''valeurN''
                }

                var opts =  $.extend(defauilts,param);

                //futur code du plug-in

                return $(this) ;
        }) ;
}) (jQuery);
```

Ceci est donc la structure type que doit respecter votre plug-in qu'il soit simplissime ou extrêmement complexe.

8.3 Exemple de plug-in

8.3.1 Code du plugin

```
// contenu du fichier : jQuery.pluginJQ.js
 (function ($) {
        $.fn.pluginJQ = function (param) {
                //déclaration des paramètres par défaut
                var defaults = {
                        speed: 2000,
                        size: "100%"
                };
                // fusionne les paramètres par défaut avec les paramètres utilisateurs
                var opts = $.extend(defaults, param);
                //cache la div id="photo"
                $('#photo').css({"display":"none"});
                //groupe les img du site dans un objet jQuery
                var source = $("img");
                var index = 0;
                var lastbool = false
                // Les retirent du site
                $('img').remove();
                 //fonction affichant à tour de rôle les photos dans la div id=photo
                var displayPhoto = function(){
                if (!lastbool) {
                        if (index ==0 ){
                                $('#photo').html('<img alt="' + source.eq(index).attr("alt") + '" src="'
+ source.eq(index).attr("src")+ '" width="' + opts.size + '" />').show(opts.speed); }
                        else {
                                $('#photo').html('<img alt="' + source.eq(index).attr("alt") + '" src="'
+ source.eq(index).attr("src")+ '" width="' + opts.size + '" />') }
                                index++;}
                else {
                        $('#photo').hide(opts.speed);
                        lastbool = false; }
                if (index == source.length) {
                        lastbool = true
                        index = 0
                }
                $("#go").on("click",displayPhoto);
                // renvoit un objet jQuery pour le chainage
                return $(this);
        };
}) (jQuery);
```

8.3.2 Code inclus dans main.js

```
// contenu du fichier : main.js
$(document).ready( function (){
        // utilisé avec les paramètres par défaut
        //$(‘’#photo ‘’).pluginJQ() ;
        //utilisation du plugin avec un des paramètres optionnels
        $('#photo').pluginJQ({
                size: "50%"
        });
})
```

8.3.3 Code minimum de la page html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
        <meta http-equiv="content-type" content="text/html; charset=utf-8" />
        <link rel="stylesheet" href="../css/main.css" type="text/css" media="screen"
title="no title" charset="utf-8">

        <script  type="text/javascript"  src=”
src="hhttps://ajax.googleapis.com/ajax/libs/jQuery/1.7.1/jQuery.min.js"></script>
        <script type="text/javascript" src="../js/jQuery.pluginJQ.js"></script>
        <script type="text/javascript" src="../js/main.js"></script>

        <title>Plug-In</title>
</head>
<body id="selector" >
        <div id="photo">
        </div>
        ……..
        < !--   quelques balises img   -->
</body>
</html>
```

9 Sources et liens

9.1