SUPPORT DE COURS JQUERY

Sommaire

Auteur : Florent Lhélias.

QU'EST CE QUE LE JAVASCRIPT ?

Javascript est un langage de programmation qui permet d'animer et de dynamiser des pages Web. A ne surtout pas confondre avec le Java qui est un autre langage informatique.
Javascript est surtout utilisé côté client, c'est-à-dire que c'est le navigateur qui lit et interprète le code que l'on écrit en Javascript (voir l'architecture client / serveur pour plus de détails). Même si Javascript est connu pour le Web, il peut être utilisé pour créer d'autres applications. Tom Tom en est un exemple.

HISTORIQUE

1995 est l'année qui marque le début de l'histoire du Javascript.
Il y a 20 ans, le leader sur le monde du Web était la société Netscape Communications avec son navigateur Web Netscape Navigator qui était utilisé par plus de 90% des internautes. Le tout premier navigateur Web s'appelait Mosaïc et vient aussi de la société Netscape.
A l'époque, Netscape est partenaire d'une société informatique du nom de Sun Microsystems, un fabricant d'ordinateurs et éditeur de logiciels dont la réputation n'est plus à faire auprès des professionnels.

Sun développa un langage informatique performant appelé JAVA. Encore très utilisé aujourd'hui par de nombreux systèmes, ce langage est destiné exclusivement à la conception de logiciels. Ce qui le rendait trop puissant et compliqué pour une utilisation au sein d'applications Web.
Ayant flairé ce nouvel eldorado virtuel, Sun et Netscape décidèrent de créer un langage exclusivement réservé au Web. Un langage moins puissant que le JAVA mais beaucoup plus simple d'utilisation, ce qui le rendrait très accessible à qui veut l'apprendre.
Cette mission fût confiée à Brendan EICH, génie technique qui rejoint l'équipe de Netscape en avril 1995.


Brendan EICH

Il avait 10 jours pour créer un langage qui s'adapterait à la nouvelle version du navigateur Netscape. Ce langage fût baptisé Livescript.
Le Livescript était un langage révolutionnaire qui donna naissance aux premières pages web dynamiques.

Par soucis marketing et voulant profiter de la notoriété de son partenaire Sun autour du langage Java, Netscape décide de renommer Livescript en Javascript durant la même année. Ainsi les clients feraient le lien entre Java de Sun et Javascript.

Netscape soumet Javascript à ECMA International (European association for standardizing information and communication systems) pour une standardisation de son langage. ECMA est une organisation active de standardisation dans le monde informatique, similaire au W3C pour le HTML.

PRESENTATION DE JQUERY

JQuery est ce qu'on appel une bibliothèque ou une API (application programming interface) Javascript, c'est à dire un ensemble de fonctions qui permettent d'écrire de manière condensée, les lignes de codes les plus utilisées en Javascript.
Exemple : nous allons souvent utiliser les fonctions Javascript qui permettent d'attraper les balises HTML et les animer, ainsi en Javascript nous allons écrire d'une certaine manière et en JQuery d'une manière plus condensée.

Javascript : *document.getElementsByTagName("LI")[0].innerHTML="Lait";*
JQuery : *$("LI:first").text("Lait");*

Les deux codes font donc la même chose mais ne s'écrivent pas de la même façon.


John RESIG

La version 1.0 stable de JQuery a été crée en aout 2006 par un jeune homme de 22 ans du nom de John RESIG. Aujourd'hui nous en sommes à la 1.11.

jQuery est une bibliothèque JavaScript créée par John Resig en 2006 afin de faciliter la lecture et l'écriture du JavaScript. Une vingtaine de lignes de code en JavaScript correspond à quelques lignes seulement en jQuery. D'où le slogan : "Write less… Do more".

