**LE FRAMEWORK JQUERY**

**I- jQuery**

Est une bibliothèque développée par John RESIG. Son slogan est « Write less, do More » en français «**Ecrire peu , Faire beaucoup de choses** ».

**II- Ses Avantages**

* C’est d’écrire moins de lignes de code que javascript ;
* Tenir compte de la compatibilité entre les navigateurs car javascript pose le problème de l’interprétation au niveau de ces derniers ;
* Il possède plusieurs plug-in
* Il se base sur une syntaxe simple et facile à comprendre

**III- Les Différentes Bibliothèques existantes**

Il existe plusiers frameworks pour Javascript sur le marché : **Mootools**(mootools.net), **DojoToolkit**(www.dojotoolkit.org), **prototypeJS**(www.prototypejs.org) **YahooUI,etc..**

**IV- Les Objectifs de jQuery**

* jQuery est une bibliothèque open-source et cross-browser.
* Elle permet de traverser et manipuler très facilement l’arbre DOM des pages Web.
* Elle permet changer/ajouter des classes CSS, crée des animations, modifier des attributs et etc....

**V- Comment inclure JQuery ?**

Directement du site web de JQuery : [http ://jquery.com/download](http://www.jquery.com/download). Une fois que vous aurez télécharger jquery, ajouter le fichier « .js » au niveau de la balise <script>.

***Syntaxe***

<s**cript src=**"**jquery.js**"></script>

Ou bien par l’intermédiaire d’un **CDN** (**Content Delivery Network**). C’est un site web qui héberge des versions de JQuery.

Il en exite plusieurs chez Microsoft, chez google, et chez jquery lui même.

<**script type=**"**text/javascript**" **src=**"**http ://ajax.googleapis.com/ajax/libs/**

**jquery/1/jquery.min.js**"**>** </script>

**jquery.mim.js** : est la version Mimifiée (compressée).

Si on veut la version non compressée, on écrit

<**script type=**"**text/javascript**" **src=**"**http ://ajax.googleapis.com/ajax/libs/**

**jquery/1/jquery.js**"> </script>

**VI- La Fonction JQuery()**

JQuery repose sur une seule fonction **jQuery()** ou **$()** (en utilisant un alias). Puisque JQuery est basé sur les **sélecteurs CSS**, donc **$()** prend comme arguments **différents sélecteurs CSS**.

Donc la syntaxe de JQuery est comme suit

**$('Selecteur').action()**

**VII- Lancement de code sur Document Ready**

Quand une page web se charge , le navigateur va essayer de lancer tous les scripts qu’il rencontre. Cela pose un problème avec un script qui manipule le contenu de la page et qui se lance avant que tout le contenu de la page soit chargé.

**Par exemple**: Essayer d’obtenir la taille d’une image qui n’est pas chargée.

Pour contourner ce problème, toutes les méthodes de JQuery seront placées dans une fonction **document.ready()**.

**Exemple :**

< !DOCTYPE html>

<html>

<head>

<title> Jquery</title>

<**script type=**"**text/javascript**" **src=**"**http ://ajax.googleapis.com/ajax/libs/jque y/1/jquery.min.js**"**> </script>**

**<script type=**"**text/javascript**"**>**

**$document.ready(function(){**

**alert('Bonjour') ;}) ;**

**< !--ou la version courte $(function(){alert( 'bonsoir')}) ;-->**

**</script>**

</head>

<body>

<p>exemple comment utiliser document.ready()</p>

</body>

</html>

**VIII- Les Différents Sélecteurs de jquery**

***1-*** **Sélecteurs d’éléments**

**$(this)**  élément HTML actuel

**$** accepte comme argument des «**ID** »

**Exemple**

**$('#nomID')** retourne un élément **<=>** (**en javascript**) document.getElementByID)

**$** accepte des classes

**$(' .nomClasse')** retourne tous les éléments qui correspondent à cette classe

**$** accepte plusieurs sélecteurs**$(' .article, .nouvelles, .edito')**

**$ accept**e des sélecteurs spécifiques **$(' .radio'), $('.header'),$(' :first-child’)**

**$** accepte des sélecteurs en forme de filtres 

**$('checked’),$('odd’), $(' :visible') , $(' :contains(du texte) ')**

*2-* **Sélecteurs d’attributs**

**$** accepte des attributs**$('a[href]'), $('a[href^=http :// '),$("img[src$= '.png']")**

**9- Manipulation du DOM par JQuery**

Il est possible en Javascript et jquery de modifier le DOM. Chaque modification consiste en :

