jQuery :

Table des matières

[Rappels CSS 5](#_Toc467228605)

[Quelle est la nature de l'objet retourné ? 5](#_Toc467228606)

[Plus loin dans la sélection d’éléments : 6](#_Toc467228607)

[Sélecteurs d’attributs 6](#_Toc467228608)

[Sélecteurs hiérarchiques 6](#_Toc467228609)

[Pseudo-sélecteurs d’éléments sélectionnés 7](#_Toc467228610)

[Sélecteurs d’éléments particuliers 7](#_Toc467228611)

[Pseudo-sélecteurs d’éléments liés aux formulaires : 7](#_Toc467228612)

[Pseudo-sélecteurs d’éléments liés aux tableaux 8](#_Toc467228613)

[Modifier contenu d’un élément 9](#_Toc467228614)

[Getter et Setter : 9](#_Toc467228615)

[Modifier les attributs : 9](#_Toc467228616)

[Lire ou ajouter : 10](#_Toc467228617)

[Supprimer : 10](#_Toc467228618)

[Travailler avec l’attribut class : 10](#_Toc467228619)

[Test de l’existence d’une classe : 10](#_Toc467228620)

[Travailler avec les formulaires : 11](#_Toc467228621)

[Travailler avec des valeurs stockées : 11](#_Toc467228622)

[Position et taille des éléments : 12](#_Toc467228623)

[Modifier la position : 12](#_Toc467228624)

[Modifier la taille : 13](#_Toc467228625)

[Connaitre les dimensions : 15](#_Toc467228626)

[Associer des données aux balises : 15](#_Toc467228627)

[Insérer et remplacer des éléments du DOM 16](#_Toc467228628)

[Insérer du contenu 16](#_Toc467228629)

[Remplacer du contenu : 16](#_Toc467228630)

[Insérer des éléments : 16](#_Toc467228631)

[Déplacer du contenu : 17](#_Toc467228632)

[Dupliquer des éléments : 17](#_Toc467228633)

[Entourer des éléments : 18](#_Toc467228634)

[Supprimer des éléments : 18](#_Toc467228635)

[La gestion évènementielle 19](#_Toc467228636)

[La souris : 19](#_Toc467228637)

[Which et type : 19](#_Toc467228638)

[Le clavier : 20](#_Toc467228639)

[Les éléments : 21](#_Toc467228640)

[Focus() et blur() : 22](#_Toc467228641)

[Changements de l’écran : 22](#_Toc467228642)

[Changement du contenu d’un élément : 23](#_Toc467228643)

[Les pages : 23](#_Toc467228644)

[Evènement personnalisés : 24](#_Toc467228645)

[Gérer plusieurs évènements : 24](#_Toc467228646)

[Désactiver un évènement : 25](#_Toc467228647)

[Etiqueter plusieurs évènements : 25](#_Toc467228648)

[Gestion évènementielle unique : 26](#_Toc467228649)

[Déclenchement d’évènement automatique : 26](#_Toc467228650)

[Créer des évènements personnalisés : 27](#_Toc467228651)

[Délégation d’évènements : 28](#_Toc467228652)

[Animations et effets 29](#_Toc467228653)

[Apparition et disparition : 29](#_Toc467228654)

[Apparition et disparition en cascade : 30](#_Toc467228655)

[Fondus enchaînés : 30](#_Toc467228656)

[Aller plus loin : 31](#_Toc467228657)

[Déplier /replier des éléments : 31](#_Toc467228658)

[Désactiver des éléments : 32](#_Toc467228659)

[Définir un délai avant animation : 32](#_Toc467228660)

[Animations personnalisées : 32](#_Toc467228661)

[Première syntaxe : 32](#_Toc467228662)

[Deuxième syntaxe : 33](#_Toc467228663)

[File d’attente et timer : 34](#_Toc467228664)

[Etat de la file d’attente : 35](#_Toc467228665)

[Manipuler la file d’attente : 36](#_Toc467228666)

[Répéter une animation sans fin : 37](#_Toc467228667)

[Arrêter et reprendre une animation : 38](#_Toc467228668)

[Mettre en place un timer : 39](#_Toc467228669)

[Exemple avec l’horloge : 39](#_Toc467228670)

[Une animation sans fin : 40](#_Toc467228671)

[Textes et images 41](#_Toc467228672)

[Les chaînes de caractères : 41](#_Toc467228673)

[Position d’un caractère dans une chaîne : 41](#_Toc467228674)

[Recherche et remplacement de texte : 42](#_Toc467228675)

[Les images : 43](#_Toc467228676)

[Agrandissement au survol : 43](#_Toc467228677)

[Formulaires et tableaux 45](#_Toc467228678)

[Formulaires : 45](#_Toc467228679)

[Donner le focus : 45](#_Toc467228680)

[Savoir qui a le focus : 45](#_Toc467228681)

[Mettre en évidence un élément qui a le focus : 46](#_Toc467228682)

[Vider un formulaire : 46](#_Toc467228683)

[Valider un formulaire : 47](#_Toc467228684)

[Les tableaux : 47](#_Toc467228685)

[Trouver les éléments du tableau : 47](#_Toc467228686)

[Recopier un tableau avec traitement 48](#_Toc467228687)

[Chercher un élément dans le tableau 49](#_Toc467228688)

[Re 50](#_Toc467228689)

[Valeur retournée 50](#_Toc467228690)

[Regrouper les infos de deux tableaux : 50](#_Toc467228691)

[Un jeu en jQuery 50](#_Toc467228692)

# 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 sur la souris. 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

## Les chaînes de caractères :

La fonction $.trim() supprime les espaces en début et fin de caractère.

Pour illustrer on créer un exemple à partir de d’un texte dans une balise <pre> (qui conserve espaces et retour à la ligne.

