jQuery :

Table des matières

[Rappels CSS 4](#_Toc467076495)

[Quelle est la nature de l'objet retourné ? 4](#_Toc467076496)

[Plus loin dans la sélection d’éléments : 5](#_Toc467076497)

[Sélecteurs d’attributs 5](#_Toc467076498)

[Sélecteurs hiérarchiques 5](#_Toc467076499)

[Pseudo-sélecteurs d’éléments sélectionnés 6](#_Toc467076500)

[Sélecteurs d’éléments particuliers 6](#_Toc467076501)

[Pseudo-sélecteurs d’éléments liés aux formulaires : 6](#_Toc467076502)

[Pseudo-sélecteurs d’éléments liés aux tableaux 7](#_Toc467076503)

[Modifier contenu d’un élément 8](#_Toc467076504)

[Getter et Setter : 8](#_Toc467076505)

[Modifier les attributs : 8](#_Toc467076506)

[Lire ou ajouter : 8](#_Toc467076507)

[Supprimer : 9](#_Toc467076508)

[Travailler avec l’attribut class : 9](#_Toc467076509)

[Test de l’existence d’une classe : 9](#_Toc467076510)

[Travailler avec les formulaires : 10](#_Toc467076511)

[Travailler avec des valeurs stockées : 10](#_Toc467076512)

[Position et taille des éléments : 11](#_Toc467076513)

[Modifier la position : 11](#_Toc467076514)

[Modifier la taille : 12](#_Toc467076515)

[Connaitre les dimensions : 14](#_Toc467076516)

[Associer des données aux balises : 14](#_Toc467076517)

[Insérer et remplacer des éléments du DOM 15](#_Toc467076518)

[Insérer du contenu 15](#_Toc467076519)

[Remplacer du contenu : 15](#_Toc467076520)

[Insérer des éléments : 15](#_Toc467076521)

[Déplacer du contenu : 16](#_Toc467076522)

[Dupliquer des éléments : 16](#_Toc467076523)

[Entourer des éléments : 17](#_Toc467076524)

[Supprimer des éléments : 17](#_Toc467076525)

[La gestion évènementielle 18](#_Toc467076526)

[La souris : 18](#_Toc467076527)

[Which et type : 18](#_Toc467076528)

[Le clavier : 19](#_Toc467076529)

[Les éléments : 20](#_Toc467076530)

[Focus() et blur() : 21](#_Toc467076531)

[Changements de l’écran : 21](#_Toc467076532)

[Changement du contenu d’un élément : 22](#_Toc467076533)

[Les pages : 22](#_Toc467076534)

[Evènement personnalisés : 23](#_Toc467076535)

[Gérer plusieurs évènements : 23](#_Toc467076536)

[Désactiver un évènement : 24](#_Toc467076537)

[Etiqueter plusieurs évènements : 24](#_Toc467076538)

[Gestion évènementielle unique : 25](#_Toc467076539)

[Déclenchement d’évènement automatique : 25](#_Toc467076540)

[Créer des évènements personnalisés : 26](#_Toc467076541)

[Délégation d’évènements : 27](#_Toc467076542)

[Animations et effets 28](#_Toc467076543)

[Apparition et disparition : 28](#_Toc467076544)

[Apparition et disparition en cascade : 29](#_Toc467076545)

[Fondus enchaînés : 29](#_Toc467076546)

[Aller plus loin : 30](#_Toc467076547)

[Déplier /replier des éléments : 30](#_Toc467076548)

[Désactiver des éléments : 31](#_Toc467076549)

[Définir un délai avant animation : 31](#_Toc467076550)

[Animations personnalisées : 31](#_Toc467076551)

[Première syntaxe : 31](#_Toc467076552)

# Rappels CSS

| **Sélecteur CSS** | **Sélecteur jQuery** | **Signification** |
| --- | --- | --- |
| ul | $('ul') | Les balises <ul> |
| ul.bleu | $('ul.bleu') | La balise <ul> de classe bleu |
| div ul | $('div ul') | La balise <ul> contenue dans la balise <div> |
| div ul li[class="pair"] | $('div ul li[class="pair"]') | La balise <li> contenue dans une balise <ul>, elle-même contenue dans une balise <div>, et dont l'attribut class vaut pair |
| li[class] | $('li[class]') | Les balises <li> qui possèdent un attribut class |
| li[class="impair"] | $('li[class="impair"]') | Les balises <li> qui possèdent un attribut classde valeur impair |
| \* | $('\*') | Toutes les balises du document |

### Quelle est la nature de l'objet retourné ?

Le résultat retourné par la fonction $() est un objet jQuery. Cet objet ressemble à un tableau : il a une propriété length et les éléments sélectionnés peuvent être accédés par un indice. Par exemple :

* $('a').length retourne le nombre de liens hypertextes contenus dans la page.
* $('ul.bleu').length retourne le nombre de balises <ul> de classe bleu.
* $('li[class="impair"]').length retourne le nombre de balises <li> qui ont un attribut class de valeur impair.
* $('body').length retourne « 1 » car le document contient une seule balise <body>.

Pour accéder à un des éléments sélectionnés, précisez son indice entre crochets à la suite du sélecteur. Par exemple :

* $('a')[0] retourne le premier lien hypertexte de la page.
* $('ul.bleu')[3] retourne la quatrième balise <ul> de classe bleu.
* $('body')[0] est équivalent à document.body.