La syntaxe de jQuery est conçue pour rendre la navigation d'un document plus facile en sélectionnant les éléments du DOM (Document Object Mode = il s'agit de l'arborescence de tous les éléments d'une page web), la manipulation des feuilles de style css par en ajoutant et/ou supprimant des attributs, créer des animations facilement que étaient parfois complexes en JavaScript pur, gérer des événements (au lieu d'utiliser des attributs HTML pour appeler des fonctions JavaScript pour la gestion d'évènements, jQuery peut être utilisé pour gérer les événements purement en JavaScript. Ainsi les balises HTML et JavaScript peuvent être séparés), récupérer des informations à partir d'un

serveur sans avoir à réactualiser la page (= AJAX), intégrer des fonctionnalités très utilisés en JavaScript comme par exemple les fadeOut() ou les fadeIn()… A ce jour, beaucoup de fonctionnalités sont proposées et facilement paramétrables sans connaissance préalable de JavaScript. L'intérêt du jQuery en plus d'être relativement simple, repose sur de nombreuses fonctionnalités. Beaucoup de plugins (ou de modules d'extension) sont développés chaque jour par de nombreux utilisateurs qui contribuent à faire de jQuery l'une des bibliothèques les plus utilisées actuellement. En utilisant jQuery, le développeur réalise des plugins avec des effets avancés, des widgets thématiques… L'approche modulaire de jQuery permet la création de puissantes pages web dynamiques et d'applications web.

INTÉGRATION DE JQUERY

La bibliothèque jQuery est un fichier JavaScript unique contenant tous ses DOM, événements, effets et fonctions qui peuvent donc être utilisés. On inclut ce fichier dans une page web par l'intermédiaire d'un lien vers une copie locale ou alors grâce à l'une des nombreuses copies disponibles à partir de serveurs publics.

Copie locale :

La première méthode consiste tout simplement à créer un dossier .js dans lequel on stocke l'ensemble des fichiers JavaScript (ceci n'est pas une obligation, juste une question d'organisation…). On y retrouvera donc notre document *jquery.js* téléchargeable depuis le site officiel de jQuery : http://jquery.com/. On nous propose généralement deux versions : l'une compressée et l'autre indentée. La première version aura une vitesse d'exécution légèrement plus rapide que la seconde. Mais la seconde sera tout simplement plus facile à lire. Puis, placer l'appel à la bibliothèque jQuery dans le <head> du document entre des balises <script></script> :

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

Serveur public (Google) :

La seconde méthode consiste à faire appel aux outils proposés par Google ou autre… Sans avoir à télécharger la bibliothèque jQuery, en vous référant aux liens proposés par Google et en les incluant, vous aurez accès aux mêmes fonctionnalités :
https://developers.google.com/speed/libraries/devguide#jquery

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"
type="text/javascript"></script>
```

Y-A-T-IL UN EMPLACEMENT PARTICULIER POUR L'APPEL À JQUERY ?

Lorsque l'on se rend sur une page web, celle-ci s'affiche (les balises, le texte, les images…) puis vient le JavaScript. C'est pourquoi, il n'est pas rare de voir un code JavaScript directement placé en bas du document html, ce qui est plutôt logique. Néanmoins, avec jQuery la question ne se pose plus. En effet, sa fonction initiale indique au navigateur : « *lorsque les éléments qui constituent mon document sont chargés…* ». On utilise pour cela l'événement *.ready()* : http://learn.jquery.com/using-jquery-core/document-ready/

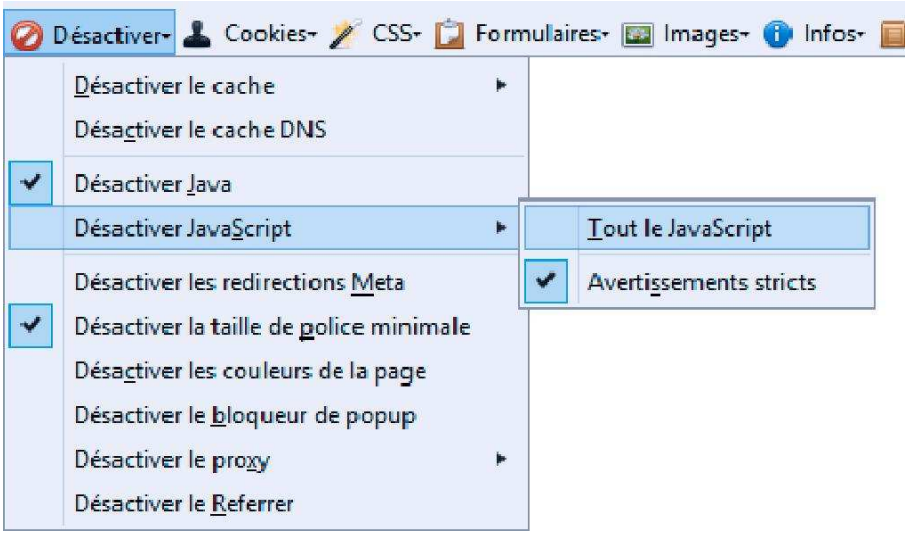
```
<script type="text/javascript" >
$(document).ready(function(){ //lorsque mon document est prêt
      //j'intègrerai du code ici
});
</script>
```

Dans son ensemble, voici ce que cela donnera :