Exemple :

Tapez du texte dans la zone de texte en le faisant commencer et/ou finir par des espaces, puis cliquez sur le bouton.<br><br>

<input type="text" id="texte" />

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

<pre id="resultat"></pre>

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

<script>

$(function() {

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

var leTexte = $('#texte').val();

$('#resultat').html('Texte original : "' + leTexte + '"' +

'<br>Après la fonction trim() : "' + $.trim(leTexte) + '"');

});

});

</script>

### Position d’un caractère dans une chaîne :

CharAt() permet de savoir exactement quel est le caractère à un endroit précis.

Il s’écrit comme ça :

var unCaractere = chaine.charAt(position);

Exemple :

Tapez du texte dans la première zone de texte, une position dans la deuxième zone de texte, puis cliquez sur le bouton.<br><br>

Texte : <input type="text" id="texte" /><br>

Position : <input type="text" id="position" /><br>

<button id="action">Cliquez ici</button><br>

<div id="resultat"></div>

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

<script>

$(function() {

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

var leTexte = $('#texte').val();

var laPosition = $('#position').val();

var leResultat = 'Le caractère en position ' + laPosition + ' est un "' + leTexte.charAt(laPosition) + '"';

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

});

});

</script>

### Recherche et remplacement de texte :

Sélection des éléments qui contiennent un mot ou un texte avec :contains()

Exemple :

$('div:contains("rouge")').css('background-color', 'red');

Ainsi que remplacer du texte avec .replaceWith() :

$('sel').replaceWith('contenu');

Exemple :

<img id="un" src="girafe.jpg"><br>

<button id="changement">Remplacer l'image</button>

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

<script>

$(function() {

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

$('#un').replaceWith('<img id="unbis" src="chat.jpg">');

});

});

</script>

## Les images :

### Agrandissement au survol :

<img id="montagne" src="montagnepetit.jpg">

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$(function() {

$('#montagne').mouseover(function() {

$(this).attr('src','montagne.jpg');

});

$('#montagne').mouseout(function() {

$(this).attr('src','montagnepetit.jpg');

});

});

</script>

Ou avec effet :

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript">

$(function() {

$('#montagnePetit').mouseover(function() {

$(this).fadeOut(1000);

$('#montagneGrand').fadeIn(1000);

});

$('#montagneGrand').mouseout(function() {

$(this).fadeOut(1000);

$('#montagnePetit').fadeIn(1000);

});

});

</script>

Ou en mode carrousel :

<style type="text/css">

img

{

padding: 5px;

}

</style>

<img src="miniPaysage1.jpg" id="paysage1.jpg" class='miniature'>

<img src="miniPaysage2.jpg" id="paysage2.jpg" class='miniature'>

<img src="miniPaysage3.jpg" id="paysage3.jpg" class='miniature'>

<img src="miniPaysage4.jpg" id="paysage4.jpg" class='miniature'>

<img src="miniPaysage5.jpg" id="paysage5.jpg" class='miniature'><br />

<img id="grand" src="paysage1.jpg" />

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

<script>

$(function() {

$('.miniature').css('border','5px white solid');

$('img:first').css('border','5px black solid');

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

$('img').css('border','5px white solid');

$(this).css('border','5px black solid');

var nom = $(this).attr('id');

$('#grand').attr('src',nom);

});

});

</script>

Ou un diaporama automatique :

<style type="text/css">

img

{

position: absolute; // Les images vont se superposer

left: 0px; // A gauche

top: 0px; // et en haut de la feuille

display: none; // Par défaut, elles ne seront pas affichées

}

</style>

<img src="paysage5.jpg" id="img5">

<img src="paysage4.jpg" id="img4">

<img src="paysage3.jpg" id="img3">

<img src="paysage2.jpg" id="img2">

<img src="paysage1.jpg" id="img1">

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

<script>

$(function() {

var i=0;

affiche();

function affiche() {

i++;

if (i==1) precedent = '#img5'

else precedent = '#img' + (i-1);

var actuel = '#img' + i;

$(precedent).fadeOut(2000);

$(actuel).fadeIn(2000);

if (i==5) i=0;

}

setInterval(affiche, 2000);

});

</script>

# Formulaires et tableaux

## Formulaires :

### Donner le focus :

<input type="text" id="user">

$('#user').focus();

### Savoir qui a le focus :

<span id="status">Cliquez sur un des éléments du formulaire</span><br /><br />

<form>

<input type="text" id="zone1"><br />

<input type="text" id="zone2"><br />

<input type="text" id="zone3"><br />

<textarea id="zone4"></textarea>

</form>

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

<script>

$(function() {

var leFocus;

$('input, textarea').focus( function() {

leFocus = $(this).attr('id');

$('#status').text(leFocus + ' a le focus');

});

});

</script>

### Mettre en évidence un élément qui a le focus :

<span id="status">Cliquez sur un des éléments du formulaire</span><br /><br />

<form>

<input type="text" id="zone1"><br />

<input type="text" id="zone2"><br />

<input type="text" id="zone3"><br />

<textarea id="zone4"></textarea>

</form>

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

<script>

$(function() {

var leFocus;

$('input, textarea').focus( function() {

leFocus = '#' + $(this).attr('id');

$(leFocus).css('background-color', '#afc');

});

$('input, textarea').blur( function() {

leFocus = '#' + $(this).attr('id');

$(leFocus).css('background-color', '#fff');

});

});

</script>

### Vider un formulaire :

Il suffit de créer :

<input type="reset" id="annuler" value="Annuler">

Ou alors on peut le faire en jQuery :

<button id="raz">RAZ du formulaire</button>

$('#raz').click(efface\_formulaire);

function efface\_formulaire () {

$(':input')

.not(':button, :submit, :reset, :hidden')

.val('')

.prop('checked', false)

.prop('selected', false);

}

* Un sélecteur sélectionne tous les inputs sauf les bouton, reset, submit et hidden.
* On vide les champs
* On désélectionne tous.

### Valider un formulaire :

Utiliser un plugin (voir plus loin).