# Plus loin dans la sélection d’éléments :

### Sélecteurs d’attributs

| **Sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| ['nom\*="valeur"'] | Éléments qui possèdent un attribut nom qui contient (partiellement ou totalement) la valeur spécifiée. |
| ['nom~="valeur"'] | Éléments qui possèdent un attribut nom qui contient la valeur spécifiée, délimité par des espaces. |
| ['nom$="valeur"'] | Éléments qui possèdent un attribut nom qui se termine par la valeur spécifiée. |
| ['nom!="valeur"'] | Éléments qui ne possèdent pas l'attribut nom, ou qui possèdent un attribut nomdifférent de la valeur spécifiée. |
| ['nom^="valeur"'] | Éléments qui possèdent un attribut nom qui commence par la valeur spécifiée. |

### Sélecteurs hiérarchiques

| **Sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| ('p > e') | Éléments e directement descendants d'éléments p |
| ('p + e') | Éléments e directement précédés d'un élément p |
| ('p ~ e') | Éléments e précédés d'un élément p |
| :empty | Éléments qui n'ont pas d'enfant |
| :first-child | Premier enfant |
| :first | Premier élément |
| :last-child | Dernier enfant |
| :last | Le dernier élément de la sélection |
| :nth-child() | Élément qui est l'énième enfant de son parent |
| :only-child | Éléments qui sont enfants uniques de leur parent |

### Pseudo-sélecteurs d’éléments sélectionnés

| **Sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| :even | Éléments pairs |
| :odd | Éléments impairs |
| :eq() | Élément dont l'index est spécifié |
| :gt() | Éléments dont l'index est supérieur à (*greater than*) l'index spécifié |
| :lt() | Éléments dont l'index est inférieur à (*lower than*) l'index spécifié |

### Sélecteurs d’éléments particuliers

| **Sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| :header | Tous les titres <h1> à <h6> |
| :hidden | Éléments cachés |
| :visible | Éléments visibles |
| :not() | Éléments qui ne correspondent pas au sélecteur spécifié |

### Pseudo-sélecteurs d’éléments liés aux formulaires :

| **Pseudo-sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| :input | Tous les éléments de type input, textarea, select et button |
| :button | Éléments de type button |
| :checkbox | Éléments de type checkbox |
| :checked | Éléments qui sont cochés |
| :radio | Éléments de type radio |
| :reset | Éléments de type reset |
| :image | Tous les boutons de type image |
| :submit | Éléments de type submit |
| :text | Éléments de type text |
| :password | Éléments de type password |
| :selected | Éléments sélectionnés |
| :focus | Sélectionne l'élément s'il a le focus |
| :enabled | Éléments validés |

### Pseudo-sélecteurs d’éléments liés aux tableaux

| **Pseudo-sélecteur** | **Éléments sélectionnés** |
| --- | --- |
| :first | Premier élément |
| :last | Dernier élément |
| :eq() | Élément dont l'index est spécifié |
| :gt() | Éléments dont l'index est supérieur à l'index spécifié |
| :lt() | Éléments dont l'index est inférieur à l'index spécifié |
| :even | Éléments d'index pair |
| :odd | Éléments d'index impair |
| :empty | Éléments qui n'ont pas d'enfant |
| :not() | Éléments qui ne correspondent pas au sélecteur spécifié |

# Modifier contenu d’un élément

## Getter et Setter :

$('h2').css('font-size'); Getter

$('h2').css('font-size','2em'); Setter

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Sélecteurs CSS</title>

</head>

<body>

<div id="listes">

<a href="http://api.jquery.com">API jQuery</a><br>

<a href="http://docs.jquery.com">Documentation jQuery</a><br>

</div>

<script src="jquery.js"></script>

<script>

$(function() {

var test = $('a').attr('href');

document.write(test);

});

</script>

</body>

</html>

## Modifier les attributs :

## Lire ou ajouter :

* $('#plus').attr('src'); retourne l'attribut src de l'élément d'identifiant plus.
* $('div').attr('class'); retourne l'attribut class du premier <div>.
* $('div').attr('class', 'madiv'); modifie ou crée l'attribut class dans les balises <div> du document et leur affecte la valeur « madiv ».
* $('#illustration').attr('src','monimage.jpg'); modifie ou crée l'attribut src dans la balise d'identifiant illustration et lui affecte la valeur « monimage.jpg ».

## Supprimer :

$(sel).removeAttr('attribut');

On peut aussi accéder aux attributs css :

Getter :

var taille = $('.para').css('font-size');

Setter :

$('.para').css('font-size', '40px');

## Travailler avec l’attribut class :

* addClass() ajoute une classe dans les éléments sélectionnés ;
* removeClass() supprime (si elle existe) une classe des éléments sélectionnés ;
* toggleClass() accomplit deux actions : si la classe spécifiée n'existe pas dans les éléments sélectionnés, elle y est ajoutée. Si elle existe, elle est supprimée.

Attention aux classes qui se chevauchent ! On ne peut pas définir une classe en rouge si elle est déjà écrite en verte :

$('#eric').removeClass('vert').addClass('rouge');

$('#paul').removeClass('vert').addClass('rouge').removeClass('grand').addClass('petit');

Peut aussi être simplifié en :

$('#pierre').removeClass('vert grand').addClass('rouge petit');

## Test de l’existence d’une classe :

L'instruction $('#jean').hasClass('rouge'); renverra la valeur true, car le <span> est de classe rouge.

L'instruction $('#jean').hasClass('petit'); renverra la valeur false, car le <span> n'est pas de classe petit.

On peut donc utiliser des ‘if’ :

if ($('#jean').hasClass('rouge'))

Pour tester l’appartenance à plusieurs classes on utilise : is(). Renvoie true s’il possède bien toutes les classes sélectionnées !

$('#jean').is('.grand.rouge');

## Travailler avec les formulaires :

Vous utiliserez la méthode val() pour tester/modifier la valeur des zones de texte, boutons radio, cases à cocher, listes déroulantes et zones de liste contenues dans un document HTML.

| **Instruction jQuery** | **Effet** |
| --- | --- |
| $('#nom').val() | Lit le nom de l'utilisateur. |
| $('#pass').val() | Lit le mot de passe. |
| $(':radio#H:checked').val() | Lit l'état du bouton radio H. Renvoie true si le bouton est sélectionné, sinon false. |
| $('#fonction').val() | Lit l'élément sélectionné dans la liste déroulante. |
| $('#nom').val('Michel') | Écrit « Michel » dans la zone de texte Nom d'utilisateur. |
| $('#pass').val('abcde') | Écrit « abcde » dans la zone de texte Mot de passe. |
| $(':radio').val(['H']); | Sélectionne le bouton radio H. |
| $('#fonction').val('retraite') | Sélectionne Retraité dans la liste déroulante. |

## Travailler avec des valeurs stockées :

* text() retourne/modifie la valeur textuelle stockée dans l'élément ;
* html() retourne/modifie le code HTML stocké dans l'élément.

