JQuery

1. Introduction

JQuery est une librairie (framework) qui permet de simplifier l'usage du langage Javascript dans une page HTML en manipulant beaucoup plus facilement le DOM grâce à un système de sélecteurs.

JQuery peut permettre de :

- Manipuler des éléments du DOM (gestion des fenêtres)
- Manipuler des éléments HTML : ajouter, modifier, supprimer des balises, des attributs, du contenu ou encore masquer
- Manipuler le CSS : ajouter, modifier, supprimer des classes, modifier les couleurs etc.
- Animer des éléments : déplacer, rotation, dégradés de couleurs, transitions CSS etc.
- Faciliter les requêtes Ajax et la manipulation de données (base de données ou fichiers)

Son fonctionnement est le suivant : on exécute du code javascript via un sélecteur sur un évènement (clic, survol, soumission de formulaire etc.). Ces évènements sont les mêmes qu'en Javascript traditionnel.

Exemple d'utilisation (fichier js externe) :

```
$(document).ready(function()
{
    $("#bouton").click(function()
    {
        alert("Vous avez cliqué sur le bouton");
    });
});
```

- `$(document).ready(function()` : initialise l'utilisation de JQuery au chargement de la page
- `$("#bouton")` : le sélecteur, qui cible l'élément passé en argument (ici une balise HTML dont l'attribut `id` est nommée bouton)
- `.click` : l'évènement, sur lequel on déclenchera le code contenu dans la fonction (ici affichage d'une boîte d'alerte). Le code Javascript de la fonction est celui que vous allez écrire : il va contenir à la fois du Javascript « normal » et du code spécifique à JQuery (fonctions etc.).

JQuery est apparu en 2006 et fut le premier outil de son genre. Il est devenu rapidement très populaire. Outre les fonctionnalités proposées, ses avantages sont sa compatibilité multi-navigateurs (en 2005, les navigateurs n'interprétaient pas tous le Javascript de la même façon), sa licence d'utilisation libre/gratuite et sa légèreté (quelques kilo-octets).

2. Intégration dans une page web

Attention, si vous utilisez Bootstrap dans votre page HTML, vous chargez déjà jQuery, mais dans une version allégée ("slim").

La solution la plus simple est l'appel via un CDN :

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></s
cript>
```

Pour obtenir la dernière version de jQuery, rendez-vous ici, choisir la version 3.x.

3. Point de démarrage

Après avoir chargé les fichiers Jquery dans une page HTML, il est nécessaire de préciser à cette page que l'on veut utiliser Jquery via l'instruction suivante à placer bien entendu dans une balise `<script>` ou un fichier JS externe :

```
$(document).ready(function()
{
   // Code à exécuter
});
```

Attention à la syntaxe.

- le sélecteur `$(document)` représente votre page web, un document (souvenez -vous du D.O.M.).
- `ready()` est un évènement qui se produit en premier lorsque le navigateur a terminé de construire le DOM à partir du code source de la page (cela ne signifie pas que la page a terminé de se charger, on parle ici uniquement du HTML, les fichiers de ressources CSS, images etc. ne sont pas forcément encore chargées en totalité).

Un seul `$(document).ready` nécessaire par script Jquery.

Ensuite, on utilisera un sélecteur associé à un évènement (`click`, `change`, `mouseover`, `blur`…) pour cibler l'élément souhaité.

4. Les sélecteurs

Les instructions JQuery commencent par la sélection d'un élément de la page web : balise HTML, attribut ou classe CSS que l'on veut manipuler.

Tout sélecteur commence par le signe $ (dollar) qui constitue un raccourci remplaçant les instruc-tions en Javascript natif du type document.getElementBy… et document.querySelector…

`$(selecteur).evenement`

Sélection d'une balise HTML

Comme en CSS, il est possible de cibler toutes les balises d'un même type. Tous les paragraphes :

```
$("p")
```

Sélection d'une classe CSS

Sélection de tous éléments de la page auxquels est appliquée la classe passée en argument (ici la classe "rouge") :

```
$(".rouge")
```

Sélection d'un élément HTML avec identifiant

Sélection d'une balise HTML nommé par un identifiant c'est-à-dire qui possède un attribut `id`. L'id est appelé par son nom précédé du signe # :

```
$("#bouton1")
```

Rappel : la valeur d'un attribut `id` doit être unique dans une page web : donc contrairement aux sélecteurs précédents ici un seul élément est pris en compte.

Sélection multiple

Jquery permet de sélectionner différents éléments en une seule fois, là où il aurait fallu autant d'appels `document.getElementBy…` que d'éléments en Javascript natif. Les éléments doivent être séparés par une virgule :