## Les tableaux :

### Trouver les éléments du tableau :

$.gep()

var tableau2 = $.grep(tableau, function(élément, index) { … }, inv);

… où :

* tableau est le tableau qui contient les données à filtrer ;
* élément et index sont les éléments et l'index des éléments du tableau ;
* inv indique si le critère doit (true) ou ne doit pas (false ou n'est pas spécifié) être inversé.

Exemple :

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

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

<button id="filtre1">Après le cinquième</button>

<button id="filtre2">Différent de Mathis, Hugo et Yanis</button>

<button id="filtre3">Avant le cinquième</button>

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

<script>

$(function() {

var tableau = ['Luca', 'Emma', 'Mathis', 'Jade', 'Léa', 'Enzo', 'Chloé', 'Nathan', 'Manon', 'Noah', 'Sarah ', 'Louis', 'Luna', 'Kylian', 'Clara', 'Ethan', 'Camille', 'Hugo', 'Lylou', 'Théo', 'Zoé', 'Yanis', 'Maélys'];

var tableau2;

$('#un').text('Données originales : ' + tableau.join(', ')); // recopie le tableau et sépare par ,

$('#filtre1').click(function() {

tableau2 = $.grep(tableau, function(el,ind) {

return (ind > 4);

});

$('#deux').text('Après le cinquième : ' + tableau2.join(', '));

});

$('#filtre2').click(function() {

tableau2 = $.grep(tableau, function(el,ind) {

return (el != 'Mathis' && el != 'Hugo' && el !='Yanis');

});

$('#deux').text('Différent de Mathis, Hugo et Yanis : ' + tableau2.join(', '));

});

$('#filtre3').click(function() {

tableau2 = $.grep(tableau, function(el,ind) {

return (ind > 4);

}, true); // comme il y a true, c’est inversé, on prend <4 et non > 4

$('#deux').text('Avant le cinquième : ' + tableau2.join(', '));

});

});

</script>

On a utilisé la fonction tab.join(‘ ; ‘) ; qui sépare les élements du tableau par la paramètre de la fonction.

### Recopier un tableau avec traitement

La fonction $.map() recopie un tableau en appliquant ou non un traitement.

tableau2 = $.map(tableau, function(el, ind) { … });

… où :

* tableau est le tableau qui contient les données à recopier ;
* élément et index sont les éléments et l'index des éléments du tableau ;
* tableau2 est le tableau dans lequel sont stockés les résultats de la fonction map().

Exemple :

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

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

<button id="copie1">Prénoms en majuscules</button>

<button id="copie2">Index et prénoms en minuscules</button>

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

<script>

$(function() {

var tableau = ['Luca', 'Emma', 'Mathis', 'Jade', 'Léa', 'Enzo', 'Chloé'];

var tableau2;

$('#un').text('Données originales : ' + tableau.join(', '));

$('#copie1').click(function() {

tableau2 = $.map(tableau, function(el,ind) {

return (el.toUpperCase());

});

$('#deux').text('Prénoms en majuscules : ' + tableau2.join(', '));

});

$('#copie2').click(function() {

tableau2 = $.map(tableau, function(el,ind) {

return (ind + ' : ' + el.toLowerCase());

});

$('#deux').text('Index et prénoms en minuscules : ' + tableau2.join(', '));

});

});

</script>

### Chercher un élément dans le tableau

Fonction $.inArray()

var position = $.inArray('valeur', tableau, index)

… où :

* valeur est la valeur recherchée ;
* tableau est le tableau dans lequel doit se faire la recherche ;
* index, s'il est précisé, est le numéro de la cellule à partir de laquelle doit commencer la recherche (attention, la première cellule a pour index 0) ;
* position est la première position de la valeur dans le tableau. Si la recherche est infructueuse, inArray() retourne -1.

Exemple :

var tableau = ['Luca', 'Emma', 'Mathis', 'Jade', 'Léa', 'Enzo', 'Chloé'];

| Re | Valeur retournée |
| --- | --- |
| $.inArray('Emma',tableau) | 1 |
| $.inArray('Léa',tableau) | 4 |
| $.inArray('Luca',tableau, 5) | -1 car Luca se trouve en position 0 et non après la position 5 |
| $.inArray('Alfred',tableau) | -1 car Alfred n'est pas dans le tableau |

### Regrouper les infos de deux tableaux :

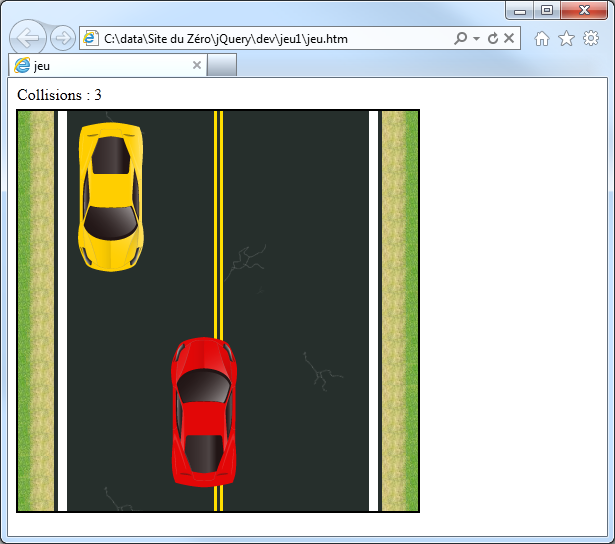
Fonction $.merge()

$.merge(tableau1, tableau2);

Le tableau 1 contient maintenant ses données suivi des données du tableau 2.