1. Sélectionner l’élément HTML qu’on souhaite modifier( par exemple afficher un div en tant que tooltip(bulle d’information) ou fenêtre modale) ;
2. appliquer sur cet élément des modifications comme :

* Changer ses propriétés par exemple :sa position, sa couleur de fond,...
* Ajouter du contenu à l’élément
* Supprimer l’élément de la page HTML
* Récupérer de l’information de l’élémenet, par exemple le texte d’un input
* Ajouter/supprimer des classes CSS, par exemple pour changer sa visibilité,

sa typographie

**9-1 Lire et modifier son texte**

La méthode **text()** permet de modifier le texte d’un élément HTML

**Exemple** : **$('#error h2').text('Pas d’erreurs trouvées') ;**

Si aucun texte n’est spécifié dans text() , on peut récupérer la valeur du texte de l’élément pour le mettre dans une variable par exemple.

La méthode **html()** permet de modifier le contenu HTML d’un élément. Si l’élément sélctionné contient déjà du **innerHTM**L, celui-ci sera remplacé par celui qu’on a spécifié en paramètr**e**

**Exemple** : **$('#errors h2')html(' <span> Pas d’erreurs trouvées</span> ') ;**

Si aucun texte en paramètre, dans ce cas on récupère la valeur de innerHTML dans une variable par exmple.

**9-2 Lire et modifier ses attributs**

La méthode **attr()** permet de lire la valeur d’un attribut de l’élément. Par exemple pour récupérer l’adresse d’une image dans la page :

**var imagePath=$('#banner img').attr('scr') ;**

Si on veut avoir la valeur d’un deuxième attribut par exemple si on veut modifier le chemin vers l’image, on aura

**var imagePath=$('#banner img').attr('scr', 'images/newimage.png') ;**

La méthode **removeAttr()** permet de supprimer un attribut, par exemple pour retirer l’attribut alt de l’image.

**$('#banner img').removeAttr('alt') ;**

**9-3 Lire et modifier son style**

***Via les classes CSS***

JQuery offre plusieurs méthodes pour manipuler l’attribut "**class**" d’un élément.

**addClass()** ajoute la classe CSS spécifiée en paramètre à l’élément sélectionné. Cette méthode n’écrase pas les classes définies précédemment mais rajoute la classe spécifiée.

**$('.btn').addClass('btn-success') ;**

**removeClass()** supprime la classe CSS spécifiée de l’élément sélectionné.

**$('.btn').removeClass('btn-success') ;**

La méthode **toggleClass()** ajoute la classe si elle n’existe pas et l’enlève si elle existe.

***Via les propriétés CSS***

On veut changer le style d’un élément sans passer par les classes CSS, c’est possible via la méthode **css()** en utilisant 3 manières différentes :

- pour trouver la valeur existante d’une propriété CSS

- pour modifier une propriété CSS d’un élément

- pour modifier plusieurs propriétés à la fois

**Exemple : modifier la propriété font-size**

**$('.btn').css('font-size', '200%') ;**

**Pour plusieurs propriétés**

**$('.btn').css("font-size" : "200%", "font-family" : "arial") ;**

**9-4 Supprimer un élément**

La méthode **remove()** permet d’effacer un ou plusieurs éléments de la page web.

Soit un div avec l’id « **popup**» qui est une fenêtre modale qu’on veut supprimer.

**Exemple**

**$('#popup').remove() ;**

La méthode **replaceWith()** permet de remplacer un élément par autre chose.