```
<!DOCTYPE
html> <html>
   <head>
      <title>jQuery</title>
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"
type="text/javascript"></script>
      <script type="text/javascript">
            $(document).ready(function(){
                  alert("c'est parti !!!");
            });
      </script>
   </head>
   <body>
   <section>Ici le contenu de votre
   site</section> </body>
</html>
```

COMMENT SAVOIR SI JQUERY A BIEN ÉTÉ INITIALISÉ DANS NOTRE DOCUMENT ?

Pour cela, il existe plusieurs méthodes. La première consiste tout simplement à s'assurer que JavaScript est activé. Bien qu'il le soit sur l'ensemble des navigateurs, nul n'est à l'abri d'un paramétrage hasardeux… Pour s'en assurer, il suffit d'accéder aux options d'affichage du navigateur. Sur Firefox, pour y accéder directement, télécharger *Web Developper* qui permettra d'accéder au panel suivant :



Sinon, utilisez dans le document html le balise **<noscript></noscript>**. Cette balise activera son contenu **seulement si** le JavaScript a été désactivé.

```
<!DOCTYPE
html> <html>
   <head>
      <title>jQuery</title>
        <script src="js/jquery.js"
   type="text/javascript"></script> </head>
   <body>
   <section>Ici le contenu de votre
   site</section> <noscript>Absence de
   JavaScript</noscript> </body>
</html>
```

La seconde méthode consiste à se reporter à la console directement accessible par l'intermédiaire de votre navigateur (F12 ou Firebug). Grâce à une fonction (*console.log()*) inclue dans notre fonction

jQuery, nous allons interroger notre navigateur. Si le message s'affiche, la bibliothèque est bien prise en compte :

```
<script type="text/javascript" >
$(document).ready(function(){
      console.log("jQuery actif !!");
});
</script>
```

COMMENT ÉCRIT-ON EN JQUERY ?

Depuis que jQuery existe, l'écriture a été rendue plus facile. En voici trois exemples. Ici on charge la bibliothèque jQuery sur l'ensemble de la page web.

```
<script>
jQuery(document).ready(function(){
    //ici j'intègrerai du code jQuery
});
</script>
```

Puis, cette écriture est devenue :

```
<script>
jQuery(function(){
      //ici j'intègrerai du code jQuery
});
</script>
```

Qui par la suite a encore été réduite à la suivante :

```
<script>
jQuery(function(){
      //ici j'intègrerai du code jQuery
});
</script>
```

Et pour réduire encore un peu plus le code, on peut remplacer jQuery par le sélecteur **$** :

```
<script>
$(function(){
      //ici j'intègrerai du code jQuery
});
</script>
```

A cette étape, il convient de faire une petite précision concernant les conflits entre bibliothèques. Car jQuery n'est pas la seule bibliothèque utilisée. On peut de temps en temps être obligé d'utiliser deux, voire plusieurs bibliothèques sur un même document. Dans ce cas, pour éviter tout conflit, l'introduction à jQuery diffère quelque peu des précédents exemples :

