Atelier N°3	*La bibliothéque Jquery*
Objectifs	• **Les bases de jQuery** • **jQuery et le document DOM** • **Traitement des évènements en jQuery** • **Animer des éléments** • **Application**

1 Les bases de jQuery

jQuery, une bibliothèque JS gratuite, dont la syntaxe est très courte, dont les noms des fonctions sont intuitifs (en anglais bien sûr), permettant de **faire des animations**
jQuery est tout simplement **un fichier JavaScript**. Il vous suffit donc de le télécharger sur le site officiel.
Pour vous montrer la simplicité de jQuery, voici le classique « Hello World » : créer le fichier html suivant et mettre votre bibliothèque jQuery « jquery.js » dans le même répertoire que ce fichier.
jQuery s'intègre comme tout autre fichier JavaScript :

Exemple 2.1

```
<!DOCTYPE html>
<head>
<title>Hello World avec jQuery</title>
<meta charset=iso-8859-1" />
<script " src="jquery.js"></script>
</head>
  <body>
    Salut tout le monde !
    <script >$('body').html('Hello World');</script>
  </body>

</html>
```

Google héberge lui aussi les fichiers jQuery : http://ajax.googleapis.com/ajax/libs/j [...] jquery.min.js pour la version de Production (compressée).

On peut donc facilement intégrer jQuery à sa page en mettant dans le **<head>** :

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
```

Vous pouvez ensuite tester et modifier le code en ligne en visitant ces deux sites: JSBin et jsFiddle

1.1 Fonction principale

Toute jQuery repose autour d'une fonction : jQuery() , abrégée **$()** qui permettra **de sélectionner des éléments dans votre page web**.

1.2 Sélecteurs basiques

Voici quelques exemples pour que vous vous rappeliez ce que sont les sélecteurs CSS :

h1 : les balises h1
a,h1,h2 : les balises a, h1 et h2
#titre : la balise ayant pour id "titre"
.nav : les balises qui ont la classe "nav"
*** :** toutes les balises
elem[attr] : Balises elem dont l'attribut "attr" est spécifié.
elem[attr="val"] : Balises elem dont l'attribut "attr" est à la valeur val…

1.3 Sélecteurs spécifiques à jQuery

:hidden Éléments invisibles, cachés.
:visible Éléments visibles.
:parent Éléments qui ont des éléments enfants.
:header Balises de titres : h1, h2, h3, h4, h5 et h6.
:not(s) Éléments qui ne sont pas sélectionnés par le sélecteur s.
:has(s) Éléments qui contiennent des éléments sélectionnés par le sélecteur s.
:contains(t) Éléments qui contiennent du texte t.
:empty Éléments dont le contenu est vide.
:eq(n) et **:nth(n)** Le n-ième élément, en partant de zéro.
:gt(n) (greater than, signifiant plus grand que) Éléments dont le numéro (on dit l'« index ») est plus grand que n.
:lt(n) (less than, signifiant plus petit que) Éléments dont l'index est plus petit que n.
:first Le premier élément (équivaut à :eq(0)).
:last Le dernier élément.
:even (pair) Éléments dont l'index est pair.
:odd (impair) Éléments dont l'index est impair.

Passer à la pratique

Exemple 2.3

```
<!DOCTYPE html>
<head>
<title>Hello World avec jQuery</title>
<meta charset=iso-8859-1" />
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
<script " src="jquery.js"></script>
</head>
  <body>
    <div id="titre">J'aime les frites.</div>
    <script > $('#titre');
    // Sélectionne notre balise mais ne fait rien.
     alert($('#titre').html()); // la méthode html() permet d'accéder au contenu des éléments elle affiche
alors "J'aime les frites."
     $('#titre').html('Je mange une pomme');
   // Remplace le contenu ("J'aime les frites.") par "Je  mange une pomme".
// Deux autres méthodes, before() et after() permettent d'ajouter du contenu ou un élément de // la page
// Ajoute du contenu après chaque balise textarea.
$('textarea').after('<p>Veuillez ne pas poster de commentaires injurieux.</p>');
// Ajoute "Voici le titre :" avant la balise ayant comme id "titre".
$('#titre').before('Voici le titre :');
// Ajoute "! Wahou !" après la balise ayant comme id "titre".
$('#titre').after('! Wahou !');
// Il faut savoir qu'on peut chaîner les méthodes
$('#titre') .before( ' avant: ')
            .after('! après !');
</script >
 </body>

