

JQUERY

LI385 - Nouvelles Technologies du web

jQuery

- Librairie Javascript qui permet de :
 - Simplifier les taches de base en Javascript.
 - Sélectionner simplement des partie d'une page HTML.
 - Modifier l'apparence et le contenu de la page.
 - Créer et modifier des événements.
 - Animer une page.
 - Faire de l'AJAX.
- En évitant les problèmes de compatibilité.

jQuery

- Projet open-source, bien supporté et documenté.
 - <http://jQuery.com/>
- Ne se substitue pas à la connaissance de Javascript
 - jQuery est du javascript
 - Tout ne se fait pas naturellement en jQuery.
 - Eviter de tout "Jqueriser".

Inclure du jQuery

- Télécharger le fichier jQuery.js
 - Il existe une version jQuery-min.js plus légère
- Inclure le fichier
 - `<script src="jQuery.js"></script>`
- La librairie JQuery est prête à être utilisée.

Philosophie jQuery

- Sélectionner une partie du document.
 - ▣ Objet jQuery = ensemble de nœuds du DOM.
 - Avec des infos en plus pour jQuery
 - ▣ Les objets se créent avec la fonction jQuery()
 - Cf la notion d'objet en javascript

```
// récupérer tous les div du DOM
```

```
jQuery("div")
```

```
// récupérer toutes les balises de classe c1
```

```
jQuery(".c1")
```

Philosophie jQuery

- Sélectionner une partie du document.
- Agir dessus
 - ▣ Fonction à appliquer à des objets

```
// Cacher tous les <div> de la page  
jQuery("div").hide();
```

```
// Ajouter la classe c2 à tous les objets de classe c1  
jQuery(".c1").addClass("c2");
```

Besoin d'aide

- Tout est sur <http://jQuery.com/>
 - Le code source
 - La doc complète
 - Des exemples
 - Plein de plugins
 - ...

Dans la suite

- Des exemples pour :
 - Sélectionner des objets.
 - Utiliser les événements.
 - Ajouter des effets d'animations.
 - Faire des requêtes "à la AJAX"
- Conception de plugin jQuery

JQUERY – PREMIERS PAS

LI385 - Nouvelles Technologies du web

Un exemple simple

- Ajout automatique d'une classe c1 à tous les div.

```
<script type="text/javascript">
function addClassC1() {
    var divs = document.getElementsByTagName("div");
    for (var i = 0; i < divs.length; i++) {
        var div = divs[i];
        div.setAttribute("class", "c1");
    }
}
</script>

<body onLoad='addClassC1();'>
```

Un exemple simple

- En JQuery
 - Sélection les divs : `jQuery("div")`
 - Ajout d'une classe : `addClass()`
 - Exécution au chargement via l'événement `onLoad`
- Problèmes :
 - Il faut modifier la partie HTML
 - Tout doit être chargé, images comprises

```
<html>
<head>
<script type="text/javascript" src="jquery.js"></script>
</head>
<body onLoad='jQuery("div").addClass("c1");' >
  <div>classe ajoutée au chargement</div>
</body>
</html>
```

Un exemple simple

- En jQuery : `jQuery(document).ready()`
 - ▣ Prend la fonction à exécuter après la création du DOM

```
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
  jQuery(document).ready(
    function(){jQuery("div").addClass("c1");}
  );
</script>
</head>

<body>
  <div>classe ajoutée au chargement</div>
</body>
```

Un exemple simple

- Exécution quand le DOM est construit
 - ▣ Équivalent du onLoad en javascript
 - ▣ Sauf qu'il est exécuté avant le chargement des images
- Syntaxe courte :
 - ▣ `jQuery(function(){...});`
- On peut utiliser \$ à la place de jQuery
 - ▣ Sauf en cas d'utilisation d'autres librairies
 - ▣ `$(function() {});`

Attention

- Tant que le DOM n'est pas construit, rien n'existe
 - ▣ Ne pas tenter d'accéder aux objets !!!

```
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript">
$("div").hide();
$(function(){jQuery("div").addClass("c1");});
</script>
</head>

<body>
  <div>classe ajoutée au chargement</div>
</body>
```