```
<script>
(function($){ //on place le sélecteur $ en paramètre de ma fonction
      //ici j'intègrerai mon code jQuery
})(jQuery);
</script>
```

```
<script>
jQuery(document).ready(function($){
      //ici j'intègrerai mon code jQuery
});
</script>
```

Voici un site répertoriant l'ensemble des bibliothèques JavaScript :

http://www.commentcamarche.net/faq/10424-librairies-javascript-prototype-scriptaculous-yahoo-ui-etc

Les auteurs de l'article parlent de 'librairies' or il s'agit bien de bibliothèques. Pour info, Library en anglais signifie Bibliothèque en français. Et Librairie en français se traduit par Bookstore en anglais. En vous référant à la documentation officielle de jQuery, ils parlent bien de Library, donc de Bibliothèque ;)

Mais cela n'enlève en rien la qualité de recherche de l'article que je vous recommande de consulter.

COMMENT FONCTIONNE JQUERY ?

jQuery reprend la méthode objet. C'est-à-dire qu'à partir d'un élément du DOM (*Document Object Model*) ayant ses propres caractéristiques, jQuery va modifier l'objet original, de référence (cf. l'élément ciblé). Prenons l'exemple de la maison témoin. Admettons que vous souhaitez acheter une maison et que vous vous adresser à une société immobilière qui vous propose de visiter une maison témoin. A partir de cette maison témoin, la société vous en proposera une copie conforme. Votre future maison tirera donc ses plans de la maison témoin que vous avez visitée. Votre maison aura donc par exemple : une salle de bain, deux chambres, un séjour, une cuisine, un wc et un garage. A partir de ce plan de base, la société immobilière vous autorise à retoucher un certain nombre de ces éléments 'par défaut' afin de créer un habitat personnalisé. On pourrait par exemple changer la salle de bain en lui assignant une couleur bleue et retoucher les dimensions du garage. Ce qui en jQuery donnerait :

```
jQuery(`ma_maison').salle_de_bain(`couleur','bleue').garage(`dimensions','20m_par_30m');
```

Revenons au jQuery et remplaçons à présent la maison par un élément du DOM par exemple un <h1> et nous allons lui modifier sa couleur (donc son css) :

```
jQuery(`h1').css(`color', `red');
```

A cet objet, nous allons pouvoir lui indiquer des instructions supplémentaires, lui attribuer des effets par l'intermédiaire de méthodes etc.

LES SÉLECTEURS

Un sélecteur permet de sélectionner un ou plusieurs éléments sur la page. C'est la base même du JavaScript. A partir d'un élément, on l'associe à un événement, une méthode, un effet afin d'obtenir quelque chose…

Voici comment cela fonctionne à travers quelques exemples rapides :

```
$('p');              // Je sélectionne tous les paragraphes
$('p:first');        // Je sélectionne le premier des paragraphes trouvé
$('p:last');         // Je sélectionne le dernier des paragraphes trouvé
$('p[id]');          // je sélectionne le paragraphe avec l'identifiant id
$('p:eq(1)');        // je sélectionne le deuxième paragraphe trouvé
$('p:not(.rouge)');// je sélectionne tous les paragraphes sauf ceux ayant
.rouge pour classe
$('.maClasse');   // Je sélectionne les éléments ayant .maClasse pour
classe
$('#monId');         // Je sélectionne les éléments possédant #monId pour Id
$('p .lien');        // je sélectionne tous les éléments ayant la classe
.lien, et descendants directement du paragraphe
$('p > .lien');   // je sélectionne seulement les éléments ayant la classe
.lien et descendants directement du paragraphe
$('.lien + .visite'); //je sélectionne les éléments ayant pour classe
.visite précédés immédiatement d'un élément ayant pour classe .lien
$('.lien ~ .visite'); // je sélectionne les éléments ayant pour classe
.visite et précédés ou pas d'un élément ayant pour classe .lien
$('input[name=pseudo]');      //je sélectionne l'élément du formulaire
ayant "pseudo" pour nom
$('input[name!=pseudo]');     // je sélectionne les éléments n'ayant pas
"pseudo" pour nom
$('input[type=button]');      // je sélectionne un input de type 'button'
$('input:button');            // je sélectionne tous les input de type
'button'
$('input:checked');           // je sélectionne tous les input 'checked'
$('input:not(.bleu,.vert)');    // je sélectionne tous les éléments du
formulaire sauf ceux ayant .bleu, et/ou .vert pour classe
$(':animated');         // je sélectionne tous les éléments animés
$(':contains(frites)'); // je sélectionne tous les éléments contenant le
mot frites