$('p').text(function(index,actuel) {

alert('Paragraphe ' + (index+1) + ' : '+actuel);

});

| **Instructions** | **Résultat** |
| --- | --- |
| alert($('p').html()); | Affiche le code HTML du premier élément (voir image suivante). |
| alert($('p:first').html()); | Affiche le code HTML du premier élément. |
| alert($('p:last').html()); | Affiche le code HTML du dernier élément. |
| $('p').html(function(index,actuel) { alert('Paragraphe ' + (index+1) + ' : '+actuel);}); | Affiche individuellement le code HTML de chaque élément. |

Comparé à text(), html() se contente du premier élément et ne les balaie pas tous comme text(). Donc les deux premiers résultats sont identiques.

On peut les utiliser en setter :

$('p:first').html('<a href="http://www.google.com">Moteur de recherche Google</a>');

## Position et taille des éléments :

### Modifier la position :

Pour connaitre et modifier la position d’un élément :

* offset() : position absolue d'un élément dans la page (getter et setter) ;
* position() : position relative d'un élément dans son parent (getter seulement).

Les positions retournées ont deux composantes : abscisse left et ordonnée top, on fera donc :

* offset().left et offset().top pour connaître la position absolue d'un élément.
* position().left et position().top pour connaître la position d'un élément dans son parent.

Prenons le cas de deux div imbriqués, avec un parent et un enfant. On a les classes css suivantes :

#parent {

width: 300px;

height:300px;

position: absolute;

top: 100px;

left: 200px;

background-color: yellow;

}

#enfant {

width: 100px;

height:100px;

position: absolute;

top: 150px;

left: 100px;

background-color: red;

}

Pour #enfant, top et left sont bien en absolu mais relative par rapport au parent !

Pour connaitre les positions on fera donc :

var posparent=$('#parent').offset();

var posenfant=$('#enfant').offset();

$('span').text('Parent : x=' + posparent.left + ', y=' + posparent.top + ' Enfant : x=' + posenfant.left + ', y=' + posenfant.top);

Si on remplace offset() par position () :

Les coordonnées renvoyées sont relatives au parent de chaque balise. Le parent de la balise #parent est le document. Elles ne devraient donc pas changer. Quant au parent de la balise #enfant, il s'agit de la balise #parent. Ses coordonnées seront donc relatives à cette balise.  
Une fois le code modifié et sauvegardé, exécutez le fichier. Vous devriez obtenir l'image suivante.

### Modifier la taille :

var posenfant = $('#enfant').offset();

posenfant.top = 100;

posenfant.left = 100;

$('#enfant').offset(posenfant);

### Connaitre les dimensions :

Plusieurs méthodes jQuery permettent de connaître les dimensions et de redimensionner les éléments de type block :

* width() : largeur de l'élément, de la fenêtre ou du document, sans inclure les marges (padding, border et margin). Cette méthode peut être utilisée comme getter (pour connaître la largeur d'un élément) ou comme setter (pour modifier la largeur d'un élément).
* innerWidth() : largeur de l'élément, en incluant le padding gauche et droit.
* outerWidth() : largeur de l'élément, en incluant le padding gauche et droit et border.
* outerWidth(true) : largeur de l'élément, en incluant padding gauche et droit, border et margingauche et droit.
* height() : hauteur de l'élément, de la fenêtre ou du document, sans inclure les marges (padding, border et margin). Cette méthode peut être utilisée comme getter (pour connaître la hauteur d'un élément) ou comme setter (pour modifier la hauteur d'un élément).
* innerHeight() : hauteur de l'élément, en incluant le padding supérieur et inférieur.
* outerHeight() : hauteur de l'élément, en incluant border et padding supérieur et inférieur.
* outerHeight(true) : hauteur de l'élément, en incluant border, padding supérieur et inférieur et margin supérieur et inférieur.

Pour fixer les dimensions du div, on utilie les setters :

$('div').width('400px');

$('div').height('200px');

### Associer des données aux balises :

Utilisée en setter, la méthode jQuery $.data() permet d'associer une ou plusieurs données textuelles à n'importe quel élément du DOM.

$.data(el, 'nom', nom\_don: don);

… où :

* el est le nom de l'élément concerné, sans apostrophes ;
* nom est le nom (entre apostrophes) de la « variable » dans laquelle sera stockée la donnée ;
* nom\_don est le nom (sans apostrophes) associé à la donnée ;
* don est une donnée quelconque. S'il s'agit d'un nombre, les apostrophes sont inutiles. Par contre, s'il s'agit d'une chaîne, mettez-la entre apostrophes.

Si vous voulez associer plusieurs données à un élément, vous utiliserez une syntaxe légèrement différente :

$.data(el, 'nom', {nom\_don1: don1, nom\_don2: don2, nom\_don3: don3, etc.});

Pour retrouver une donnée :

var uneVariable = $.data(el,'nom').nom\_don;

Pour retirer les données :

var div = $('div')[0];

$.removeData(div, 'mesValeurs');

# Insérer et remplacer des éléments du DOM

## Insérer du contenu

Plusieurs méthodes très pratiques permettent d'insérer du contenu dans ou en dehors de la sélection (entendez par là des éléments retournés par le sélecteur jQuery) :

* append() insère du contenu à la fin de la sélection ;
* prepend() insère du contenu au début de la sélection ;
* before() insère du contenu avant la sélection ;
* after() insère du contenu après la sélection.

## Remplacer du contenu :

$('hr').replaceWith('<br>');

Pour remplacer un titre, on ne remplacerait que <h2>, </h2> va donc disparaitre et il n’y aura aucun rendu.