Exemple on a une img avec l’ID « prod1 » et on veut remplacer par du texte à la place car le produit a été ajouté à un panier d’achats.

**$('#prod1').replaceWith('<p>Ajouté au panier</p>') ;**

**9-5 Ajouter du contenu à la page**

Créer un nouvel élément

Avec JQuery on peut ajouter dynamiquement du contenu à la page. Il faut spécifier à l’objet JQuery le tag (avec balises) qui correspond à l’élément à créer

**Exemple**

**var newSpan=$" <span></span>") ;**

Rajouter un élément à la page

Une fois le nouvel élément crée, on utilise les méthodes suivantes pour l’ajouter à la page web :

* append() ajoute l’élément (ou le code HTML spécifié en string) en paramètre comme dernier enfant de l’élément sélectionné.
* apenndTo() ajouter l’élément sélectionné à un parent, comme premier enfant.
* prepend() ajoute l’élément (ou le code HTML spécifié en string) en paramètre comme premier enfant de l’élément sélectionné.
* prependTo() ajouter l’élément sélectionné à un parent, comme dernier enfant.
* after() ajoute l’élément (ou le code HTML spécifié en string) directement après l’élément sélectionné.
* before() ajoute l’élément (ou le code HTML spécifié en string) directement avant l’élément sélectionné.

**Exemple**

**var newSpan=$("<span></span> ") ;**

**$('#tag').append(newSpan) ;**

**9-6 Enchaînement des méthodes**

Une fonctionnalité très intéressante de JQuery est l’enchaînement de méthodes. Par exemple toutes les méthodes vues peuvent être enchaînées pour permettre de faire plusieurs choses en une seule ligne de code .

Si on veut en une seule étape faire :

- créer un élément span

- ajouter une classe « orange »

- lui ajouter du texte « JQuery »

- le rajouter à un autre élément avec l’id "tags "

**Exemple**

**$('#tags').append(**

**$(" <span></span>")**

**.addClass("orange ")**

**.text('JQuery')**

**) ;**

**9-7 Agir sur plusieurs éléments d’une sélection**

Par exemple si une page contient plusieurs chapitres et on veut générer une table de matières automatiquement. Si on ajoute un autre chapitre la table des matières se mettrait à jour.

La méthode **each()**  peut êtte chaînée à une sélection afin de boucler sur chaque élément de la sélection et leur appliquer la même action, au sein d’une fonction anonyme.

**La syntaxe**

**$('sélecteu').each(function(){**

**action1 ;**

**action2 ;**

**action3 ;**

**action4 ;**

**.....**

**})**

A l’intérieur du each(), on accède à l’élément courant afin de lire ou modifier ses propriétés.

Pour cela on utilise le mot-clè « this », ce mot-clè correspond à l’élément courant du DOM(Element).

**Exemple**

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

**// on récupère le texte**

**var chapName=$(this).text() ;**

**console.log(chapName) ;**

**})**

**9-8 Les évènements**

Les principaux évènements des éléments HTML

| **Nom** | **Action pour le déclencher** |
| --- | --- |
| click | Cliquer(appuyer puis relâcher) sur l’élément |
| dbclick | Double-cliquer sur l’élément |
| mouseover | Glisser le curseur sur l’élément |
| mouseout | Faire sortir le curseur de l’élément |
| mousedown | Appuyer (sans relâche) sur le bouton gauche de la souris sur l’élément |
| mouseup | Relâcher le bouton gauche de la souris de l’élément |
| mousemove | Faire déplacer le curseur sur l’élément |
| keydown | Appuyer (sans relâcher) sur une touche de clavier sur l’élément |
| keyup | Relâcher une touche de clavier de l’élément |
| keypress | Frapper (appuyer puis relâcher) une touche de clavier sur l’élément |
| focus | "Cibler" l’élément comme cliquez dans un champ de formulaire |
| blur | Annuler le « ciblage » de l’élément - comme sortir d’un champ d’un formulaire |
| change | Changer la valeur d’un élément spécifique aux formulaires (input, checkbox,etc..) |
| select | Sélectionner le contenu d’un champ de texte (input, textarea,etc..) |
| submit | Envoyer un formulaire (spécifique au tag <form> |