$(':empty');      // Je sélectionne tous les éléments ayant ou des éléments
n'ayant aucun nœuds enfants ou texte
```

Pour résumer, on cible un élément grâce à :

- **sa balise :** $(“h4”)
- **son #id :** $(“#mon_id”)
- **sa .class :** $(“.ma_class”)
- **par lui-même :** $(this)
- **par le type :** $(“input:text”)
- **tous les types :** $(“:radio”)
- **par l’attribut :** $(“img[*alt*]”)
- **tous les attributs :** $(“[*alt*]”)
- **par le nom de l’attribut ET sa valeur :** $(“img[title=’chaton’]”)
- **sélectionner plusieurs sélecteurs en même temps :** $(“h1, p, #mon_id”)

Retrouver l’ensemble des sélecteurs : http://jquery.developpeur-

web2.com/documentation/selecteurs.php

LES ÉVÉNEMENTS

Très utilisés, les évènements se déclenchent quand le visiteur effectue quelque chose qui correspond au script. Par exemple, un évènement se déclenchera quand le visiteur cliquera sur tel élément (donc un sélecteur) ou lorsque le visiteur appuiera sur telle touche de son clavier cela fera apparaître un élément, etc. le principe d'un évènement est de lancer un bout de code quand une action est remplie. Voici une liste non-exhaustive de déclencheurs d'évènements directement amorcés par le visiteur :

```
.ready()       //au chargement de l'élément
.click()       //simple-clic
.dblclick()    //double-clic
.hover()       //passage de la souris
.mouseenter()  // rentrer dans un élément
.mouseleave()  // sortir d'un élément
.mousedown()   //clic souris enfoncé
.mouseup()     //clic souris relevé
.scroll()      //scroll de la souris
.heydown()     //appui clavier
.keyup()       //relâche d'une touche
.keypress()    //touche maintenue enfoncée
```

Pour utiliser un déclencheur d'évènement, il faut cibler un élément déclencheur.

```
$('element_cible').evenement(function(){
    //qu'est-ce-qui se passe quand... ?
});
```

```
$('h1').click(function(){
    //quand je clique sur un <h1>...
    alert("j'ouvre une boite de dialogue");
});
```

Cela peut être une balise, une image… Dans l'exemple ci-dessous, je cible tous mes <h1>. Lorsque je clique sur un <h1> de mon document, j'affiche une boite de dialogue *alert(")* :

```
<script type="text/javascript">
	$(document).ready(function(){
		$('h1').click(function(){	//je cible mon élément h1. Lorsque que le visiteur cliquera dessus...
			alert('Vous venez de cliquer sur la balise h1');
	//...une boite de dialogue s'affichera avec le texte mentionné
		});
	});
</script>
```

Le principe reste exactement le même pour les autres évènements ; Ici je place deux évènements à l'intérieur de ma fonction *.ready()* :

```
<script type="text/javascript">
	$(document).ready(function(){
		$('header').mouseenter(function(){ alert('Vous êtes dans
			le <header>');
		}); $('header').mouseleave(function(){
			alert('Vous êtes sorti du <header>');
		});
	});
</script>
```

On peut également passer un paramètre dans la fonction. Ce paramètre représente la cible ainsi que ses propriétés. L'exemple ci-dessous montre comment désactiver la propriété native de la balise <a> qui consiste à rediriger le visiteur vers un emplacement particulier, télécharger un document… Ici on désactive donc le comportement par défaut de l'élément ciblé ('a') par l'intermédiaire de la propriété JavaScript *preventDefault()* :