# Un jeu en jQuery

Le jeu ressemblera à ça :



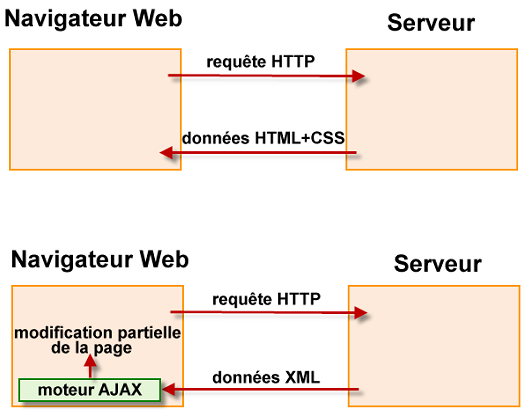
La route défile du bas vers le haut. Le joueur pilote la voiture jaune et doit éviter les voitures rouges qui apparaissent aléatoirement sur l'écran. La voiture jaune se dirige avec les touches **Droite** et **Gauche** du clavier. La zone de jeu n'est autre qu'une balise <div> dans laquelle on place les différents éléments graphiques :

* Deux portions de route #fond1 et #fond2, toutes deux de classe .fond ;
* La voiture jaune #vj ;
* La voiture rouge #vr.

Code complet sur le cours.

# Ajax

On peut voyager de page en page, pa de problèmes, mais quand on ne veut mettre à jour qu’une partie de la page : 🡺 AJAX.



AJAX est l'acronyme d'Asynchronous JavaScript and XML

## Charger un fichier :

Fonction load() :

$('sel').load('nom\_page', function() {

//une ou plusieurs instructions

});

… où :

* Sel est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
* nom\_page est le nom d'une page Web quelconque dont le contenu sera utilisé pour effectuer la mise à jour ;
* Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

Exemple :

<style type="text/css">

div { width: 400px; height: 300px; float: left; margin: 5px; }

#premier { background-color: #F6E497; }

#troisieme { background-color: #CAF1EC; }

#quatrieme { background-color: #F1DBCA; }

</style>

<button id="majPremier">Mise à jour première zone</button>

<button id="majDeuxieme">Mise à jour deuxième zone</button><br /><br />

<div id="premier">

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

</div>

<div id="deuxieme">

<img src="image1.jpg">

</div>

<div id="troisieme">

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</div>

<div id="quatrieme">

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

</div>

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

<script>

$(function() {

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

$('#premier').load('maj1.html', function() {

alert('La première zone a été mise à jour');

});

});

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

$('#deuxieme').load('maj2.html', function() {

alert('La deuxième zone a été mise à jour');

});

});

});

</script>

## Charger une partie d’un fichier :

Pour ne sélectionner qu’une partie, on garde la même syntaxe et on met un espace puis un deuxième sélecteur :

$('sel').load('nom\_page sel2', function() {

//Une ou plusieurs instructions

});

… où :

* Sel est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
* nom\_page est le nom d'une page Web quelconque dont le contenu sera utilisé pour effectuer la mise à jour ;
* sel2 est un sélecteur jQuery quelconque, sans le signe $ ni les parenthèses, qui permettra d'isoler certaines données dans la page nom\_page;
* Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

## Passez des paramètres à un programme PHP :

### Première forme :

$('sel').load(url,param);

… où :

* Sel est un sélecteur jQuery quelconque qui permet d'identifier l'élément (ou les éléments) à mettre à jour ;
* url est le nom de la page PHP qui sera utilisée pour faire la mise à jour ;
* param est une chaîne qui contient un ou plusieurs couples paramètres/valeurs. Par exemple, si param vaut « id=5, p=14 » et url vaut http://site.fr/page.php, la page utilisée pour la mise à jour sera http://site.fr/page.php?id=5&p=14.

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Ajax - Load</title>

</head>

<body>

<input type="text" id="ref">

<button id="action">Afficher</button><br />

<div id="r">Entrez un nombre compris entre 1 et 10 pour afficher un proverbe chinois</div>

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

<script>

$(function() {

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

var param = 'l=' + $('#ref').val();

$('#r').load('http://www.proverbes.php',param);

});

});

</script>

</body>

</html>

Avec la partie PHP :

<?php

$proverbe = array("On ne rassasie pas un chameau en le nourrissant à la cuillère.",

"Connaître son ignorance est la meilleure part de la connaissance.",

"Une maison en paille où l'on rit, vaut mieux qu'un palais où l'on pleure.",

"Le vrai voyageur ne sait pas où il va.",

"Point n'est besoin d'élever la voix quand on a raison.",

"Un ami c'est une route, un ennemi c'est un mur.",

"Un peu de parfum demeure toujours sur la main qui te donne des roses.",

"Si élevé que soit l'arbre, ses feuilles tombent toujours à terre.",

"Si ce que tu as à dire n'est pas plus beau que le silence, tais toi.",

"Trois coupes de vin font saisir une doctrine profonde.");

$l=$\_GET["l"];

if (($l != "") && ($l>0) && ($l<11))

{

echo "<u>Proverbe chinois N° ".$l."</u><br><br>";

echo "<b>".$proverbe[$l-1]."</b>";

}

else

echo "<font color=red>Entrez un nombre compris entre 1 et 10 !</font>";

?>

### Deuxième méthode : passer un objet :

Pour passer un objet :

$('sel').load('http://www.site.com/page.php',{ id:50, nom: 'durand'});

Ici on utilise POST() et non LOAD().

## Get et post :

### La fonction $.get

$.get(adresse, données, function() {

// Une ou plusieurs instructions

});

… où :

* adresse est le nom d'une page Web quelconque dont le contenu sera récupéré par la fonction get();
* données représente les données à passer à la page Web par une requête http GET;
* Si elle est précisée, la fonction de rappel est, comme toujours, exécutée lorsque la méthode a été exécutée, c'est-à-dire lorsque l'élément (ou les éléments) a (ont) été mis à jour.