Il existe des évènements spécifiques pour l’objet **"Window**"

| **Nom** | **Action pour le déclencher** |
| --- | --- |
| resize | Redimensionner la fenêtre du navigateur |
| scroll | Scroller dans la fenêtre |
| load | La page web a fini d’être chargée (tous les objets du DOM sont chargés ainsi que les images et sous-frames) |

**Remarque**

L’évènement resize est utile si on veut faire un site responsive, c’est à dire qui s’adapte en fonction de l’écran du visiteur (pc, tablette, smartphone,watch,etc..)

**9-8-1 Associer un évènement à un élément**

Avec jQuery

**Syntaxe :**

**$('sélecteur').évènement(function(){**

**action1 ;**

**.......**

**}) ;**

**Exemple**

**// action sur le clic du bouton avec l’ID "start "**

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

**console.log("c’est un click que vous avez effcetué")}) ;**

Il existe un évènement supplémentaire en JQuery qui n’existe pas nativement en Javascript, c’est **hover()**. Il permet d’effectuer une action lorsque la souris passe sur un élément et puis une autre action lorsque la souris sort de l’élément.

Donc il faut lui donner deux codes (2 fonctions anonymes) correpondants à deux actions.

**Syntaxe**

**$('sélecteur').hover(**

**function(){**

**// Action quand la souris passe sur l’élément**

**},**

**function(){**

**// Action quand la souris part de l’élément**

**}**

**) ;**

**9-9 L’objet Event**

cet objet sert à par exemple récupérer la position de la souris ou savoir sur quelle touche l’utilisateur vient d’appuyer ?

Cet objet est accessible que lorsqu’un évènement est déclenché. Son accès se fait dans une fonction exécutée par un évènement comme suit :

**// variable e en paramètre : l’objet Event**

**$('#myButton').click(function(e) {**

**console.log(e,pageX) ;**

**console.log(e,pageY) ;**

**}) ;**

L’objet Event contiendra des propriétés différentes en fonction du type d’évènements. Si l’évènement est une action avec la souris, il contiendra pageX et pageY (les coordonnées de la souris), tandis que si c’est un keypressed il contiendra which (le code ASCII de la touche)

**10 Les effets visuels avec JQuery**

Cacher et Afficher

Les méthodes sont :

| **Méthode** | **Rôle** |
| --- | --- |
| show() | affiche un élément caché |
| hide() | cache un élément affiché |
| toggle() | si l’élément est caché, il s’affiche. Sinon s’il est affiché, il se cache |
| fadeIn() | affiche un élément caché de manière progressive |
| fadeOut() | cache un élément affiché de manière prograssive |
| fadeToggle() | si l’élémenet est caché, il s’affiche. Sinon il se cache mais de manière progressive |
| slideDown() | affiche un élément caché avec un effet de glissement |
| slideUp() | cache un élément affiché avec un effet de glissement |
| slideToggle() | si l’élément est caché, il s’affiche. Sinon il se cache avec un effet de glissement |
| fadeTo() | faire passer un élément à une ceratine opacité |

**Remarque**

Toutes ces méthodes peuvent recevoir un paramètre pour contrôler la vitesse d’apparition ou de disparition. Ses valeurs sont en string "slow", "normal"(par défaut), ou "fast". Soit en nombre de millisecondes. Mais pour **fadeTo()**, en plus du paramètre de la vitessse, il y a un pourcentage pour l’opacité

**Exemple**

**// Fait disparître lentement tous les div**

**$('div').fadeOut('slow') ;**

**$('div').fadeOut(2000) ;**

**11- La méthode d’animation avec JQuery**

La syntaxe de la méthode de jquery pour faire des animations personnalisées est comme suit :

**$(Selector).animate({params},[duration],[easing]) ;**

Elle permet d’utiliser toutes les propiétés CSS numériques comme : taille de la police, position d’un élément, opacité, largeur d’une bordure, etc...

