

jQuery

Dans le cadre de son métier un concepteur web ou d'un développeur front, vous rencontrerez généralement du JavaScript, souvent appelé JS et son fils naturel jQuery. Dans les 10 000 sites les plus importants, JavaScript est utilisé dans plus de 92% d'entre eux et jQuery est utilisé dans plus de 63% d'entre eux. Inutile de dire qu'ils sont populaires. Vous pouvez même aspirer à écrire du JavaScript ou du jQuery pour créer vos propres programmes à un moment ou un autre.

jQuery Intro

Avec cette quelques considération de base sur JavaScript, il est temps de jeter un oeil à jQuery. jQuery est une bibliothèque JavaScript open source écrit par John Resig qui simplifie l'interaction entre HTML, CSS et JavaScript. Depuis 2006, jQuery a été libéré, il a décollé et est utilisé par les sites Web et les entreprises qu'elles soient grandes et petites.

Ce qui a rendu jQuery si populaire est c'est la facilité d'utilisation, avec notamment des sélections ressemblant aux sélecteurs CSS. Les avantages de jQuery sont énormes, mais pour notre but, nous ne considérons que sa capacité à sélectionner des éléments et effectuer des actions avec eux.

Mise en route de jQuery

La première étape pour utiliser jQuery est déclarer sa présence en y faisant référence dans un document HTML. Comme mentionné précédemment avec JavaScript, cela se fait en utilisant l'élément `script` juste avant la fermeture de la balise `</body>`. Puisque jQuery est une bibliothèque il est préférable de le garder séparée de tous les autres JavaScript.

Lorsque vous faites référence jQuery il y a plusieurs options, en particulier comment souhaitez vous utiliser la version minifiée ou non compressé, ou si vous le souhaitez vous pouvez utiliser un réseau de distribution de contenu, CDN pour Content Delivery Network , comme [Google qui propose les principales bibliothèques](#). Si le code écrit est pour un environnement de production, il est recommandé d'utiliser la version minifiée pour de meilleurs temps de chargement. De plus, en utilisant un

CDN cela aide aussi avec le temps de chargement, avec les avantages potentiels de mise en cache.

```
1 <script src="//code.jquery.com/jquery-3.4.1.min.js">
2 </script>
3 <script src="script.js"></script>
```

Dans l'exemple de code ci-dessus, notez la deuxième référence `script` d'un second fichier JavaScript. Toutes doivent être déclarée dans la page, et les fichiers se trouver dans la bonne arborescence locale ou distantecqfd. En outre, ce fichier est spécifiquement placé après le fichier jQuery afin qu'il puisse faire référence à des fonctions jQuery déjà définies.

Quid du http?

Vous avez peut-être remarqué qu'il n'y a pas de référence dans l'exemple de référence Google CDN ci-dessus. Le http a été omis intentionnellement pour permettre les connexions http et https. Lorsque vous travaillez localement, sans bénéficier d'un serveur Web, le http principal devra être inclus afin d'éviter au navigateur de tenter de localiser le fichier sur le disque local des systèmes.

L'objet jQuery

jQuery est livré avec son propre objet qui est représenté par le signe dollar, `$`, également connu comme `jQuery`. L'objet `$` est spécialement conçu pour la sélection d'un élément, puis retournera ce noeud d'élément pour effectuer une action sur celui ci.

Ces sélections et les actions devront être écrites dans un nouveau fichier, référencé en dehors du fichier de la bibliothèque jQuery.

```
1 $();
2 jQuery();
```

Document ready

Avant de déclencher tout jQuery pour "parcourir" et manipuler une page, il est préférable d'attendre que le chargement complet du DOM soit terminé. Heureusement, jQuery a un événement `.ready()`, en raccourci on

n'a pas besoin de le citer, qui peut être appelé et qui commencera à exécuter les lignes de code que lorsque le document HTML est prêt à être modifié. En plaçant tout notre autre jQuery dans cette fonction, nous pouvons garantir qu'il ne sera pas exécuté tant que la page ne sera pas chargée et que le DOM est prêt.

```
1 $( function(event) {  
2     // jQuery code  
3 });
```