```
// Plusieurs ID
$("#panneau1, #panneau2, #panneau3")

// Plusieurs classes
$(".rouge, .gras .noir")…

// Plusieurs balises HTML
$("p, div footer")…
```

Il est aussi possible de mixer balises, identifiants ou classes :

```
$("div, #panneau1, .rouge")…
```

Sélection descendante

Le système de sélecteurs permet également d'atteindre les enfants d'un élément dans la hiérarchie D.O.M. Il existe plusieurs méthodes, en fonction des besoins.

Exemple 1 : cibler tous les enfants d'un même type connu (ici, des `<li>`)

```
// HTML
<ul id="liste">
   <li>Amiens</li>
   <li>Paris</li>
   <li>Lille</li>
</ul>

// Javascript/Jquery
// Sélectionne les enfants de l'élément #liste et met leur texte en rouge)
$("#liste li").css("color", "red");
```

Il est possible de cibler tous les enfants de types différents.

Sélection d'éléments « commençant par »

On peut sélectionner tous les éléments dont les valeurs des attributs `id` ou `class` « commencent par ». Il faut pour cela utiliser le joker `^` :

```
// HTML
<ul>
   <li id="listeA">Amiens</li>
   <li id="listeB">Paris</li>
   <li id="listeC">Lille</li>
</ul>

// JS
$("[id^='liste']")
```

Sélection d'éléments « finissant par »

A contrario, on peut vouloir sélectionner tous les éléments qui **se terminent par**. Il faut pour cela utiliser le joker `$` :

```
// HTML
<ul>
   <li id="Aliste">Amiens</li>
   <li id="Bliste">Paris</li>
   <li id="Cliste">Lille</li>
</ul>

// JS
$("[id$='liste']")
```

Sélection d'un attribut

On peut cibler un attribut, cela se fait avec la méthode `attr()` :

```
// HTML
<a id="monLien" href="index.html" title="Page d'accueil">Accueil</a>

// JS
// Accède à l'attribut href de l'id 'monLien'
// la variable 'valeur' vaudra 'index.html'
var valeur = $("#monLien").attr("href");
```

Autres sélecteurs

Il existe bien d'autres sélecteurs ou fonctions applicables aux sélecteurs, en voici la liste.

5. Manipuler l'objet sélectionné

Comme dans Javascript tout est objet, l'utilisation d'un sélecteur renvoie l'élément ciblé en tant qu'objet D.O.M. avec l'ensemble des propriétés et méthodes qui lui sont attachées. Cet objet est alors représenté par le mot-clé `this` qui en pratique sera lui-même utilisé comme sélecteur sous la forme `$(this)`.

Exemple :

```
// HTML
<div id="p1">Le développeur logiciel est un scribe informatique. Sa
mission : concevoir et tester des sites de contenu, d'e-commerce ou
applications.</div>

// JS
$("#div1").mouseover(function() {

  // Au passage de la souris, récupère le texte de l'élément #div1
  // et le stocke dans la variable a
  var a = $(this).text();

  alert(a);
});
```

6. Fonction de manipulation des objets

Manipuler les instructions CSS

La méthode `css()` permet d'appliquer ou de modifier la valeur d'une propriété CSS. Son schéma est `css(nom de la propriété CSS, valeur souhaitée)`.

Exemple :

```
// Au passage de la souris, applique la couleur rouge au texte de
l'élément #div1
$("#div1").mouseover(function()
{
    $(this).css("color", "red");
});
```

Il est possible d'appliquer plusieurs propriétés à la fois; dans ce cas la syntaxe diffère : il faut ajouter des accolades dans les parenthèses et spécifier chaque couple de propriété/valeur séparé par une virgule :

```
$("#div1").mouseover(function()
{
   $(this).css({
                "border" : "1px solid red",
                "font-weight" : "bold",
                "cursor" : "help"
               });
});
```

Enfin, la méthode `css()` permet aussi de récupérer la valeur d'une propriété CSS :

```
$("#div1").mouseover(function()
{
   var a = $(this).css("background-color);

   // Affiche, par exemple, rgba(0, 0, 255) si fond rouge
   alert(a);
});
```

Documentation officielle

Modifier la hauteur/largeur d'un élément HTML

On obtient ou change la hauteur ou la largeur d'un élément grâce respectivement aux méthodes `height()` et `width()`, qui fonctionnent de la même manière :

```
$("#div1").click(function()
{
   // Récupère la hauteur de #p1 et le stocke dans la variable h
   var h = $(this).height();