Retour des fonctions et chainage

```
.eq(2)  
.show()  
.addClass("main")  
.html("Hello jQuery");
```

JQUERY SÉLECTION



Sélecteur

- Principe de base :
 - ▣ Un sélecteur retourne un tableau d'objets jQuery :
 - \$("*") retourne toutes les balises
 - \$("div") retourne un tableau contenant tous les div
 - \$("div").length permet de connaître le nombre de div
 - ▣ On peut également créer des balises :
 - \$("<div />")

Sélection restreinte

- Possibilité de sélectionner de manière similaire aux css :
 - ▣ par type de bloc : \$("balise")
 - ▣ par identifiant (id) : \$("#id")
 - ▣ par classe (class) : \$(".class")

```
// <div>Test</div>
$("div")

// <span id="test">tt</span>
$("#test")

// <ul class="test">tt</ul>
$(".test")
```

Sélection restreinte

- Possibilité de combiner les sélecteurs.

```
// tous les divs de classe main
$("div.main")

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

// objets d'id "content" ou de classe "menu"
$("#content, .menu")
```

Sélection dans un contexte

- Sélection dans un contexte :
 - `$(recherche, contexte)`
 - Sélectionne d'abord le *contexte* (une partie du DOM) puis effectue la *recherche* à l'intérieur.

```
// comme en CSS :  
$("div .header").hide();  
  
// ou chercher tous les div  
// chercher des balises de class header à l'intérieur  
$(".header", $("div"))  
  .hide(); // cacher ces balises
```

Sélection dans un contexte

- Dans certaines situations on a besoin du contexte :
 - ▣ On peut attribuer une action sur le contexte
 - ▣ Puis agir sur une partie de celui-ci.

```
// si on clique sur un div
// les paragraphes à l'intérieur vont disparaître ou apparaître
$('div')
    .click(function() {
        $('p', this).toggle();
    });
```

Sélection dans un contexte

- On peut aussi faire des recherches avec `find()`.
- Supprimer le dernier filtrage avec `end()`.

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

// similaire à :
// $(selecteur, contexte)
$(".header", $("p")).show();

// cacher tous les h1 qui contiennent un p
$("h1").find("p").end().hide();
```

Sélecteurs de hiérarchie

- Possibilité d'atteindre :
 - Les fils (>).
 - 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")
// ne cache rien
$("ul > ol")
```

Filtres sur les nom d'attributs

```
// les éléments <div /> ayant un identifiant
$("div[id]")

// les éléments <div id="test" />
$("div[id='test']")

// les éléments <div id="test..." />
$("div[id^='test']")

// les éléments <div id="...test" />
$("div[id$='test']")

// les éléments <a href="...test..." />
$("a[href*='test']")
```

La fonction each

- appelle une fonction pour chaque élément sélectionné
 - ▣ `$(this)` = élément courant
 - ▣ `i` - index de l'élément courant

```
$("#table tr")  
  .each(function(i){  
    if (i % 2)  
      $(this).addClass("odd");  
  });
```

Mais encore

- Des sélecteurs pour :
 - Des parties d'un formulaire
 - Des éléments cochés ou sélectionnés
 - Le premier ou dernier élément
 - Les éléments pairs ou impairs
 - ...

MODIFICATION



Modifier le contenu HTML

```
// modifier le contenu de tous les p du document
$("p").html("<div>Bonjour</div>");

// modifier le contenu de div.a en celui de div.c
$("div.a").html($("div.c").html());
// idem pour div.b avec échappement du contenu de div.c
$("div.b").text($("div.c").html());
```

```
<p></p>
```

```
<div id="a">Bonjour</div>
```

```
<div id="b"><a href="#">Au  
revoir</a></div>
```

```
<div id="c"><a href="#">Au  
revoir</a></div>
```

```
<p><div>Bonjour</div></p>
```

```
<div id="a"><a href="#">Au  
revoir</a></div>
```

```
<div id="b">&lt;a href="#"&gt;Au  
revoir&lt;/a&gt;</div>
```

```
<div id="c"><a href="#">Au  
revoir</a></div>
```