$('h2').each(function(){

var elemH2 = $(this);

elemH2.replaceWith('<h3>' + elemH2.text() + '</h3>');

});

## Insérer des éléments :

Insérer des éléments dans l’arborescence du DOM :

* eai.appendTo(cible) insère un élément à la fin de la cible ;
* eai.prependTo(cible) insère un élément au début de la cible ;
* eai.insertBefore(cible) insère un élément avant la cible ;
* eai.insertAfter(cible) insère un élément après la cible.

Où eai représente l’élément à insérer.

Ex :

$('<li>Deuxième élément bis</li>').insertAfter($('li:nth-child(2)'));

Ex :

$('<li>Deuxième élément bis</li>').insertBefore($('li:nth-child(2)'));

## Déplacer du contenu :

Pour déplacer un élément existant dans le document, vous utiliserez les méthodes append(), prepend(), before() ou after() :

* $('sel').append(depl);
* $('sel').prepend(depl);
* $('sel').before(depl);
* $('sel').after(depl);

… où sel sélectionne l'élément avant ou après lequel doit se faire le déplacement et depl représente l'élément à déplacer.

## Dupliquer des éléments :

On fait comme pour le déplacement mais on rajoute clone() :

Ex :

$('#deux').clone().insertBefore($('#un'));

Quand on duplique, tous les fils sont dupliqués avec. Ainsi pour faire un sommaire on peut faire :

$('<h1>Sommaire</h1>').insertBefore($('h2:first'));

$('h2').clone().insertAfter($('h1'));

## Entourer des éléments :

La méthode wrap permet d’entourer les éléments.

$('sel').wrap('elwrap');

Autre exemple, si on veut afficher en rouge, gras italique et souligné :

$('li').wrap('<font color="red"><b><i><u></u></i></b></font>');

La méthode wrap() a deux variantes :

* wrapInner(), pour entourer le contenu d'un élément par un autre élément créé à la volée ;
* wrapAll(), pour entourer d'une façon globale les éléments sélectionnés avec un autre élément créé à la volée.

Exemple :

Si on a :

<p>Le texte du paragraphe</p>

Qu’on applique :

$('p').wrapInner('<i></i>');

On aura :

<p><i>Le texte du paragraphe</i></p>

Alors que si on avait juste utilisé wrap normal, on aurait eu :

<i><p>Le texte du paragraphe</p></i>

## Supprimer des éléments :

On utilise la méthode remove().

Ex :

$('h2').remove();

Ou encore : supprimer tous les paragraphes qui contiennent le mot quelconque :

$('p').remove(':contains("quelconque")');

# La gestion évènementielle

## La souris :

Elle se présente sous cette forme :

$(sel).mge(function() {

// Une ou plusieurs instructions jQuery

// pour gérer l'événement lorsqu'il se produit

}

Où sel est un sélecteur et mge une méthode de gestion d’évènement.

| **Méthode** | **Événement géré** |
| --- | --- |
| click() | Clic gauche |
| dblclick() | Double-clic |
| mousedown() | Appui sur le bouton gauche ou droit de la souris alors que le pointeur est au-dessus de l'élément |
| mouseenter() ou mouseover() | Début de survol de l'élément |
| mouseleave() ou mouseout() | Arrêt de survol de l'élément |
| mousemove() | Déplacement du pointeur au-dessus de l'élément |
| mouseup() | Relâchement du bouton gauche ou droit alors que le pointeur est au-dessus de l'élément |
| scroll() | Utilisation de la roulette alors que le pointeur se trouve au-dessus d'un élément concerné par ce type d'événement |

### Which et type :

Il peut être intéressant de savoir quel bouton a été pressé parmi les 3. On utilise event.which

Pour connaitre le type on utilise event.type. La valeur renvoyée pourra être click, dblclick, mousedown, mouseenter, mouseover, mouseleave, mouseout, mousemove ou mouseup.

Exemple :

Cliquez sur l'image avec un des boutons de la souris.<br ></code>

<img id="target" src="petitchat.jpg"><br />

<span id="rapport"></span>

<script src="jquery.js"></script>

<script>

$(function() {

$('#target').mousedown(function(e){

$('#rapport').html('Événement : ' + e.type + '. Bouton pressé : ' + e.which );

});

});

</script>

## Le clavier :

| **Méthode** | **Événement géré** |
| --- | --- |
| keydown() | Appui sur une touche du clavier |
| keyup() | Relâchement d'une touche du clavier préalablement enfoncée |
| keypress() | Maintien d'une touche du clavier enfoncée |

Exemple :

<script src="jquery.js"></script>

<script>

$(function() {

$('#target').keydown(function(){

$('#lumiere').css('background-color', 'green');

});

$('#target').keyup(function(){

$('#lumiere').css('background-color', 'white');

});

});

</script>

Quand on appuie sur une touche, un carré devient vert, quand on relâche il devient blanc.

Pour savoir de quel caractère il s’agit, on utilise event.which, comme pour la souris.

Exemple :

<form>

Laissez aller votre imagination : saisissez quelques mots<br />

<textarea id="saisie"></textarea>

</form><br />

Caractère saisi : <span id="unelettre"></span>

<script src="jquery.js"></script>

<script>

$(function() {

$('#saisie').keypress(function(e) {

$('#unelettre').text(e.which); //keyCode

});

});

</script>

Il retourne le code ASCII et non pas la lettre. Avec keydown() et jeyup() les caractères min et maj sont confondus dans le code ASCII. Ainsi a et A seront tous les deux à valeur 65.

Pour connaitre le vrai caractère et non son code on utilise keypress().

$('#saisie').keypress(function(e) {

var c = String.fromCharCode(e.which); // récupère le code et le converti en caractère

$('#unelettre').text(c);

## Les éléments :

| **Méthode** | **Événement géré** |
| --- | --- |
| focus() | Réception de focus par l'élément |
| blur() | Perte de focus par l'élément |
| focusin() | Réception de focus par l'élément ou un de ses enfants |
| focusout() | Perte de focus par l'élément ou un de ses enfants |
| resize() | Redimensionnement d'un élément |
| change() | Modification d'un élément |

### Focus() et blur() :

<form>

Cliquez sur les zones de texte<p>

<input type="text" class="f" id="Zone-de-texte-1"><p>

<input type="text" class="f" id="Zone-de-texte-2"><br />

</form><br />

Focus : <span id="resultat"></span><br />

Perte de focus : <span id="resultat2"></span>

<script src="jquery.js"></script>

<script>

$(function() {

$('.f').focus(function() {

$('#resultat').text($(this).attr('id'));

});

$('.f').blur(function() {

$('#resultat2').text($(this).attr('id'));

});

});

</script>

## Changements de l’écran :

Resize(), pour détecter quand la taille de la fenêtre change de taille.