   // Si on passe une valeur, l'élément HTML est modifié en conséquence
   // Ici, la hauteur sera modifié à 500px
   $(this).height(500);
});
```

Documentation : height(), width().

Ajouter ou supprimer une classe

Les méthodes `addClass()` et `removeClass()` ajoutent et suppriment respectivement une classe CSS à un élément HTML.

Pour `addClass()`, la classe doit être définie préalablement dans la feuille de styles.

Documentation : addClass(), removeClass().

7. Ajouter/modifier du contenu

Jquery propose différentes fonctions pour ajouter, modifier ou supprimer du contenu HTML.

Récupérer ou changer un texte : text()

```
// Au clic sur l'élément #lien
$("#lien").click(function()
{
   // On récupère le texte de div1
   var letexte = $("#div1").text();

   // Changer le texte (ici, copie le texte de div1 dans div2) :
   $("#div2").text(letexte);
});
```

La fonction `text()` peut contenir du HTML.

Documentation.

Ajouter ou remplacer du code HTML

La fonction `html`

Permet de modifier le contenu d'un élément HTML (équivalent de `innerHtml` du JS classique).

```
// Modifie le contenu (innerHTML) de l'élément id="div1"
$('#div1').html("coucou<br>c'est moi");
```

Documentation.

Autres méthodes

Les différents fonctions d'insertion de contenu.

8. Gestion des formulaires

Récupérer ou changer la valeur d'un input dans un formulaire

Fonction `val()`, exemple :

```
// Lorsqu'on quitte le champ (perte du 'focus')
$("#champ").blur(function()
{
    // Récupère la valeur saisie dans le champ input #champ
    var valeur = $(this).val();
});

// Attribue la valeur au champ input #champ
$("#champ").val('Hello world');

// On peut bien sûr passer une variable
var valeur = 'Hello world';
$("#champ").val(valeur);
```

Documentation.

Vérifier un formulaire avant soumission

On rencontre ici un cas un peu particulier. La logique voudrait qu'on déclenche un évènement de type `submit`, mais là il ne se passe rien, ou plutôt si, le formulaire est bien envoyé mais aucune fonction de vérification que vous avez pris soin de coder n'est exécutée.

La bonne méthode est la suivante : lancer l'action de vérification lorsque le bouton d'envoi du formulaire est cliqué, et exécuter vos fonctions de validation sur cet évènement.

Exemple

```
// HTML
<form action="#" method="post">
    <input type="text" name="prenom" id="prenom">
    <input type="button" name="btn_envoyer" id="btn_envoyer"
value="Envoyer">
</form>
```

```
// JQuery
function verif()
{
     var envoi = true;

     // Récupère la valeur saisie dans le champ input #prenom
     var leprenom = $("#prenom").val();

     if (leprenom == "")
     {
         envoi = false;
         alert("Le prénom doit être renseigné");
     }

     // +++ Ici contrôles pour d'autres champs +++

     // Si envoi est toujours à true, on peut soumettre le formulaire
     if (envoi == true)
     {
         document.forms[0].submit();
     }
     else
     {
         return false;
     }
}

$("#btn_envoyer").click(function(e)
{
    /* On doit bloquer l'èvènement par défaut avec l'instruction
    * ci-dessous
    * 'e' est un objet nommé librement représentant l'évènement
    */
    e.preventDefault();

    // Appel de la fonction verif()
    verif();
});
```

9. Effets

Afficher, cacher ou inverser un élément

fonctions `show()`, `hide()`, `toggle()` :

```
// Affiche (fait apparaître) l'élément div1
$('#div1').show();

// Cache l'élément div1
$('#div1').hide();

/* Inverse l'état actuel de visibilité de div1 :
* Si visible, toggle() applique hide(),
* Si caché, toggle() applique show()
*/
$('#div1').toggle();
```

Autres fonctions d'effet

- Afficher, cacher ou inverser un élément avec l'effet **fade** : fadeIn(), fadeOut(), fadeIn().

- Afficher, cacher ou inverser un élément avec l'effet _slide__ : slideDown(), slideUp(), slideToggle().
- Animer un élément : animate().

10. Jquery avancé

JQuery propose des possibilités très avancées via ses nombreuses fonctions et notamment la possibilité d'écrire des plugins (extensions) dont certains sont disponibles sur le net.

Il existe aussi des "dérivés" de JQuery :

- JQueryUI (pour **User Interface**) qui augmente les possibilités de gestion des interfaces.
- JQuery Mobile pour le développement d'applications mobiles.

11. Exercice

Reprenez l'exercice de validation du formulaire "sondage stagiaires" pour le vérifier avec JQuery.

Les messages d'erreur doivent s'afficher sous le champ `input` correspondant. Utilisez éventuellement les classes d'alerte de Bootstrap.