Modifier des valeurs

- `val()` : permet d'obtenir la valeur des objets
- `val(valeur)` permet de modifier la valeur des objets

```
// obtenir la valeur de toutes les checkbox validées
$("input:checkbox:checked").val();

// modifier la valeur d'une textbox de nom txt
$("text[name='txt']").val("Hello");

// select or check lists or checkboxes
$("#lst").val(["NY", "IL", "NS"]);
```

Manipulation de CSS

```
$("#p").removeClass("blue").addClass("red");  
$("#div").toggleClass("main");  
if ($("#div").hasClass("main")) { ... }  
  
$("#div").css("background-color"); //getter  
$("#div").css("float", "left"); //setter
```

Insérer des éléments

```
// Ajouter à la fin de l'objet
$("ul").append("<li>Test</li>");
$("<li/>").html("Test").appendTo("ul");

// Ajouter au début
$("ul").prepend("<li>Test</li>");
$("<li/>").html("Test").prependTo("ul");
```

Remplacer des éléments

```
// remplace tous les h1 par <div>Test</div>
$("h1").replaceWith("<div>Test</div>");

// Remplace tous les h1 par un div
$("<div>Test</div>").replaceAll("h1");

// Sans modifier le contenu :
$("h1").each(function(){
    $(this)
        .replaceWith("<div>"+ $(this).html()+"</div>");
});
```

Supprimer des éléments

```
// supprimer
$("span.names").remove();

// vider tout le contenu
$("#mainContent").empty();

// déplacer
$("p")
    .remove(":not(.red)")
    .appendTo("#main");
```

Gestion des attributs

```
// <a href="home.htm">...</a>
$("a").attr("href", "home.htm");

// met tous les attributs comme celui du premier bouton
$("button:gt(0)")
    .attr("disabled",
        $("button:eq(0)").attr("disabled"));

// supprime l'attribut disabled
$("button").removeAttr("disabled")
```

LES ÉVÉNEMENTS

LI385 - Nouvelles Technologies du web

Le premier événement

- Chargement de la page
 - ▣ `$(document).ready(...)`
 - ▣ Assez similaire à `onLoad()` :
 - `onLoad` : quand tout est chargé
 - `ready` : quand le DOM est prêt (avant chargement des images)

```
$(document).ready(function(){  
    ...  
});
```

Gestionnaire d'événements

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

- Événements et balises
 - ▣ on(event,function) : associe la fonction à l'événement
 - Uniquement pour les objets qui existent à la création
 - ▣ off : supprime l'événement

Événements

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

// exécuter une seule fois (pour chaque objet)
$("div").one("click", function() {$(this).text("test");});

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

Trigger

- Simule un événement utilisateur
 - ▣ Voir aussi triggerHandler

```
<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);  
}
```

L'objet Event - attributs

- type :
 - ▣ nom de l'événement exécuté
- target :
 - ▣ objet qui a exécuté l'événement
 - ▣ cf propagation des événements
- currentTarget :
 - ▣ = this
- pageX et pageY :
 - ▣ position de la souris
- ...

Un exemple : position de la souris

- Exemple avec événement lié au déplacement de la souris.
 - ▣ Via un événement lié à tout le document.

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

<script>
$(document).on("mousemove", function(event) {
    $("#log").text(event.pageX + " " + event.pageY);
});
</script>
```

L'objet Event - méthodes

- preventDefault / isDefaultPrevented :
 - ▣ bloque le fonctionnement par défaut

- stopPropagation / isPropagationStopped / stopImmediatePropagation / isImmediatePropagationStopped

Propagation des événements

- Exemple : menu déroulant multi-niveaux.
- Un clic se propage sur tous les objets associés :
 - ▣ Si on clique sur ` Niveau 3 : item 2` alors on clique aussi sur le `` 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>
```