Exemple :

<span id="resultat"></span>

<script src="jquery.js"></script>

<script>

$(function() {

$(window).resize(function() {

var taille = 'Taille de la fenêtre : ' + $(window).width() + 'px x ' + $(window).height() + 'px';

$('#resultat').text(taille);

});

});

</script>

## Changement du contenu d’un élément :

Change() détecte les changements de contenu.

Exemple :

<form>

Sélectionnez une valeur dans la liste déroulante

<select>

<option>J'aime jQuery</option>

<option>J'adore jQuery</option>

<option>Je raffole de jQuery</option>

<option>jQuery ? Jamais entendu parler !</option>

</select>

</form><br />

<span id="resultat"></span><br />

<script src="jquery.js"></script>

<script>

$(function() {

$('select').change(function() {

$('#resultat').text('Vous venez de sélectionner "' + $(this).val() +'".');

});

});

</script>

## Les pages :

Appliqué à window, la méthode load() permet de vérifier le chargement d’une page. La méthode unload() permet de détecter les demandes de changement de page par l’utilisateur.

Exemple :

<img src="canard.jpg"><br />

<a href="http://www.siteduzero.com">Cliquez ici pour aller sur le Site du Zéro</a>

<script src="jquery.js"></script>

<script>

$(function() {

alert('Le DOM est chargé');

$(window).load(function() {

alert('La page est entièrement chargée');

});

$(window).unload(function() {

alert('Vous avez demandé à changer de page');

});

});

</script>

Unload() doit obligatoirement être associé à window, la fenêtre du navigateur, alors que load() peut être appliqué )des images ou autre.

## Evènement personnalisés :

La méthode on() :

| **Méthode traditionnelle** | **Méthode on() équivalente** |
| --- | --- |
| $(sel).click(function() {…}) | $(sel).on('click', function() {…}) |
| $(sel).scroll(function() {…}) | $(sel).on('scroll', function() {…}) |
| $(sel).keydown(function() {…}) | $(sel).on('keydown', function(){…}) |
| $(sel).focus(function() {…}) | $(sel).on('focus', function() {…}) |
| $(sel).load(function() {…}) | $(sel).on('load', function() {…}) |

La méthode on() a plusieurs avantages. Elle permet de :

* Limiter l'écriture en associant une même méthode événementielle à plusieurs éléments ;
* Relier plusieurs méthodes événementielles à un élément en une seule instruction ;
* Désactiver une méthode événementielle précédemment attachée avec la méthode on() ;
* Relier plusieurs méthodes événementielles entre elles. Par la suite, vous pourrez toutes les déclencher, ou toutes les désactiver en une seule instruction jQuery.

### Gérer plusieurs évènements :

* Si on veut relier mouseenter et mousemover, on peut faire :

$('img').on('mouseenter mousemove', function() { … })

* Plusieurs méthodes en une seule instruction :

$('img.grand').on({mouseenter:traitement1, mousemove:traitement2});

### Désactiver un évènement :

Exemple : ajout puis suppression d’évènement par bouton

Cliquez sur l'image après avoir activé le clic<br />

<img src="canard.jpg" id='image'><br />

<button id="activer">Activer le clic</button>

<button id="desactiver">Désactiver le clic</button>

<script src="jquery.js"></script>

<script>

$(function() {

function traitement() {

alert('Image cliquée');

}

$('#activer').on('click', function() {

$('#image').click(traitement);

});

$('#desactiver').on('click', function() {

$('#image').off('click', traitement);

});

});

</script>

### Etiqueter plusieurs évènements :

On peut étiqueter un évènement parce que certains plugins utilisent les mêmes commandes que jQuery de base. On les défini comme ça :

$('img').on('mouseenter.depanPC', function() { //traitement });

$('img').on('mouseleave.depanPC', function() { //traitement });

$('img').on('mousemove.depanPC', function() { //traitement });

En rajouant un point puis un nom. On les supprime ainsi du coup :

$('img').off('mouseleave.depanPC');

Ou supprimer les méthodes rattachées à l’espace de nom :

$('img').off('.depanPC');

## Gestion évènementielle unique :

One() permet de déclencher l’évènement qu’une seule fois.

Exemple :

Cliquez sur l'image<br />

<img src="bon.gif"><br />

<span id='message'></span>

<script src="jquery.js"></script>

<script>

$(function() {

$('img').one('click', function() {

$('#message').text('Vous avez cliqué sur l\'image. Désormais, je resterai insensible aux clics.').fadeIn(1000).fadeOut(5000); // apparition et disparition

});

});

</script>

## Déclenchement d’évènement automatique :

Déclenchement sans l’aide de l’utilisateur on utilise un trigger.

$(sel).trigger('ev');

Exemple :

Cliquez sur l'image<br />

<img src="bon.gif"><br />

<span id='message'></span><br />

<button>Cliquez ici</button>

<script src="jquery.js"></script>

<script>

$(function() {

$('img').click(function() {

$('#message').text('L\'image a été cliquée.').fadeIn(1000).fadeOut(1000);

});

$('button').click(function() {

$('img').trigger('click');

});

});

</script>

Seuls ces évènements peuvent être déclenchés : blur, change, click, dblclick, error, focus, keydown, keypress, keyup, select et submit.