</html>
```

1.4 jQuery et le document DOM

La page web **contient des éléments** (des balises), qui elles-mêmes sont susceptibles de contenir d'autres éléments.
Cette **organisation** de la page web forme un « arbre » où un élément parent **contient des fils**, qui eux-mêmes sont susceptibles d'être parents.
Le **DOM** est un ensemble d'interfaces standardisées, décrivant cet « arbre ». Concrètement, on les utilise en JavaScript grâce à l'objet document, qui grâce à ses nombreuses méthodes (getElementById(), createElement()) nous permettent de récupérer des éléments, et ainsi les modifier grâce à leurs propriétés et à leurs méthodes (setAttribute(), innerHTML, addEventListener, etc...).

1.4.1 Chargement du DOM

Quand on fait un appel à la fonction principale, il se peut parfois qu'elle ne retourne rien. On a beau placer son code en fin de body, **les éléments de la page web ne sont pas encore placés**.
La solution de ce problème en JavaScript est le fameux :
Code : JavaScript

```
window.onload = function(){
        // Fonctions du genre document.getElementById('balise') qui marchent,
        // on peut accéder aux éléments.
        };
```

jQuery offre une syntaxe à peu près similaire : la fonction doit juste **être donnée à la fonction principale** jQuery() ou $().
Code : JavaScript

```
$(function(){
        // On peut accéder aux éléments.
        // $('#balise') marche.
        });
```

1.4.2 Créer des éléments en utilisant le DOM...

Au lieu de donner un sélecteur à la fonction principale, on peut aussi **créer des éléments** : la chaîne de caractères est alors le **nom de la balise comme si elle était simple** (c'est-à-dire le nom de la balise entourée de chevrons ainsi que le slash utilisé afin de fermer les balises avant le chevron droit).
Afin de créer un élément, on utilisait document.createElement('balise') ; c'est donc $('<balise/>') avec jQuery. Par exemple $('<h1 />') crée un titre de niveau 1.

1.5 Traitement des évènements en jQuery

Un événement en JS est une action de l'utilisateur (mouvements de la souris, clic, etc...) ou même une action interne au navigateur (chargement de la page par exemple).
jQuery dispose de méthodes simples pour **attacher des événements à des fonctions** (ou « écouter un événement »). Nous allons donc passer en revues ces méthodes, avec lesquelles vous pourrez enfin dynamiser votre page !
Les événements avec jQuery seront créés grâce à des **méthodes** ayant pour nom le type de l'événement, l'argument étant la **fonction de retour**.

Code : JavaScript

```
// Écoute d'un événement
elements_jQuery.evenement(function(){
// Action
});
```

Toutes les méthodes que nous allons voir ici peuvent aussi être appelées sans argument : elles servent alors à **déclencher l'action par défaut du navigateur** ainsi que celle que **vous avez définie** (grâce aux méthodes avec fonction de retour).
Pour faire simple :

```
Code : JavaScript
elements_jQuery.evenement(function(){
// Ce qu'il faut faire
alert('Action !');
});
// Action !
elements_jQuery.evenement();
```

1.5.1 Evénement relatif aux Formulaires

a. *Sélection*

select est déclenché lorsque du texte est sélectionné dans un <input type="text" /> ainsi que dans un <textarea></textarea>.

```
//Code : JavaScript
$(':text,textarea').select(function(){
alert($(this).val());
});
```

b. *Changement*

change est déclenché lorsque le texte d'un <input type="text" />,
d'un <input type="password" />, d'un <textarea></textarea> ainsi que le choix d'un <select></select> est **changé** (si après édition il est le même qu'avant édition, l'événement ne sera pas déclenché), ainsi que quand un <input type="checkbox" /> ou un <input type="radio" /> est cliqué.

```
//Code : JavaScript
$(':input').change(function(){
alert($(this).val());
});
//Code : JavaScript
// Déclenche l'action du navigateur par défaut
// ainsi que l'action que vous avez définie.
$(':input').change();
```

c. *Soumission du formulaire*

submit est déclenché lorsqu'un formulaire est soumis (soit par l'intermédiaire d'un <input type="submit" /> mais aussi grâce au JavaScript). La fonction passée en paramètre peut renvoyer *false*, et alors le formulaire n'est pas soumis.
Cette méthode s'applique donc aux balises <form></form>.