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Ajax - Get</title>

</head>

<body>

<button id="action">Lancer la requête HTTP GET</button><br />

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

<script>

$(function() {

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

$.get('proverbes.php?l=9', function(data) {

alert(data);

});

});

});

</script>

</body>

</html>

### $.post() :

Exemple :

$.post('traiteFormulaire.php', { nom: 'Pierre34', heure: '2pm', post='Un peu de texte récupéré dans un formulaire HTML et destiné à être posté dans un forum.' },

function(data) {

alert(data);

});

## Faire patienter l’utilisateur avec une animation :

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Ajax - Load</title>

</head>

<body>

<input type="text" id="ref">

<button id="action">Afficher</button><br />

<div id="r">Entrez un nombre compris entre 1 et 10 pour afficher un proverbe chinois</div>

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

<script>

$(function() {

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

$('#r').html('<img src="http://www.mediaforma.com/sdz/jquery/ajax-loader.gif">');

var param = 'l=' + $('#ref').val();

$('#r').load('http://www.mediaforma.com/sdz/jquery/data.php',param);

});

});

</script>

</body>

</html>

## Charger un script et données JSON :

### Script :

Pour charger un script, il suffit de renseigner son adresse :

$.getScript('adresse');

Ou :

$.getScript('adresse', function() {

// Une ou plusieurs instructions

});

### JSON :

Pour charger des données JSON : (JavaScript Object Notation) :

Exemple de notation JSON :

{

  'menu': 'Fichier',

  'commande': [

      {

          'nomCde': 'Nouveau',

          'action':'CreateDoc'

      },

      {

          'nomCde': 'Ouvrir',

          'action': 'OpenDoc'

      },

      {

          'nomCde': 'Enregistrer sous',

          'action': 'SaveAs'

      }

      {

          'nomCde': 'Fermer',

          'action': 'CloseDoc'

      }

   ]

}

C’est un ensemble de pair : ‘nom’ : ’valeur’

Exemple :

Soit le fichier JSON :

{

  "nom": "Pierre Durand",

  "age": "27",

  "ville": "Paris",

  "domaine": "HTML5, CSS3, JavaScript"

}

Voici le code HTML et jQuery pour le récupérer avec $.getJSON :

<button id="charger">Charger et traiter les données</button>

<div id="r">Cliquez sur "Charger et traiter les données" pour lancer la lecture et le traitement des données JSON</div>

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

<script>

$(function() {

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

$.getJSON('fichier.json', function(donnees) {

$('#r').html('<p><b>Nom</b> : ' + donnees.nom + '</p>');

$('#r').append('<p><b>Age</b> : ' + donnees.age + '</p>');

$('#r').append('<p><b>Ville</b> : ' + donnees.ville + '</p>');

$('#r').append('<p><b>Domaine de compétences</b> : ' + donnees.domaine + '</p>');

});

});

});

</script>

Ici on met « donnees » en paramètre de fonction pour récupérer les données du fichier JSON. Ainsi si on a un ensemble ‘non’ : ‘valeur’ où ‘nom’ = age, on le récupère avec donnees.age.

## La fonction $.ajax()

Deux syntaxes possibles :

$.ajax(adresse, {options});

$.ajax({options});

… où

* adresse est l'adresse à laquelle la requête doit être envoyée, et options correspond à une ou plusieurs des options suivantes :
  + type : type de la requête, GET ou POST (GET par défaut).
  + url : adresse à laquelle la requête doit être envoyée.
  + data : données à envoyer au serveur.
  + dataType : type des données qui doivent être retournées par le serveur : xml, html, script, json, text.
  + success : fonction à appeler si la requête aboutit.
  + error : fonction à appeler si la requête n'aboutit pas.
  + timeout : délai maximum (en millisecondes) pour que la requête soit exécutée. Si ce délai est dépassé, la fonction spécifiée dans le paramètre error sera exécutée.

Les autres options, [sur cette page](https://api.jquery.com/jQuery.ajax/).

On peut l’utiliser comme les autres fonctions, mais en plus détaillé :

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<title>Ajax - La fonction ajax()</title>

</head>

<body>

<button id="action">Lancer la requête AJAX</button><br />

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

<script>

$(function() {

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

$.ajax({

type: 'GET',

url: 'proverbes.php?l=7',

timeout: 3000,

success: function(data) {

alert(data); },

error: function() {

alert('La requête n\'a pas abouti'); }

});

});

});

</script>

</body>

</html>

## Evènements associés à ajax :

| **Méthode** | **Événement** |
| --- | --- |
| $(document).ajaxSend(function(ev, req, options)) | Requête sur le point d'être envoyée |
| $(document).ajaxStart(function()) | Début d'exécution de la requête |
| $(document).ajaxStop(function()) | Fin de la requête |
| $(document).ajaxSuccess(function(ev, req, options)) | La requête a abouti |
| $(document).ajaxComplete(function(ev, req, options)) | La requête est terminée |
| $(document).ajaxError(function(ev, req, options, erreur)) | La requête n'a pas abouti |

… où :

* ev représente l'événement ;
* req représente la requête ;
* options contient les paramètres passés à la requête ;
* erreur est le nom de l'erreur détectée par jQuery.

Exemple :

<button id="action">Lancer la requête AJAX</button><br /><br />

<div id="donnees" style="background-color: yellow"></div><br />

<div id="message"></div>

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

<script>

$(function() {

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

$(document).ajaxStart(function() {

$('#message').html('Méthode ajaxStart exécutée<br>');

});

$(document).ajaxSend(function(ev, req, options){

$('#message').append('Méthode ajaxSend exécutée, ');

$('#message').append('nom du fichier : ' + options.url + '<br>');

});

$(document).ajaxStop(function(){

$('#message').append('Méthode ajaxStop exécutée<br>');

});

$(document).ajaxSuccess(function(ev, req, options){

$('#message').append('Méthode ajaxSuccess exécutée<br>');

});

$(document).ajaxComplete(function(ev, req, options){

$('#message').append('Méthode ajaxComplete exécutée<br>');

});

$(document).ajaxError(function(ev, req, options, erreur){

$('#message').append('Méthode ajaxError exécutée, ');

$('#message').append('erreur : ' + erreur + '<br>');

});

$('#donnees').load('affiche.htm');

});

});

</script>

# Trouver et utiliser un plugin

## Installer un plugin :

Deux sites de recherches (pour le moment) :

* <http://jquerylist.com>
* <https://plugins.jquery.com>

DL le plugin, extraire le fichier.min.js et .css et les mettre dans le répertoire de travail. Lire la doc. Coder.