Les sélecteurs

Comme mentionné précédemment, l'un des concepts de base de jQuery est de [sélectionner des éléments](#) et effectuer une action. jQuery a fait un excellent travail ce qui rend la tâche de sélection d'un élément, ou de plusieurs éléments, extrêmement facile en mimant celle de CSS. En plus des sélecteurs généraux CSS, jQuery supporte tous les sélecteurs CSS3 uniques, qui fonctionnent indépendamment du navigateur qui est utilisé. L'invocation de l'objet jQuery, `$()` contenant un sélecteur retourne ce noeud de DOM à manipuler. Le sélecteur sera inscrit dans les parenthèses, `('...')` et peut sélectionner des éléments, tout comme celui de CSS.

```
1 $('.feature');           // Sélecteur de classe  
2 $('li strong');          // Sélecteur descendant  
3 $('em, i');              // Sélecteur multiple  
4 $('a[target="_blank"]'); // Sélecteur d'attribut  
5 $('p:nth-child(2)');     // Sélecteur de pseudo-classe
```

Le mots clés "this"

Lorsque vous travaillez à l'intérieur d'une fonction jQuery vous pouvez sélectionner l'élément qui a été référencé par le sélecteur d'origine. Dans ce cas, le mot-clé `this` peut être utilisé pour faire référence à l'élément sélectionné.

```
1 $('#menu').click(function(event) {  
2     $(this).css(...);  
3 });
```

Filtres de sélection jQuery

Si les sélecteurs CSS ne sont pas assez précis il y a aussi des [filtres](#) personnalisés intégrés dans jQuery. Ces filtres sont une extension de CSS3 et fournissent plus de contrôle sur la sélection d'un élément ou de ses parents.

```
1 $('div:has(strong)');
```

En l'état actuel ces filtres peuvent être utilisés à l'intérieur du sélecteur, mais ne pas être originaire du DOM, ils sont un peu lent. Les meilleurs résultats avec l'utilisation de ces filtres est réalisé en utilisant la méthode `:filter()`, qui fait partie de la fonction de parcours du DOM dans jQuery.

Parcourir

jQuery fournit une poignée de méthodes pour traverser l'arbre DOM, filtrer et sélectionner les éléments si nécessaire.

Pour commencer par des éléments de filtrage à l'intérieur du DOM, il faut effectuer une sélection générale, à partir de laquelle sera parcouru de façon relative à la sélection. Dans l'exemple ci-dessous, la sélection d'origine trouve tous les éléments `div` dans le DOM, qui sont ensuite filtrés à l'aide de la méthode `.not()`. Avec cette méthode spécifique, tous les éléments `div` sans classe `type` ou de `collection` seront sélectionnés.

```
1 $('div').not('.type, .collection');
```

Méthodes de chaînage

Pour encore plus de contrôle sur les éléments sélectionnés, différentes méthodes de déplacement peuvent être enchaînées simplement en utilisant un point entre elles.

L'exemple de code ci-dessous utilise à la fois le procédé `.not()` et la méthode `.parent()`. Combinés ensemble cela ne sélectionnera que les éléments parents des éléments `div` sans classe `type` ou `collection`.

```
1 $('div').not('.type, .collection').parent();
```

Méthodes “traversantes”

JQuery dispose de quelques méthodes de [parcours](#) disponibles. En général, elles se regroupent en trois catégories, en filtrante, en traversante divers et en parcours de l’arborescence DOM. Les méthodes spécifiques dans chaque catégorie peuvent être vues ci-dessous

Filtre

- `.eq()`
- `.filter()`
- `.first()`
- `.has()`
- `.is()`
- `.last()`
- `.map()`
- `.not()`
- `.slice()`

Divers traversant

- `.add()`
- `.andSelf()`
- `.contents()`
- `.end()`

Parcours de l’arborescence DOM

- `.children()`
- `.closest()`
- `.find()`
- `.next()`
- `.nextAll()`
- `.nextUntil()`
- `.offsetParent()`
- `.parent()`
- `.parents()`
- `.parentsUntil()`
- `.prev()`
- `.prevAll()`
- `.prevUntil()`
- `.siblings()`

Manipulation

Sélectionner et parcourir les éléments du DOM est seulement une partie de ce que propose jQuery, une autre partie importante est ce qui est possible de faire avec ces éléments une fois trouvé. Une des possibilités est de [manipuler](#) ces éléments, soit par la lecture, l'ajout ou la modification d'attributs ou de styles. En outre, les éléments peuvent être modifiés dans les DOM, en changeant leur placement, de les supprimer, d'ajouter de nouveaux éléments, et ainsi de suite. Globalement, les options pour manipuler des éléments sont assez vastes.