```
//Code : JavaScript
$('form[name="inscription"]').submit(function(){
if($('form[name="inscription"] :password :first').val().length <6){
alert('Veuillez rentrer au moins 6 caractères dans votre mot de passe');
return false;
}
});
//Code : JavaScript
// Soumet tous les formulaires
// et déclenche l'action du navigateur par défaut
// ainsi que l'action que vous avez définie.
$('form').submit();
```

Le tableau suivant dresse la liste des événements liés aux actions effectuées dans un formulaire. En HTML, vous utiliserez les attributs correspondants. En jQuery, vous appliquerez les événements à l'élément concerné.

Événement	Attribut HTML	Exécution du script
`Blur`	`onblur`	Lorsqu'un élément perd le focus
`Change`	`onchange`	Lorsque la valeur/le contenu d'un élément change
`Contextmenu`	`oncontextmenu`	Lorsqu'un menu contextuel est déroulé
`Focus`	`onfocus`	Lorsqu'un élément reçoit le focus
`Formchange`	`onformchange`	Lorsque le contenu du formulaire change
`Forminput`	`onforminput`	Lorsque l'utilisateur entre des données dans le formulaire
`Input`	`oninput`	Lorsqu'un élément reçoit des données entrées par l'utilisateur
`Invalid`	`oninvalid`	Lorsqu'un élément n'est pas valide
`Select`	`onselect`	Lorsqu'un élément est sélectionné
`Submit`	`onsubmit`	Lorsque le formulaire est soumis (généralement au clic sur le bouton Submit)

1.5.2 Utiliser les effets

Les effets de jQuery sont des **animations prédéfinies simples d'utilisation** mais peu configurables qui suffisent parfois à réaliser ce que l'on cherche.

a. Visibilité

show() permet d'**afficher** les éléments en question.
hide() permet de **cacher** les éléments en question.
toggle() permet de jongler entre la présence ou l'absence de l'élément (**si l'élément est caché, l'afficher, sinon le cacher**).

Code : JavaScript
```
// Utilisé pour afficher / cacher une balise secret.
$('blockquote.secret').toggle('normal');
```

Ces trois méthodes peuvent ne prendre aucun argument (à ce moment là, il n'y a pas d'animation et tout se fait **de façon brute**) ou les arguments classiques la **durée de l'animation** et la **fonction de retour**: à ce moment là, la hauteur, la largeur et l'opacité changent **de manière progressive**.
On peut utiliser toggle() avec les arguments classiques ou avec un seul argument, un booléen qui, à *true* affiche l'élément avec show(), mais qui à *false* cache l'élément avec hide().

b. Glissement

slideDown() permet de **dérouler** verticalement les éléments en question.
slideUp() permet d'**enrouler** verticalement les éléments en question.
slideToggle() permet de jongler entre la présence ou l'absence de l'élément (**si l'élément est caché, le dérouler, sinon l'enrouler**).

c. Disparition

fadeIn() permet de faire **apparaître** les éléments en question en modifiant l'opacité de manière progressive.
fadeOut() permet de faire **disparaître** les éléments en question en modifiant l'opacité de manière progressive.

fadeTo() s'utilise avec les arguments classiques, excepté que la fonction de retour est en troisième position et en seconde position se trouve un nombre compris entre 0 et 1, qui est l'**opacité** à laquelle les éléments doivent être, toujours de manière progressive.

Code : JavaScript
```
$('div').fadeIn(2000);
$('a').fadeTo('slow',0.5);
```

1.6 Application

Créons un petit jeu de ninja ! :)
Lorsque l'utilisateur clique sur le bouton "Play", le jeu génère 3 ninjas qui sont prépositionnés dans un tableau 3 lignes 3 colonnes dans une balise « section#container » puis sont cachés.
Ensuite l'utilisateur devra cliquer sur les ninjas cachés afin de les trouver.
Lorsqu'il clique sur un ninja l'image apparaît et le score du joueur est mis à jour (compteur de ninjas restant).
La partie se termine lorsque le joueur a trouvé tous les ninjas.