Cette méthode prend plusieurs paramètres :

-Un **objet littéral** avec la liste des propriétés CSS qu’on désire changer et les valeurs que doivent atteindre l’animation.

Exemple On veut animer un élément en le déplaçant de 650px vers la gauche, en changeant son opacité à 50% et agrandir sa police à 24px

**{**

**// un objet littéral c’est propriété : 'valeur'**

**left : '650px' , // un objet littéral**

**opacity : '0.5',**

**fontSize : '24px'**

**}**

* la durée de l’animation en millisecondes- faculattif par défaut 400.
* l’effet de transition (easing)- facultatif. De base uniquement les effets « swing » et « linear ».

**Exemple complet**

**animate({left : '650px', opacity : '0.5',fontSize : '24px'}, 'slow') ;**

**12 Parcourir le DOM**

Il existe plsueiurs méthodes qui à partit d’un élément sélectionné, elles fournissent d’autres éléments, toujours via CSS.

| **Méthode** | **Rôle** |
| --- | --- |
| find() | permet de trouver des éléments particuliers à partir de la sélection en cours |
| children() | comme find() mais se limite aux éléments qui sont des enfants directs de la sélection en cours |
| parent() | permet de «**remonter** » et de « récupérer le parent direct de la sélection en cours |
| closest() | permet de trouver l’ancêtre le plus proche qui correspond à un sélecteur |
| siblings() | permet de récupérer tous les éléments qui sont au même niveau que la sélection en cours |
| next() | permet de trouver l’élément de même niveau que la sélection en cours et qui suit directement dans HTML |
| prev() | même chose que next() mais pour l’élément précédent |

**13- Les plug-ins**

Un plug-in est un code javascript qui est utilisé pour un but bien précis : afficher un calendrier, fournir des effest visuels, permettre une gestion des cookies plus aisée, afficher des lightbox, afficher des sliders, afficher des notifications ,etc...

**Exemple d’un plug-in est jQueryUI .**

**Remarque**

Il existe quantité de plug-ins gratuits, parmi eux :

- Effets pour les animations jQuery : http ://gsgd.co.uk/sandbox/jquery/easing

- Formater, transformer et manipuler des dates :http ://momentjs.com

- Gestion des cookies : http ://github.com/ScottHamper/Cookies

**14- Les formulaires et la validation avec jQuery**

jQuery permet de récupérer (et de modifier) facilement les valuers d’un formulaire, mais aussi de construire des intéractions et permet aussi de valider les données avant l’envoi du formulaire.

**14-1 Sélectionner des éléments du formulaire**

Grâce à l’id attribué à un élément du formulaire, on peut y accéder, mais jQuery offre des sélecteurs spécifiques aux champs de formulaire

| **Sélecteur** | **Rôle** |
| --- | --- |
| :input | Sélectionner tous les éléments de formulaire |
| :text | Sélectionner tous les champs texte |
| :password | Sélectionner tous les champs mot de passe |
| :radio | Sélectionner tous les boutons radio |
| :checkbox | Sélectionner toutes les cases à cocher |
| :submit | Sélectionner tous les boutons submit |
| :image | Sélectionner tous les boutons images |
| :reset | Sélectionner tous les boutons reset |
| button | Sélectionner tous les champs avec le type « **button** » |
| :file | Sélectionner tous les champs d’upload |
| :hidden | Sélectionner tous les champs cachés |

Ces sélecteurs peuvent être ombinés à d’autres. Par exmple sélectionner tous les champs cachés du formulaire « **monform** **»**

**var hiddenchapms=$('#monform :hidden') ;**

Jquery offre 2 sélecteurs pour permettre de récupérer les valeurs sélectionnées dans des boutons radio/cases à cocher et dans des listes déroulantes

| **Sélecteurs** | **Rôle** |
| --- | --- |
| :checked | Sélectionner les éléments cochés (radio/cases à cocher) |
| :selected | Sélectionner les éléments sélectionnés dans une liste déroulante |

**Exemple**

**var valcochée=$('input[name= "promotion"] :checked').val() ;**

**var valselect=$('#etat :selected').val() ;**

**14-2 Lire er Modifier les valeurs dans un formulaire**

Pour lire et modifier des valeurs, on utilise la méthode  **val()**. Sans paramètre on récupère la valeur et s’il y a une valeur en paramètre alors la valeur du champ sera modifiée.