## Exemples de plugins :

### Parseur RSS/Atom

On utilise le plugin FeedEk.

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>FeedEk jQuery RSS/ATOM Feed Plugin Demo | jQuery RSS/ATOM Parser Plugin FeedEk Demo</title>

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

<script src="http://momentjs.com/downloads/moment-with-langs.min.js"></script>

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

<link href="css/FeedEk.css" rel="stylesheet" type="text/css" />

<style>

body { font-family: "Helvetica Neue",Helvetica,Arial,sans-serif;background-color: #efefef;font-size: 13px;line-height: 17px !important; }

.rssDiv{float: left;padding-right: 35px;}

ul{width: 300px !important;}

</style>

</head>

<body>

<h1>Le Monde - A la Une</h1>

<div id="divRss"></div>

<script>

$(document).ready(function () {

$('#divRss').FeedEk({

FeedUrl: 'http://www.lemonde.fr/rss/une.xml',

MaxCount: 3

});

});

</script>

</body>

</html>

### Validation de formulaire avec « validation »

<script>

$(document).ready(function() {

$("#commentForm").validate();

});

</script>

Pour définir le type de validation souhaité, on lui affecte la classe correspondante :

* required: champs requis ;
* email: adresse e-mail ;
* url: adresse URL ;
* date: date ;
* number: nombre ;
* creditcard: numéro de carte bancaire.

Exemple :

<input id="cemail" name="email" class="required email" />

Pour afficher des messages français en cas d’erreur :

<script src="../dist/localization/messages\_fr.js"></script>

### Menu déroulant :

Plugin : dropdownPlain.

Une fois le plugin appelé (jquery.dropdownPlain.js), on définit le menu sous cette forme :

<ul class="dropdown">

<li><a href="#">For Facilities</a>

<ul class="sub\_menu">

<li><a href="#">Artificial Turf</a></li>

<li>

<a href="#">Batting Cages</a>

<ul>

<li><a href="#">Indoor</a></li>

<li><a href="#">Outdoor</a></li>

</ul>

</li>

<li><a href="#">Benches &amp; Bleachers</a></li>

<li><a href="#">Communication Devices</a></li>

<li><a href="#">Dugouts</a></li>

<li><a href="#">Fencing &amp; Windscreen</a></li>

etc.

Penser à insérer style.css aussi.

### Cartographie :

Plugin gMap.

<div id="map1" style="width: 800px; height: 600px; border: 1px solid #777; overflow: hidden;"></div>

<script>

$(function() {

$('#map1').gMap();

});

</script>

Pour utiliser google map, il faut 3 script :

<script type="text/javascript" src="jquery.js"></script>

<script type="text/javascript" src="http://maps.google.com/maps?file=api&amp;v=2&amp;sensor=false&amp;key=ABQIAAAA6cQIrMEc9zlaKBjWiPM5rxSjlBXfTSDcGsB79vzL90uiOHMpbBRa1FFoX2YfuQNFvFKxQtpz0ZCeuw&amp;hl=fr"></script>

<script type="text/javascript" src="jquery.gmap-1.1.0-min.js"></script>

Pour lui passer des paramètres :

* La latitude et la longitude : propriétéslatitudeetlongitude;
* Des informations sur le centre de l'affichage : propriétéaddress;
* Le facteur de zoom : propriétézoom(entre 1 et 19) ;
* Le type de la carte : propriétémaptype(G\_NORMAL\_MAP,G\_SATELLITE\_MAP,G\_HYBRID\_MAP,G\_DEFAULT\_MAP\_TYPES,G\_PHYSICAL\_MAP).

Exemple :

$("#map").gMap({ markers: [{

latitude: 43.92,

longitude: 2.14,

html: "Albi",

popup: true }],

maptype: G\_SATELLITE\_MAP,

zoom: 10 });

# jQuery UI

Vaste assemblage de plugin accessibles à travers un seul fichier JavaScript.

Pour y faire référence, on pointe vers un CDN UI : <https://code.jquery.com>

Pour améliorer le rendu on peut s’appuyer sur un thème CSS avec la balise <link> dans l’entête du document.

## Déplacer et redimensionner des éléments :

Méthode draggable().

<style>

div{

width: 150px;

height: 150px;

padding: 0.5em;

border: 1px black solid;

}

</style>

<span>Déplacez les images et le div comme vous l'entendez</span><br /><br />

<img src="miniPaysage1.jpg">

<img src="miniPaysage2.jpg">

<img src="miniPaysage3.jpg">

<div>Déplacez-moi</div>

$(function() {

$('img,div').draggable();

});

## Agrandir des éléments :

Méthode resizable() :

<link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/themes/base/jquery-ui.css">

<img id="redim" src="visage.jpg">

<script src="http://code.jquery.com/jquery.min.js"></script>

<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>

<script>

$(function() {

$("#redim").resizable();

});

</script>

## L’accordéon :

Une seule case visible à la fois :

<div id="accordeon">

<h3><a href="#">Titre du bloc 1</a></h3>

<div>Contenu du bloc 1</div>

<h3><a href="#">Titre du bloc 2</a></h3>

<div>Contenu du bloc 2</div>

…

</div>

<html>

$(document).ready(function() {

$("#accordeon").accordion();

});

</html>

## Sélection de date :

Méthode datepicker()

<link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/themes/base/jquery-ui.css">

Date <input type="text" id="datepicker">

<script src="http://code.jquery.com/jquery.min.js"></script>

<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>

<script>

$(function() {

$( "#datepicker").datepicker();

});

</script>

Fait apparaître un calendrier quand on clique sur un champ date.