Récupérer et assigner, "get" et "set"

Les méthodes de manipulation à suivre sont les plus couramment utilisées dans l'une des deux directives, à savoir l'*obtention* ou l'*assignation* d'information. Pour obtenir de l'information il faut l'aide d'un sélecteur en plus une méthode pour déterminer quelle information est à récupérer. En outre, le même sélecteur et le procédé peuvent également être utilisés pour assigner un élément d'information.

```
1 // Récupère la valeur de l'attribut alt
2 $('img').attr('alt');
3
4 // Assigne la valeur "Kangourou" à l'attribut alt
5 $('img').attr('alt', 'Kangourou');
```

Dans les exemples et extraits de suivre les méthodes seront principalement utilisées pour faire une assignation, mais ils peuvent aussi être en mesure d'être utilisé pour récupérer des informations.

Manipulation des attributs

Une partie des éléments pouvant être inspectés et manipulés sont des attributs. Quelques options incluent la possibilité d'ajouter, supprimer ou modifier un attribut ou sa valeur. Dans les exemples ci-dessous la méthode `.addClass()` est utilisée pour ajouter une classe à tous les éléments de la même liste, la méthode `.removeClass()` est utilisée pour supprimer toutes les classes de tout paragraphes, et enfin la méthode `.attr()` est utilisée pour déterminer la valeur de l'attribut `title` d'un élément `abbr` et mis à `Hello World`.

```
1 $('li:even').addClass('even-item');
2 $('p').removeClass();
3
```

```
$('abbr').attr('title', 'Hello World');
```

Méthodes de manipulation d'attribut

- `.addClass()`
- `.attr()`
- `.hasClass()`
- `.prop()`
- `.removeAttr()`
- `.removeClass()`
- `.removeProp()`
- `.toggleClass()`
- `.val()`

Manipulation de style

En plus de manipuler les attributs, le style d'un élément peut également être manipulé en utilisant une variété de méthodes. Pour lire ou assigner la hauteur, la largeur ou la position d'un élément, il y a une poignée de méthodes spéciales disponibles, et pour toutes les autres manipulations de style la méthode `.css()` peut gérer toute modification CSS.

La méthode `.css()` peut en particulier être utilisée pour définir une propriété, ou plusieurs, et la syntaxe pour chaque variable. Pour définir une propriété, le nom de la propriété et la valeur doivent chacun être entre guillemets et séparés par des virgules. Pour définir les propriétés multiples, les propriétés doivent être imbriquées à l'intérieur des accolades avec le nom de la propriété, en supprimant les traits d'union si nécessaire, suivi par deux points et la valeur indiquée. Chacune des paires de propriétés et de valeur doivent être séparés par des virgules. La hauteur, la largeur, ou mes méthodes de positionnement utilisent par défaut les pixels, cependant, d'autres unités de mesure peuvent être utilisées. Comme on le voit ci-dessous, pour changer l'unité de mesure d'identifier la valeur puis utilisez un signe plus suivi par l'unité de mesure choisie.

```

1 $('h1 span').css('font-size', 'normal');
2 $('div').css({
3     fontSize: '13px',
4     background: '#f60'
5 });
6 $('header').height(200);
7 $('.extend').height(30 + 'em');

```

Méthodes de manipulation de style

- .css()
- .height()
- .innerHeight()
- .innerWidth()
- .offset()
- .outerHeight()
- .outerWidth()
- .position()
- .scrollLeft()
- .scrollTop()
- .width()

Manipulation DOM

Enfin, nous sommes en mesure d'inspecter et manipuler le DOM, en changeant le placement des éléments, l'ajout et la suppression d'éléments. Les options ici sont variées, ce qui permet des changements potentiels à apporter à l'intérieur du DOM.