```
<script type="text/javascript">
	$(document).ready(function(){
		$('a').click(function(event){
			event.preventDefault();
		});
	});
</script>
```

Tous les évènements ne sont pas à connaitre. Vous pouvez vous rendre sur la documentation officielle jQuery afin de consulter les différents évènements proposés : http://api.jquery.com/category/events/

LES MÉTHODES

A l'instar des évènements qui sont déclenchés par le visiteur, les méthodes n'ont pas cette utilité. Par l'intermédiaire des méthodes, on va effectuer une action précise sur un ou plusieurs éléments du DOM. L'écriture reste sensiblement la même. On peut bien évidemment assigner plusieurs méthodes à un seul élément cible :

```
<script>
      jQuery('elementCible').methode1().methode2().methode3();
</script>
```

Il existe plusieurs méthodes. En voici une liste non-exhaustive :

```
.css() //permet de modifier la css de l'élément cible
.html() //permet de lire/écrire/récupérer le contenu d'un élément en respectant le balisage html
.text() //permet de lire/écrire/récupérer le contenu d'un élément sans respecter le balisage html
.offset() // définit la position de l'élément par rapport au point d'origine du document
.position() //définit la position de l'élément par rapport au point d'origine du parent
.height() //définit la hauteur de l'élément
.width() //définit la largeur de l'élément
.attr() // permet d'affecter des attribut à l'élément (#id, .class,…)
.addClass() //rajoute une .class à l'élément
.removeClass() //supprime une .class à l'élément
.hasClass() //permet de savoir si l'élément possède telle .class en renvoyant un booléen
.after() //insère du contenu après l'élément
.before() //insère du contenu avant l'élément
.replaceWith() //remplace le contenu de l'élément par autre un autre contenu
.val() //récupère la valeur de l'élément
```

JQuery répertorie l'ensemble des méthodes proposées par la bibliothèque :
http://api.jquery.com/category/manipulation/

Pour certaines méthodes comme par exemple *.attr()* ou *.css()*, vous pouvez indiquer une information unique à votre élément cible :

```
$('p').css('color','blue');
```

Mais vous pouvez aussi attribuer plusieurs paramètres de la manière suivante. Attention à la syntaxe et l'emploi ici des accolades, des : et l'absence de la virgule lors de la dernière propriété css apportée

```
$('p').css({
            'color' : 'blue',
            'font-style' : '20px',
            'font-weight' : 'bold'
   });
```

LES EFFETS

Vous pouvez apporter des effets particuliers et très simples d'utilisation à l'élément cible. L'écriture est toujours la même :

Voici une liste non-exhaustive d'effets proposés par jQuery :

```
.hide() //permet de cacher l'élément. Prendra en paramètre la durée d'exécution exprimée en millisecondes.
.show() // permet de révélé l'élément caché. Prendra en paramètre la durée d'exécution exprimée en millisecondes.
.fadeOut() //permet de cacher l'élément en utilisant un fondu. Prendra en paramètre la durée d'exécution exprimée en millisecondes.
.fadeIn() //permet de cacher l'élément en utilisant un fondu. Prendra en paramètre la durée d'exécution exprimée en millisecondes.
.toggle() //permet d'afficher et de cacher l'élément quand l'action est dépendante par exemple d'un déclencheur
.animate() //permet d'animer les propriétés css de l'élément. Prend en paramètre la durée d'exécution exprimée en millisecondes.
```

Exemple avec *animate()* avec plusieurs paramètres :

```
$('#div_vert').animate({
      //'margin-top':'20px', marginTop :
      '20px', 'width' : "500px", 'height' :
      '500px',
      //'margin-left':'75px' marginLeft :
      '75px'
}, 10000); //ici, je lui indique que l'animation doit se dérouler en 10s.
```

Vous pouvez associer autant d'effets et de méthodes à un élément cible que vous souhaitez.

Je vous invite à vous référer aussi à la documentation officielle de jQuery : http://learn.jquery.com/effects/