Astuce : pour avoir un calendrier français :

$.datepicker.regional['fr'] = {

closeText: 'Fermer',

prevText: 'Précédent',

nextText: 'Suivant',

currentText: 'Aujourd\'hui',

monthNames: ['Janvier','Février','Mars','Avril','Mai','Juin','Juillet','Août','Septembre','Octobre','Novembre','Décembre'],

monthNamesShort: ['Janv.','Févr.','Mars','Avril','Mai','Juin','Juil.','Août','Sept.','Oct.','Nov.','Déc.'],

dayNames: ['Dimanche','Lundi','Mardi','Mercredi','Jeudi','Vendredi','Samedi'],

dayNamesShort: ['Dim.','Lun.','Mar.','Mer.','Jeu.','Ven.','Sam.'],

dayNamesMin: ['D','L','M','M','J','V','S'],

weekHeader: 'Sem.',

dateFormat: 'dd/mm/yy',

firstDay: 1,

isRTL: false,

showMonthAfterYear: false,

yearSuffix: ''};

Pour indiquer que l’on veut utiliser le calendrier français :

$.datepicker.setDefaults($.datepicker.regional['fr']);

## Boites de dialogues :

Elles sont de type alert().

1. Créez une balise <div>.
2. Définissez le titre de la boîte de dialogue dans son attribut title.
3. Appliquez la méthode dialog() à la balise <div>.

$('sel').dialog();

Si la boîte de dialogue doit être ouverte et fermée plusieurs fois, vous utiliserez d'autres instructions :

$('sel').dialog({ autoOpen: false; }); //Crée une instance de la boîte de dialogue sans l'ouvrir

$('sel').dialog('open'); // Ouvre la boîte de dialogue

$('sel').dialog('close'); // Ferme la boîte de dialogue

D’autres options utilisables dans les paramètres de dialog()

| **Options** | **Signification** |
| --- | --- |
| height et width | Hauteur et largeur de la boîte de dialogue à l'ouverture. |
| modal | Initialisé à true, rend la boîte de dialogue modale (c'est-à-dire interdit l'accès à la page). La valeur par défaut est false. |
| position | Position de la boîte de dialogue sur la page (elle est centrée par défaut). |
| zindex | Z-index de la boîte de dialogue (1000 par défaut). |
| buttons | Un ou plusieurs boutons affichés dans la boîte de dialogue. |

<link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/themes/base/jquery-ui.css">

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium

…

fugiat quo voluptas nulla pariatur?

<div id="dialog" title="Boîte de dialogue de base">

Cette boîte de dialogue peut être redimensionnée, déplacée et fermée.

</div>

<script src="http://code.jquery.com/jquery.min.js"></script>

<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>

<script>

$(*function*() {

$( "#dialog" ).dialog();

});

</script>

| **Options** | **Signification** |
| --- | --- |
| height et width | Hauteur et largeur de la boîte de dialogue à l'ouverture. |
| modal | Initialisé à true, rend la boîte de dialogue modale (c'est-à-dire interdit l'accès à la page). La valeur par défaut est false. |
| position | Position de la boîte de dialogue sur la page (elle est centrée par défaut). |
| zindex | Z-index de la boîte de dialogue (1000 par défaut). |
| buttons | Un ou plusieurs boutons affichés dans la boîte de dialogue. |

Exemple :

<link rel="stylesheet" href="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.12/themes/base/jquery-ui.css">

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium

…

fugiat quo voluptas nulla pariatur?

<div id="dialog" title="Boîte de dialogue de base">

Cette boîte de dialogue peut être redimensionnée, déplacée et fermée.

</div>

<script src="http://code.jquery.com/jquery.min.js"></script>

<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>

<script>

$(*function*() {

$( "#dialog" ).dialog();

});

</script>

Pour une boite de dialogue modale :

<script>

$(*function*() {

$( "#dialog" ).dialog({

modal: true,

buttons: {

"Oui": *function*() {

$('body').css('background', 'yellow');

$( this ).dialog( "close" );

},

"Non": *function*() {

$( this ).dialog( "close" );

}

}

});

});

</script>

## Les onglets :

On applique la méthode tabs().

<div id="onglets">

<ul>

<li><a href="#onglet-1">Titre onglet 1</a></li>

<li><a href="#onglet-2">Titre onglet 2</a></li>

<li><a href="#onglet-3">Titre onglet 3</a></li>

</ul>

<div id="onglet-1">

<!--contenu -->

</div>

<div id="onglet-2">

<!--contenu -->

</div>

<div id="onglet-3">

<!--contenu -->

</div>

</div>

<script>

$(*function*() {

$('#onglets').tabs();

});

</script>

Le contenu peut aussi être actualisé en AJAX :

<ul>

<li><a href="#onglet-1">Titre onglet 1</a></li>

<li><a href="http://www.site.com/page2.htm">Titre onglet 2</a></li>

<li><a href=" http://www.site.com/page3.php">Titre onglet 3</a></li>

</ul>

## Animer la couleur des éléments :

* backgroundColor
* borderBottomColor
* borderLeftColor
* borderRightColor
* borderTopColor
* color
* outlineColor

La syntaxe de la méthode animate() ne change pas :

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

//Une ou plusieurs instructions

});

… où :