On click sur l’image ou sur le bouton. Le bouton simule un click souris. On peut le modifier pour savoir si on a clické sur l’un ou l’autre :

Cliquez sur l'image<br />

<img src="bon.gif"><br />

<span id='message'></span><br />

<button>Cliquez ici</button>

<script src="jquery.js"></script>

<script>

$(function() {

$('img').click(function(event,texte) {

if (texte == undefined)

texte = "par vous";

$('#message').text('L\'image a été cliquée ' + texte).fadeIn(1000).fadeOut(1000);

});

$('button').click(function() {

$('img').trigger('click', 'par jQuery');

});

});

</script>

Si on veut passer plusieurs paramètres au trigger, on doit les mettre entre [ ].

$(sel).trigger('ev', ['param1', 'param2', 'param3', 'etc.']);

## Créer des évènements personnalisés :

<button id="bonjour">Cliquez ici</button>

<script src="jquery.js"></script>

<script>

$(function() {

$('#bonjour').on('bonjour\_jquery', function() {

alert('jQuery vous dit bonjour !');

});

$('#bonjour').click(function() {

$('#bonjour').trigger('bonjour\_jquery');

});

});

</script>

Ici on a créé un évènement et on le simule avec un trigger.

On peut aussi y ajouter des paramètres :

<button id="bonjour">Cliquez ici</button>

<script src="jquery.js"></script>

<script>

$(function() {

$('#bonjour').on('bonjour\_jquery', function(event, param) {

alert(param + ', jQuery vous dit bonjour !');

});

$('#bonjour').click(function() {

$('#bonjour').trigger('bonjour\_jquery', 'Michel');

});

});

</script>

## Délégation d’évènements :

Plutôt que de définir des évènements à chacun, on peut définir un écouteur d’évènement sur un élément parent et écouter sur les enfants :

$('#del').on('ev', 'sel', function () {

//Une ou plusieurs instructions

});

Où :

* del est l'élément sur lequel les enfants seront écoutés ;
* ev est le nom de l'événement concerné ;
* sel est un sélecteur qui indique le type des enfants écoutés ;
* function() est la fonction à exécuter lorsque l'événement ev est détecté.

Lorsque l'événement ev se produit sur un enfant sel de l'élément $('#del') , la fonction est exécutée.

Pour le supprimer, on utilise off().

# Animations et effets

## Apparition et disparition :

Animations sur hide() et show().

Exemple :

<button id="affiche">Faire apparaître les lignes paires</button>

<button id="cache">Faire disparaître les lignes paires</button><br />

<table border>

<tr><td>a</td><td>b</td><td>c</td></tr>

<tr><td>d</td><td>e</td><td>f</td></tr>

<tr><td>g</td><td>h</td><td>i</td></tr>

<tr><td>j</td><td>k</td><td>l</td></tr>

<tr><td>m</td><td>n</td><td>o</td></tr>

</table>

<script src="jquery.js"></script>

<script>

$(function() {

$('tr:even').css('background','yellow');

$('td').css('width','200px');

$('td').css('text-align','center');

$('#affiche').click(function() {

$('tr:even').show('slow');

} );

$('#cache').click(function() {

$('tr:even').hide(1000);

});

});

</script>

On peut redéfinir la vitesse, par exemple :

jQuery.fx.speeds.slow = 1500;

Ou même créer des valeurs :

jQuery.fx.speeds['super-slow'] = 3000;

On peut y ajouter linear ou swing (par défaut) :

show('slow','linear');

hide(1000,'swing');

### Apparition et disparition en cascade :

<button id="affiche">Faire apparaître les images</button>

<button id="cache">Faire disparaître les images</button><br />

<img src="bon.png">

<img src="mauvais.png">

<img src="question.png">

<script src="jquery.js"></script>

<script>

$(function() {

$('#affiche').click(function() {

$('img').first().show('slow', function showNextOne() {

$(this).next('img').show('slow', showNextOne);

});

});

$('#cache').click(function() {

$('img').first().hide('slow', function hideNextOne() {

$(this).next('img').hide('slow', hideNextOne);

});

});

});

</script>

L’apparition ou la disparition de l’un entraîne les autres, comme dans un jeu de domino.

On utilise la première image et on applique la méthode showNextOne().

## Fondus enchaînés :

fadeIn() et fadeOut() sont complémentaires de show() et hide(), elles agissent avec un fondu.

On peut applquer ce qu’on a vu avant :

$('sel').fadeIn(1200,'linear');

$('sel').fadeOut(1000,'swing');

On peut modifier l’opacité :

$('sel').fadeTo(durée, opacité);

* sel est un sélecteur jQuery ;
* durée est la durée de l'animation. Indiquez un entier qui représente une durée en millisecondes ou une chaîne (fast, normal ou slow pour fixer la durée à 200, 400 ou 600 millisecondes) ;
* opacité est un nombre décimal compris entre 0 (transparent) et 1 (opaque).

On peut créer un diaporama avec un fonction récursive :

<style type="text/css">

img { position: absolute; left: 0px; top: 0px; }

#img1 {z-index: 1;}

#img2 {z-index: 2;}

#img3 {z-index: 3;}

#img4 {z-index: 4;}

#img5 {z-index: 5;}

</style>

<img src="zozor5.jpg" id="img5">

<img src="zozor4.jpg" id="img4">

<img src="zozor3.jpg" id="img3">

<img src="zozor2.jpg" id="img2">

<img src="zozor1.jpg" id="img1">

<script src="jquery.js"></script>

<script>

$(function() {

$('img').first().fadeOut(2000, function suivante() {

$(this).next('img').fadeOut(2000,suivante);

});

});

</script>

Plus la propriété z-index est élevée, plus l'image se trouve en avant-plan

## Aller plus loin :

### Déplier /replier des éléments :

* slideDown() augmente la hauteur et la largeur de la sélection jusqu'à atteindre la dimension « native » de chaque élément ;
* slideUp() diminue la hauteur et la largeur de la sélection jusqu'à ce qu'elle disparaisse ;
* slideToggle() inverse l'animation : si les objets sélectionnés ont une taille nulle, leur hauteur et leur largeur sont augmentées jusqu'à ce qu'ils atteignent leur dimension. Par contre, s'ils ont une taille non nulle, leur hauteur et leur largeur sont diminuées jusqu'à ce qu'ils disparaissent.