Propagation des événements

- On peut stopper la propagation des événements :
 - ▣ `stopPropagation();`
 - ▣ Ou de manière similaire faire `return false`

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

Événements dans le futur

- Un événement n'est appliqué qu'aux éléments qui existent :
 - Un div créé après le on ne sera pas cliquable.
 - Utilisation de délégation d'événement
 - Ajouté dans un parent

```
// Un événement par ligne du tableau
$("table tr").on("click", function() {
    console.log("evenement");
});
```

```
// Un événement pour tout le tableau avec délégation
$("table").on("click", "tr", function() {
    console.log("evenement");
});
```

Remarque

- 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>
```

LES EFFETS

LI385 - Nouvelles Technologies du web

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);
```

Translation et fading

```
// Translation
$("div").slideUp();
$("div").slideDown("fast");
$("div").slideToggle(1000);

// Fading
$("div").fadeIn("fast");
$("div").fadeOut("normal");

// Fading avec opacité fixée
$("div").fadeTo ("fast", 0.5);
```

Fin d'un effet

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

Effet personnalisé

- `.animate(options, durée, transition, complete, ...)` :
 - Options : ensemble de propriétés CSS.
 - Transition : comment se déroule l'animation (linéaire ou pas).
 - Complete : callback exécuté après la fin de l'animation.
 - ...

```
// 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: "90%",  
     opacity: 0.5,  
     borderWidth: "5px"},  
    1000);
```

Enchaînement d'animations

- Par défaut les animations sont effectuées l'une à la suite de l'autre.
- Modifiable en utilisant "queue:false".

```
// enchaînement 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);
```

FONCTIONNALITÉS "AJAX"

LI385 - Nouvelles Technologies du web

Charger du contenu

- Pour mettre du contenu dans un objet :
 - ▣ c'est "un peu plus simple" qu'en javascript.
 - ▣ On peut ne récupérer qu'une partie du fichier
 - Attention : tout le fichier est récupéré puis parsé.

```
// version sans arguments :
$("div").load("fichier.html");

// version avec arguments :
$("div#content").load(
    "fichier.php",
    {"nom":"guillaume"});

// version ne récupérant que l'objet d'id monid
$("div").load('test.html #monid');
```

Charger du contenu

□ Avec GET / POST

```
$.get(  
  "fichier.html",  
  {id:1},  
  function(data){  
    alert(data);  
  });
```

```
$.post(  
  "fichier.html",  
  {id:1},  
  function(data){  
    alert(data);  
  });
```

Les fonctions AJAX

- `.ajaxComplete()`
 - ▣ Fonction à appeler à la fin de la requête.
- `.ajaxError()`
 - ▣ Fonction à appeler en cas de fin sur erreur
- `.ajaxStart()`
 - ▣ Fonction à appeler au lancement de la requête.
- `.ajaxSend()`
 - ▣ Fonction à appeler avant l'envoi.
- `.ajaxStop()`
 - ▣ Fonction à appeler quand toutes les requêtes Ajax sont terminées.
- `.ajaxSuccess()`
 - ▣ Fonction à appeler quand la requête termine avec succès.

Les fonctions AJAX

```
$('.log').ajaxStart(function() {$(this).append('Start.')});
$('.log').ajaxSend(function() {$(this).append('Send.')});
$('.log').ajaxComplete(function() {$(this).append('Complete.')});
$('.log').ajaxStop(function() {$(this).append('Stop.')});
$('.log').ajaxSuccess(function() {$(this).append('Success.')});

$('.trigger').click(function() {
    $('.result').load('fichier.json');
});
```

```
<div class="trigger">
    Trigger
</div>
<div class="result"></div>
<div class="log"></div>
```

```
<div class="trigger">
    Trigger
</div>
<div class="result">[0]</div>
<div class="log">
    Start.Send.Success.Complete.Stop.
</div>
```

Récupérer du JSON - Javascript

```
$.getJSON(  
    "fichier.json",  
    {id:1},  
    function(users) {  
        alert(users[0].name);  
    });
```

```
$.getScript(  
    "script.js",  
    function() {  
        ...;  
    });
```