* sel est un sélecteur jQuery ;
* prop1, prop2, prop3 sont des propriétés CSS et val1, val2, val3 les valeurs associées ;
* durée est la durée de l'animation ;
* modèle est le modèle de progression de l'animation ;
* function() contient une ou plusieurs instructions qui seront exécutées lorsque l'animation sera terminée.
* .
* <style>
* #contenu
* {
* *border*: 4px gray solid;
* *background-color*: #aaeae1;
* *color*: black;
* *width*: 100px;
* *position*: relative;
* }
* #contenu h3
* {
* *margin*: 0;
* *padding*: 0.4em;
* *text-align*: center;
* *background-color*: #777;
* }
* </style>
* <div id="contenu">
* <h3>Cliquez ici</h3>
* At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint occaecati cupiditate non provident, similique sunt in culpa qui officia deserunt mollitia animi, id est laborum et dolorum fuga.
* </div>
* <script src="http://code.jquery.com/jquery.min.js"></script>
* <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"></script>
* <script>
* $(*function*() {
* $('#contenu').toggle(
* *function*() {
* $( '#contenu').animate({
* backgroundColor: '#fff',
* color: 'red',
* left: '+=200',
* width: 500
* }, 1000 );
* },
* *function*() {
* $('#contenu').animate({
* backgroundColor: '#aaeae1',
* color: 'black',
* left: '-=200',
* width: 100
* }, 1000 );
* }
* );
* });
* </script>

## Modèle de progression :

Autre que linéar et swing ,jQuery UI permet un plus large éventail :

|  |  |  |  |
| --- | --- | --- | --- |
| easeInOutQuad | easeInCubic | easeOutCubic | easeInOutCubic |
| easeInQuart | easeOutQuart | easeInOutQuart | easeInQuint |
| easeOutQuint | easeInOutQuint | easeInSine | easeOutSine |
| easeInOutSine | easeInExpo | easeOutExpo | easeInOutExpo |
| easeInCirc | easeOutCirc | easeInOutCirc | easeInElastic |
| easeOutElastic | easeInOutElastic | easeInBack | easeOutBack |
| easeInOutBack | easeInBounce | easeOutBounce | easeInOutBounce |

# Créer un plugin :

## Le squelette :

Tout est défini par fn.

jQuery.*fn*.gis = *function*(*param1*, *param2*, *param3*, …)

{

// Une ou plusieurs instructions JavaScript

}

On définit la méthode « gris » ainsi.

### Conflit entre plusieurs méthodes :

En créant un plugin avec des nom déjà utilisés, il peut y avoir un conflit. Pour parer le problème, on utilise jquery au lieu de $ grace à noConflit( ).

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

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

<script>

$.noConflict();

// Ici, vous pouvez insérer une ou plusieurs instructions

// Qui utilisent le signe $ en accord avec les spécifications

// De la bibliothèque uneBibliothequeJavaScript.js

</script>

Utiliser quand même $ à la place de jQuery :

(*function*($) {

// Entrez ici le code de votre plugin jQuery

})(jQuery);

Notre fonction devient donc :

(*function*($) {

$.*fn*.gis = *function*(*paramètres*)

{

…

};

})(jQuery);

### Appliquer la méthode à un sélecteur :

Quand on applique notre méthode à un sélecteur, par exemple une classe, il se peut qu’il y en ait plusieurs :

(*function*($) {

$.*fn*.gis = *function*(*paramètres*)

{

this.each(*function*() {

// Les instructions du plugin

});

};

})(jQuery);

### Continuer de chainer :

On peut appliquer des méthodes les unes derrière les autres car elle renvoie le même objet. Pour continuer de pouvoir chainer, on doit renvoyer le même élément :

(*function*($) {

$.*fn*.gis = *function*(*paramètres*)

{

this.each(*function*() {

// Les instructions du plugin

});

Return this; // On renvoit l’élément appelé

};

})(jQuery);

## Un premier plugin :

On construit un plugin qui met en gras, italique et souligné.

(*function*($) {

$.*fn*.gis = *function*()

{

this.each(*function*() {

$(this).wrap('<b><i><u></u></i></b>');

});

return this;

};

})(jQuery);

Si on enregistre ça dans un fichier.js, on peut l’appelé dans le main e méthode sur un sélecteur.

Exemple :

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>Utilisation du plugin gis</title>

</head>

<body>

<p class="grasItaliqueSouligne">Ce texte devrait apparaître en gras, italique, souligné après avoir cliqué sur le bouton</p>

<p>Ce texte devrait rester inchangé</p>

<p class="grasItaliqueSouligne">Ce texte devrait apparaître en gras, italique, souligné après avoir cliqué sur le bouton</p>

<p>Ce texte devrait rester inchangé</p><br />

<button id="action">Cliquez ici pour utiliser le plugin gis</button>

<script src="http://code.jquery.com/jquery.min.js"></script>

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

<script>

$(*function*() {

$('#action').click(*function*(){

$('.grasItaliqueSouligne').gis(); //On applique la méthode

});

});

</script>

</body>

</html>

## Un plugin plus costaud :

1. ( function($) {
2. $.fn.wideText = function() {
3. return this.each( function() {
4. // Add "wtext" class to each element and then set up the magic
5. var obj = $(this),
6. rtext = obj.addClass( 'wtext' );
7. // Work that magic each time the browser is resized
8. $(window).on( 'resize', function() {
9. obj.css( { 'fontSize': parseInt( obj.css( 'fontSize' ) ) \* ( obj.parent().width() / obj.width() ) + 'px', 'visibility' : 'visible' } );
10. } ).resize();
11. });
12. };
13. } )(jQuery);

Ici le texte s’adapte à la taille du conteneur : ici la fenêtre. On peut l’appliquer à d’autre conteneurs comme des <div> ou autre.

## Try et catch :

En cas de moment sensible, sujet à erreur :

*var* message='';

try

{

alort('un message');

}

catch(err)

{

message='Une erreur s\'est produite.\n\n';

message+='Description : ' + err.message + '\n\n';

message+='Cliquez sur OK pour poursuivre.';

alert(message)

}

## Capturer toutes les erreurs :

*function* gestionErreurs(*err*)

{

alert('Erreur : \n' + err);

return true;

}

window.onerror = gestionErreurs;

## Utiliser la console sous Firefox !

On installe le plugin Firebug sur Firefox et dans l’onglet console on peut accéder à la console.

for (x = (Math.PI)/2; x < (4\*Math.PI); x = x+.2)

{

y = (Math.abs(Math.sin(x)))/x;

window.console.log('y = ' + y);

…

}