### Désactiver des éléments :

On peut stopper les animations de son site avec :

jQuery.fx.off = true;

Ou on peut faire en sorte de les activer / désactiver avec un bouton.

$('.stopAnim').click(function() {

jQuery.fx.off = true;

});

$('.execAnim').click(function() {

jQuery.fx.off = false;

});

### Définir un délai avant animation :

On utilise .delay() avec un temps en milliseconde :

<style>

#message { display: none; background-color: yellow; }

</style>

<span id="message">Ce texte sera affiché pendant deux secondes</span><br /><br />

<button id="afficheMessage">Afficher le message</button>

<script src="jquery.js"></script>

<script>

$(function() {

$('#afficheMessage').click(function() {

$('#message').fadeIn('slow').delay(2000).fadeOut('slow');

});

});

</script>

### Animations personnalisées :

On utilise animate() pour créer soi-même une animation :

### Première syntaxe :

La plus souvent utilisée.

$('sel').animate({ prop1: val1, prop2: val2, prop3: val3, etc. }, durée, modèle, function() {

//Une ou plusieurs instructions

});

* sel est un sélecteur jQuery.
* prop1, prop2 et prop3 sont des propriétés CSS et val1, val2 et val3 les valeurs associées. Une valeur numérique sera interprétée comme un nombre de pixels. Une valeur du type « +=50 » demandera d'ajouter progressivement 50 pixels à la propriété. Inversement, la valeur « -=50 » demandera de soustraire progressivement 50 pixels à la propriété. Vous pouvez également utiliser les chaînes show, hide et toggle pour respectivement afficher, dissimuler et inverser la propriété.
* durée est la durée de l'animation (entier en millisecondes ou chaîne fast, normal ou slow).
* modèle est le modèle de progression de l'animation (swing, linear ou un modèle issu d'un plugin).
* function() contient une ou plusieurs instructions qui seront exécutées lorsque l'animation sera terminée.

Dans cette syntaxe, durée, modèle et la fonction sont facultatifs. Si vous ne spécifiez qu'un ou plusieurs couples propriétés/valeurs CSS, l'animation durera 400 millisecondes.

Les données peuvent être :

* Des valeurs littérales : « 0.25 », « 50 », « 'yellow' » ;
* Des valeurs relatives : « +=100 » pour augmenter de 100 pixels ou « -=30 » pour diminuer de 30 pixels ;
* Des pourcentages : « 15% » ;
* Des modèles de progression : swing, linear ou d'autres modèles accessibles via des plugins ;
* Le mot clé toggle pour inverser l'animation.

### Deuxième syntaxe :

$('sel').animate({ prop1: val1, prop2: val2, prop3: val3, etc. }, {options});

… où :

* sel est un sélecteur jQuery ;
* prop1, prop2 et prop3 sont des propriétés CSS et val1, val2 et val3 les valeurs associées ;
* options est un objet qui peut contenir une ou plusieurs des propriétés suivantes :
  + duration : durée de l'animation (entier en millisecondes ou chaîne fast, normal ou slow) ;
  + easing : modèle de progression de l'animation (swing, linear ou un modèle issu d'un plugin) ;
  + complete : fonction appelée lorsque l'animation est terminée ;
  + step : fonction appelée à chaque étape de l'animation ;
  + queue : valeur booléenne qui indique si l'animation doit (true) ou ne doit pas (false) être placée dans une file d'attente réservée aux animations. Si la valeur false est attribuée à cette propriété, l'animation démarre immédiatement. Dans le cas contraire, elle est en attente de déclenchement.
  + specialEasing : un ou plusieurs couples propriétés CSS/modèle de progression.

# File d’attente et timer :

Les animations jQuery sont asynchrones, elles s’exécutent en tâche de fond.

Si on écrit :

$('sel').animate(…).animate(…);

La deuxième s’exécute après la première. Si on veut qu’elles s’exécutent en même temps on utilisera la syntaxe des animations personnalisées :

$('sel').animate({ prop1: val1, prop2: val2, } …);

Mais elles auront la même vitesse d’exécution. Pour qu’elles en aient une différente chacune :

$('sel').animate({ prop1: val1, prop2: val2, prop3: val3, etc. }, {queue: …});

Exemple :

<button id="enchainer">Enchaîner les animations</button>

<button id="nePasEnchainer">Ne pas enchaîner les animations</button><br />

<button id="executerEnMemeTemps">Exécuter les animations en même temps</button>

<button id="etatInitial">État initial</button><br /><br />

<img src="logo.png" style="border: 2px black solid;">

<script src="jquery.js"></script>

<script>

$(function() {

$('#enchainer').click( function() {

$('img').animate({ 'border-width': '100'}, 1500 )

.animate({ 'width': '-=100'}, 1500);

});

$('#nePasEnchainer').click( function() {

$('img').animate({ 'border-width': '100'}, { queue: false, duration: 1500 })

.animate({ 'width': '-=100'}, 1500);

});

$('#executerEnMemeTemps').click( function() {

$('img').animate({ 'border-width': '100', 'width': '-=100' }, 1500);

});

$('#etatInitial').click( function() {

$('img').css({'border-width': '2px', width: '200'});

});

});

</script>

* Quand queue (file d’attente) est à false, les deux s’appliquent en même temps, si elles ont la même durée, elles s’exécuteront exactement en même temps.
* Les boutons 2et 3 produisent le même résultat alors que le code 3 est plus simple. Mais même s’ils s’exécutent en même temps, on réduit la taille de 100 pixel de l’image en 1.5s d’un côté et on l’augmente de 100 celle des bordures en 3s, il y aura donc un décalage. Alors qu’avec le code 2, on a un timer unique pour toutes les instructions.

## Etat de la file d’attente :

Queue() renvoie un tableau avec toutes les animations en attente. Donc on peut connaître le nombre d’animation avec queue.length.

Dans l’ordre :

var resultat = $(‘#monElement’).queue().