**Exemple**

**var zonevide=$('#input').val() ;**

**// si la zone est vide on rajoute un texte d’introduction**

**if( !zonevide){**

**zonevide= 'faites un choix' ;**

**}**

**$('#input').val(zonevide) ;**

Pour le cas des boutons radio et cases à cocher, on utilise le filtre **is().**

**Exemple**

**var checkedbox=$('#b1').is(' :checked') ;**

Pour cocher/décocher un bouton radio, on utilise la méthode **attr()** sur l’attribut

"checked"

**Exemple**

**$('b1').attr('checked',true) ;**

**14-3 Les évènements pour les formulaires**

submit()

cet évènement est déclenché lors de l’envoi du formulaire ou en Appuyant sur « Enter » lorsqu’on tape dans un champs texte du formulaire.

focus()

Dans le cas de focus(), le focus sera mis sur l’élément sélectionné.

blur()

Cet évènement est déclenché lorsqu’on quitte le champ d’un formulaire.

click()

Cet évènement est déclenché lorsqu’on clique par exmeple sur une case à cocher.

change()

Evènement déclenché lorqu’on change la valeur d’une liste déroulante.

**14-4 Activer/Désactiver des champs**

En HTML, l’attribut "**disabled**" permet de désactiver un champ de formulaire ou un bouton dans le cas où on ne veut pas que l’utilisateur met du contenu dans le champ ou clique sur le bouton (par exemple pour éviter le réenvoi du formulaire).

En jQuery il est possible de contrôler l’activation d’un champ ou bouton en lui appliquant ou pas cet attribut

**Exemple :**

**//activer**

**$('#SendFormButton').attr('disabled',false) ;**

**//Désactiver**

**$('#SendFormButton').attr('disabled',true) ;**

**14-5 Valider un formulaire**

Valider un formulaire revient à par exemple est-ce que le format des emails a été respecté, est-ce que tous les champs obligatoires ont été remplis, etc...

En Javascript c’est compliqué mais avec jQuery, il suffit d’utiliser le plug-in

jQuery Validate(**jqueryvalidation.org**).

**La Syntaxe pour la validation**

Il faut donner une valeur à l’attribut "**name**" pour les champs du formulaire.

après on fait appel à la méthode **validate()**.

**$('#monform').validate({**

**rules :{**

**nameduchamp : {'typeDeValidation' ;}**

**},**

**messages :{**

**{typeDeValidation : 'Message d’erreurs' ;}}) ;**

**Exemple**

**$('#monform')validate({**

**rules :{**

**inputMail :{required :true,**

**email :true}},**

**messages :{**

**inputchamp :{**

**required : "Le champ est obligatoire",**

**email : "L’-email est incorrect"}**

**}}) ;**

Quelques types de validation de jQuery Validation

| **Type de Validation** | **Rôle** |
| --- | --- |
| required | Le champ est requis |
| date | La valeur du champ doit être une date |
| email | La valeur du champ doit être un email |
| url | La valeur du champ doit être un URL |
| number | La valeur du champ doit être un décimal |
| digits | La valeur du champ doit être un nombre entier |
| equalTo | La valeur du champ doit être égale à la valeur d’un autre champ |
| minlength | La longueur de la valeur du champ doit être supérieure à une certaine valeur |
| maxlength | La longueur de la valeur du champ doit être inférieure à une certaine valeur |
| min | La valeur du champ doit être supérieure à une certaine valeur |
| max | La valeur du champ doit être inférieure à une certaine valeur |