Requête Ajax inter domaines

- Appel vers une URL en définissant un callback
 - ▣ Fonction appelée au retour
- Le serveur doit retourner une chaîne à évaluer
 - ▣ maFonction({monCodeJSON})
 - ▣ Tout est géré par jQuery
 - ▣ Le code qui va être exécuté sur le client est envoyé par le serveur !

```
// coté client
$(function() {
  $.getJSON(
    "http://jlguillaume.free.fr/t.php?callback=?",
    function(data) {alert(data.nom);});
});

// coté serveur
$jsonData = '{"nom" : "G", "prenom" : "JL"}';
echo $_GET['callback'] . '(' . $jsonData . ');';
```

De manière générale

- Il existe la fonction ajax générique :
 - ▣ Toutes les autres fonctions sont un cas particulier de celle-ci.

```
$.ajax({
  async: false,
  type: "POST",
  url: "test.html",
  data: "nom=JL",
  success: function(msg){
    alert( "Data Saved: " + msg );}
});
```

JQUERY CRÉATION DE PLUGIN



Un plugin, c'est quoi

- Une fonction qu'on voudrait ajouter dans jQuery
 - ▣ jQuery est un objet javascript classique
 - ▣ On peut le modifier via son prototype
 - Ajout d'un slot de code
 - En jQuery, on utilise fn au lieu de prototype

```
// jQuery, window.jQuery, $, window.$ sont synonymes
var jQuery = function( selector, context ) {
    ...
    return (window.jQuery = window.$ = jQuery);
})();

// fn et prototype sont synonymes
jQuery.fn = jQuery.prototype = { ... }
```

Premier plugin

- Création d'un plugin affichant le contenu des objets

```
jQuery.fn.monAlerte = function() {  
    alert(this.html());  
};  
  
$(function() {  
    $("div").monAlerte();  
});
```

Plugin chainable

- Assurer que le plugin puisse être chaîné
 - ▣ Il faut donc retourner l'objet courant
 - ▣ Toute fonction doit retourner un objet jQuery
 - Sauf si le plugin n'a pas vocation à être chaîné

```
jQuery.fn.monAlerte = function() {  
    alert(this.html());  
    return this;  
};  
  
$(function() {  
    $("div").monAlerte().hide();  
});
```

Tableau d'objets

- Un objet peut correspondre à plusieurs éléments
 - ▣ Il faut agir sur chacun d'eux
 - ▣ Utilisation de la fonction each
 - Cette fonction retourne déjà tout les objets

```
jQuery.fn.monAlerte = function() {  
    return this.each(function(){  
        alert($(this).html());});  
};  
  
$(function() {  
    $("div").monAlerte().hide();  
});
```

Clôture

- Mieux vaut éviter le signe \$ dans le code
 - ▣ Risque de conflits avec d'autres librairies
- Ou clôture
 - ▣ Ici \$ n'est qu'une variable locale à la définition

```
(function($) {  
  $.fn.monAlerte = function() {  
    return this.each(function(){alert($(this).html());});  
  };  
})(jQuery);  
  
$(function() {  
  $("div").monAlerte().hide();  
});
```

Version finale

- Possibilité de chaînage : return this
- Exécution sur plusieurs objets : each
- Clôture
- fn au lieu de prototype

```
(function($) {  
    $.fn.monAlerte = function() {  
        return this.each(function(){  
            alert($(this).html());  
        });  
    };  
})(jQuery);
```

Ajouter des options

- Un plugin est généralement configurable
 - ▣ Ajout d'options au plugin
 - ▣ Appel du plugin avec `.monPlugin(options)`
 - ▣ Obligation de passer tous les paramètres à chaque fois !

```
(function($) {  
    $.fn.monAlerte = function(debut, fin) {  
        return this.each(function(){  
            alert(debut + $(this).html() + fin);  
        });  
    };  
})(jQuery);  
  
$(function() {$("#div").monAlerte(">>> ", " <<<").hide();});
```