Console.log(« il y a : »+Resultat.length+ « elements en attente ») ;

Exemple :

<button id="droite">Droite</button>

<button id="gauche">Gauche</button>

<button id="bas">Bas</button>

<button id="haut">Haut</button>

<button id="etatFile">Etat de la file d'attente</button><br />

<span id="infos">Cliquez sur Etat de la file d'attente</span><br /><br />

<img src="logo.png" style="position: relative;">

<script src="jquery.js"></script>

<script>

$(function() {

$('#droite').click( function() {

$('img').animate({left: '+=50'}, 2000);

});

$('#gauche').click( function() {

$('img').animate({left: '-=50'}, 2000);

});

$('#bas').click( function() {

$('img').animate({top: '+=50'}, 2000);

});

$('#haut').click( function() {

$('img').animate({top: '-=50'}, 2000);

});

$('#etatFile').click(function() {

var n = $('img').queue();

$('#infos').text('Nombre d\'animations dans la file d\'attente : ' + n.length);

});

});

</script>

## Manipuler la file d’attente :

* queue() ajoute une animation dans la file d'attente ;
* dequeue() joue puis supprime une animation de la file d'attente ;
* clearQueue() vide la file d'attente.

Exemple :

<button id="ajouter">Ajouter animation</button>

<button id="annuler">Annuler la file d'attente</button><br />

<button id="remplacer">Remplacer la file d'attente</button>

<button id="retour">Ajouter une fonction de retour</button><br />

<img src="bon.gif" id="bon" style="position: relative;">

<img src="mauvais.gif" id="mauvais" style="position: relative;">

<script src="jquery.js"></script>

<script>

$(function() {

$('#ajouter').click( function() {

$('#bon').toggle(5000)

.queue(function() {

$('#mauvais').animate({left: '+=200'}, 'slow')

.animate({top: '+=200'}, 'slow')

.animate({left: '-=200'}, 'slow')

.animate({top: '-=200'}, 'slow');

});

});

$('#annuler').click( function() {

$('img').clearQueue();

});

$('#remplacer').click( function() {

$('#mauvais').css('left', 200).css('top', 200);

$('#mauvais').queue(function() {

$(this).animate({top: '-=200'}, 'slow')

.animate({top: '+=200', 'left': '-=200'}, 'slow')

.animate({top: '-=200'}, 'slow');

$(this).dequeue();

});

});

$('#retour').click( function() {

$('img').queue(function() {

alert('Animation terminée.');

$(this).dequeue();

});

});

});

</script>

Avec ce code on peut :

* Jouer une animation, puis, lorsqu'elle sera terminée, ajouter d'autres animations dans la file d'attente avec la méthode queue() ;
* Supprimer le contenu de la file d'attente ;
* Remplacer le contenu de la file d'attente ;
* Ajouter une fonction de retour à la file d'attente.

## Répéter une animation sans fin :

Exemple :

<style type="text/css">

#balle {

width: 10px;

height: 10px;

background-color: red;

border: black 2px solid;

border-radius: 10px;

position: relative;

}

</style>

<div id="balle"></div>

<script src="jquery.js"></script>

<script>

$(function() {

function bis() {

$('#balle').animate({left: '+=200'}, 'slow')

.animate({top: '+=200'}, 'slow')

.animate({left: '-=200'}, 'slow')

.animate({top: '-=200'}, 'slow', bis);

};

bis();

});

</script>

Grace à un appel récursif sur la fonction sans condition de sortie.

## Arrêter et reprendre une animation :

On utilise :

$('sel').stop(efface, fin);

… où :

* sel est un sélecteur jQuery ;
* efface est une valeur booléenne qui indique si les animations en attente d'exécution doivent être (true) ou non (false) supprimées de la file d'attente ;
* fin est une valeur booléenne qui indique si l'animation doit prendre son état final (true) ou non (false). Si ce paramètre n'est pas spécifié, l'animation reste dans l'état où elle se trouvait au moment de son arrêt.

Exemple :

<button id="stopFin">Stop et fin</button>

<button id="stopAnnuleFin">Stop, annule et fin</button>

<button id="stop">Stop</button>

<button id="reprise">Reprise</button><br /><br />

<img src="image.png" style="position: relative;">

<script src="jquery.js"></script>

<script>

$(function() {

$('img').animate({left: '+=500'}, 2000).animate({top: '+=300'}, 2000);

$('#stopFin').click( function() {

$('img').stop(false, true);

});

$('#stopAnnuleFin').click( function() {

$('img').stop(true, true);

});

$('#stop').click( function() {

$('img').stop(true, false);

});

$('#reprise').click( function() {

$('img').css('left', 0).css('top', 0);

$('img').animate({left: '+=500'}, 2000).animate({top: '+=300'}, 2000);

});

});

</script>

## Mettre en place un timer :

On peut utiliser un timer entre les instructions au lieu de les répéter immédiatement.

function nom() {

// Une ou plusieurs instructions JavaScript et/ou jQuery

}

setInterval(nom, période);

… où :

* nom est le nom de la fonction qui doit être exécutée périodiquement ;
* durée est la période (c'est-à-dire le temps) entre deux exécutions consécutives des instructions contenues dans la fonction.

### Exemple avec l’horloge :

<span id="heure"></span>

<script src="jquery.js"></script>

<script>

$(function() {

function Horloge() {

var laDate = new Date();

var h = laDate.getHours() + ":" + laDate.getMinutes() + ":" + laDate.getSeconds();

$('#heure').text(h);

}

setInterval(Horloge, 1000);

});

</script>

### Une animation sans fin :

<!DOCTYPE html>

<style type="text/css">

#balle {

width: 10px;

height: 10px;

background-color: red;

border: black 2px solid;

border-radius : 10px;

position: relative;

}

</style>

<body>

<div id="balle"></div>

<script src="jquery.js"></script>

<script>

$(function() {

function bis() {

$('#balle').animate({left: '+=200'}, 'slow')

.animate({top: '+=200'}, 'slow')

.animate({left: '-=200'}, 'slow')

.animate({top: '-=200'}, 'slow');

};

setInterval(bis, 2400);

});

</script>

# Textes et images