jQuery

Inroduction

Les bases de jQuery

Les bibliothèques de jQuery

- **Ajax**
- Attributs
- Callbacks
- Core
- CSS
- Data
- Deffered Objects
- Dimensions
- Events
- Forms
- Internals
- Manipulation
- Miscelaneous
- Offset
- Properties
- Selectors
- Traversing
- Utilities

noConflict()

1.

```
jQuery.noConflict();
// Exécuter jQuery
jQuery("div p").hide();
// Une autre bibliothèque $()
$("content").style.display = 'none';
```

2.

```
jQuery.noConflict();
(function($) {
  $(function() {
    // $ peut être utilisé comme alias à jQuery
  });
})(jQuery);
// $ est l'alias pour une autre bibliothèque
```

Si l'on utilise plusieurs bibliothèques (Mootools, Prototype, etc.) noConlict permet de désactiver l'alias $ pour jQuery.

Il est possible de rétablir cet alias dans un contexte « protégé » au sin d'une fonction anonyme.

Manipuler le DOM

html.tpl.php

- jQuery possède tout une série de fonctions pour manipuler le DOM :
  - changer les attributs d'un élément
  - modifier les styles CSS
  - modifier l'arbre du DOM

Les attributs

- .attr()
- .prop()
- removeAttr()
- removeProp()
- html()
- val()

Remarque: **attr** et **prop** peuvent différer légèrement. **prop** retourne un booléen, alors que **attr** retourne la valeur de l'attribut.

Se méfier aussi que ces fonctions modifient et se basent sur le DOM (pas sur le code HTML)

Les styles CSS

- css()
- hasClass()
- addClass()
- toggleClass()
- removeClass()

Fonction permettant de changer les classes CSS d'un ensemble d'éléments

La structure du DOM

- clone()
- wrap(), unwrap()
- append(), prepend(), html(), text()
- after(), before(), insertAfter(), insertBefore()

jQuery permet de modifier l'arbre du DOM en
- dupliquant un arbre,
- ajoutant à l'intérieur ou autour de la sélection
- remplaçant un arbre par une autre
- en supprimant un arbre (ou un ensemble)

Dimensions

Calcul des dimensions

- height(), innerheight(), outerHeight()
- width(), innerwidth(), outerWidth()

Trois modes de calcul des dimensions (basés sur les valeur calculées) :

1) inner = contenu + padding

2) <> = contenu

3) outer = contenu + padding + border + margin

Parcourir le DOM

Filtrer

- .eq()
- .filter(), slice()
- .first(), .last()
- .has(), .is(), .not()
- .map()

Créer un filtre

jQuery offre un grand nombre de filtres pour les tests logiques de base, les tests sur les enfants et les tests sur le contenu. Par exemple, voici comment filtrer les éléments impairs d'une liste (ici l'équivalent d'une pseudo-classe CSS3 :nth-child(odd) :

```
$("tr:odd").css("background-color", "#bbbbff")
```

On peut créer ses propres filtres en étendant jQuery ainsi :

```
jQuery.extend (jQuery.expr[':'], {
   containsExactly: "$(a).text() == m[3]"
});
```

```
$('element:containsExactly(text)')
```

où a est l'élément à filtrer, m[3] pour la pour passée en argument du filtre, i l'index de l'élément testé et r le tableau complet des éléments à filtrer

Exemple

Dans cet exemple, on mettra le fonc des cellules ayant une valeur positive en vert, et celui des cellules ayant une valeur négative en rouge.

```
jQuery.extend(jQuery.expr[":"], {
  positive : "parseFloat($(a).text()) > 0",
  negative : "parseFloat($(a).text()) < 0",
 });

$('td:negative').css('color', 'red');
$('td:positive').css('color', 'green');
```

Exercices filter & map

Parcourir

- .find()
- .children()
- .parent(), parents(), parentsUntil()
- .next(), .prev()
- .siblings()

Trouver dans le DOM les éléments relatifs à l'élément-cible

Sélection

- .add()
- .addBack()
- contents()
- end()

Fonction permettant la manipulation de la liste des éléments sélectionnés

Evènements

Event Helpers

Méthodes, reprenant le nom des événements DOM. Elles prennent comme unique paramètre une fonction, et leur nom correspond à celui de l'événement DOM, allégé de son préfixe « on » (onclick, onmouseout, etc.).

Si aucun paramètre n'est passé, cette méthode déclenche l'événement.

Si une fonction est passée en paramètre, elle s'exécutera lorsque l'événement sera déclenché (depuis le code comme vu juste au-dessus, ou « naturellement »).

```
$("a#test1").click(function(e){
  e.preventDefault(); // Empêche le navigateur de suivre le lien.
  window.alert("Clic sur a#test1.");
});
```

La fonction passée en argument du helper accepte l'événement lui-même `e` comme paramètre.

bind(), trigger()

La fonction bind est une alternative aux helpers, qu permet d'associer une série d'événement à une fonction :

```
$("a#test3").bind("mouseover mouseout", function(e){
 $(this).text( "Done !" );
});
```

Le déclenchement des événements peut être commandé dans le code par le biais de la fonction trigger :

```
$("a#test4").click function(){
 $("a#test3").trigger( mouseover  );
});
```

L'utilisation de trigger permet de plus de passer des argument au callback associé à l'évènement (le premier argument étant toujours l'évènement lui-même, bien entendu).

unbind()

Il est possible de délier un évènement grâce à unbind() :

```
$("a#test3").unbind("mouseover");
```

Créer un événement

Il est tout à fait possible de baptiser ses propres événements en dehors de la nomenclature jQuery.

Dans ce cas, Ils seront déclenchés par la fonction trigger comme les autres.

```
$("a#test6").bind("StyleFunky", function(e, color){
   var jElt = $(this);
   jElt.css(“color”, color);
});

$("a#test6").trigger(“StyleFunky, “#E4579A”);
```

Evénements nommés

Si vous voulez associer plusieurs actions à un événement, il est possible de les distinguer par les labels et de les exécuter séparément ou globalement. Cela permet aussi de délier séparément les actions.

```
$(element).bind("click.action1", function(){ /* Action 1 */ });
$(element).bind("click.action2", function(){ /* Action 2 */ });
$(element).bind("click.action3", function(){ /* Action 3 */ });

$(element).trigger("click.action3");
```

triggerHandler(), toggle()

triggerHandler permet de déclencher les événements sans associer le comportement par défaut du navigateur (la sélection d'un champ pour focus, la requête HTTP pour click, etc.)

toggle permet d'alterner les réponses à un clic (si vous cliquez deux fois sur un bouton par exemple.

jQuery vs. JS