Fonction extend

- Fusionne plusieurs objets (récursivement si besoin)

```
var settings = {validate: false, limit: 5, name: "foo"};
var options = {validate: true, name: "bar"};
jQuery.extend(settings, options);
// settings : {validate: true, limit: 5, name: "bar"}

var empty = {}
var defaults = {validate: false, limit: 5, name: "foo"};
var options = {validate: true, name: "bar"};
var settings = $.extend(empty, defaults, options);
// empty == {validate: true, limit: 5, name: "bar"}
```

Ajouter des options

```
(function($) {  
  $.fn.monAlerte = function(options) {  
    var defaults = {'debut': '>>> ', 'fin': ' <<<'};  
  
    return this.each(function(){  
      if (options) {  
        $.extend(defaults, options );  
      }  
      alert(defaults.debut + $(this).html() + defaults.fin);  
    });  
  };  
})(jQuery);  
  
$(function() {  
  $("div").monAlerte({'fin': '---'}).hide();  
});
```

Remarques finales

- `$.fn.monAlerte = function() {...}`
 - ▣ Crée une "méthode" `monAlerte` dans l'objet `jQuery`
 - ▣ Risque de conflit si
 - La méthode existe déjà
 - Quelqu'un d'autre crée une méthode `monAlerte`
- `jQuery` recommande :
 - ▣ Choix de nom intelligent (éviter `hide` par exemple)
 - ▣ Pas plus d'une fonction par plugin
 - Risque d'encombrement du prototype et de conflit entre plugins

Plugin final

```
(function($) {  
  var settings = {'debut':'>>> ', 'fin':' <<<'};  
  var methods = {  
    init : function( options ) { $.extend( settings, options ); return this;},  
    show : function( ) {  
      return this.each(function(){  
        alert(settings.debut + $(this).html() + settings.fin);  
      })  
    }  
  };  
  $.fn.monAlerte = function( method ) {  
    // si la méthode existe, on l'exécute on lui passant les arguments  
    if (methods[method]) {  
      return methods[method].apply(this, Array.prototype.slice.call(arguments, 1));  
    }  
    // si pas d'argument ou un argument de type objet alors on appelle init par défaut  
    else if (typeof method === 'object' || ! method) {  
      return methods.init.apply( this, arguments );  
    }  
    // sinon on lève une exception  
    else {  
      $.error('Method ' + method + ' n'existe pas dans jQuery.monAlerte' );  
    }  
  };  
})(jQuery);  
$(function() {$("#div").monAlerte('init',{'fin':'---'}).monAlerte('show');});
```

CONCLUSION



iQuery

- Possibilité de récupérer des objets du DOM.
- Événements et animations.
- AJAX.

- Tout ça de manière indépendante des navigateurs.

Mais encore

- En plus des fonctions, on a accès à de nombreuses fonctions d'interface utilisateur (jQuery-UI) :
 - Drag and drop
 - Tableaux triables
 - Accordéons
 - Éléments triables
 - Barres de progression
 - Onglets
 - ...
- Plugins
 - plus de 4000 plugins actuellement
- Attention : HTML5 offre de nouvelles fonctionnalités présentes dans jQuery.

Sortable – éléments triables

```
<head>
<script type="text/javascript" src="jquery.js"></script>
<script type="text/javascript" src="jquery-ui.js"></script>
<script type="text/javascript">
  $(function() {
    $( "#sortable" ).sortable();
  });
</script>
</head>
<body>
<ul id="sortable">
  <li>Item 1</li>
  <li>Item 2</li>
  <li>Item 3</li>
  <li>Item 4</li>
</ul>
</body>
```

Draggable / droppable

```
<script type="text/javascript">
  $(function() {
    $( "#draggable" ).draggable();
    $( "#droppable" ).droppable({drop: function( event, ui
) {
      $( this ).html( "laché!" );}}});
  });
</script>
</head>

<body>
<div id="draggable" >move</div>
<div id="droppable" style="border:1px solid red">on peut
dropper ici</div>
```