Chaque méthode de manipulation du DOM a sa propre syntaxe , mais certaines d'entre eux sont décrites ci-dessous. La méthode `.prepend()` est l'ajout d'un nouvel élément `h3` juste à l'intérieur toute `section`, la méthode `.after()` est l'ajout d'un nouvel élément `em` juste après la liaison, et la méthode `.text()` remplace le texte des éléments `h1` avec le texte `Hello World`.

```

1 $('section').prepend('<h3>Featured</h3>');
2 $('a[target="_blank"]').after('<em>New window.</em>');
3 $('h1').text('Hello World');

```

Méthodes de manipulation du DOM

- .after()

- `.append()`
- `.appendTo()`
- `.before()`
- `.clone()`
- `.detach()`
- `.empty()`
- `.html()`
- `.insertAfter()`
- `.insertBefore()`
- `.prepend()`
- `.prependTo()`
- `.remove()`
- `.replaceAll()`
- `.replaceWith()`
- `.text()`
- `.unwrap()`
- `.wrap()`
- `.wrapAll()`
- `.wrapInner()`

Événements

L'une des beautés de jQuery est la possibilité d'ajouter facilement des [gestionnaires d'événements](#) , qui sont des méthodes qui sont appelées uniquement sur un événement spécifique ou une action qui se déroule. Par exemple, la méthode d'ajout d'une classe à un élément peut être configurée pour se produire uniquement lorsque cet élément sera cliqué. Ci-dessous un sélecteur standard, saisissant tous les éléments de la liste. La méthode d'événement `.click()` est liée au sélecteur d'élément de liste, mise en place d'une action qui aura lieu après avoir un click sur tout élément de liste. A l'intérieur du `.click()` on va ajouter une fonction qui va englober les actions à exécuter lorsque la méthode d'événement `click` va être exécutée. Les parenthèses directement après la fonction sont disponibles pour passer des paramètres dans la fonction, l'objet `event` est utilisé dans cet exemple.

A l'intérieur de la fonction on à un sélecteur sur lequel on utilise la méthode `.addClass()`. Maintenant, lorsqu'un élément de liste est cliqué cet élément de liste, via le mot-clé `this`, reçoit la classe `saved-item`.

```
1 $('li').click(function(event){
2     $(this).addClass('saved-item');
3 });
```

Flexibilité de l'événement

La méthode d'événement `.click()`, avec une poignée d'autres méthodes d'événement, est en fait une méthode abrégée qui utilise la méthode `.on()` introduite dans jQuery 1.7. La méthode `.on()` offre un peu de flexibilité, en utilisant la délégation automatique pour les éléments qui sont ajoutés à la page dynamique.

Pour faire usage de la méthode `.on()`, le premier argument doit être le nom de l'événement natif alors que le second argument doit être la fonction de gestionnaire d'événements. En regardant l'exemple précédent, la méthode `.on()` est appelée à la place de la méthode `.click()`. Le nom de l'événement `click` est transmis comme premier argument dans la méthode `.on()` avec la fonction de gestionnaire d'événements restant le même qu'avant.

```
1 $('li').on('click', function(event){
2     $(this).addClass('saved-item');
3 });
```

Imbriquer des événements

Il est possible d'avoir plusieurs gestionnaires d'événements et déclencheurs, imbriqués l'un dans l'autre. A titre d'exemple, en dessous de la méthode d'événement `.on()` l'argument `hover` est transmis étant appelée ainsi en survol de la souris au-dessus de tout élément de la classe `pagination`. Lors de l'appel de `.on()` sur le survol l'événement `.click()` est appelé pour l'ancre avec l'id `up`.

```
1 $('.pagination').on('hover', function(event){
2     $('a#up').click();
3 });
```

Démo Event <http://codepen.io/shayhowe/pen/qnzKC>

L'utilisation d'un message d'alerte comme une démo, les extraits de code suivant montre comment créer un message d'alerte, puis retirer ce message lorsque l'on clique sur l'icône de fermeture.

HTML

```
1 <div class="notice-warning">
2   <div class="notice-close">&#215;</div>
3   <strong>Warning!</strong> I&#8217;m about to lose my
4   cool.
5 </div>
```

JavaScript

```
1 $('<strong>Warning!</strong> I&#8217;m about to lose my
2   cool.
3 </div>')<strong>Warning!</strong> I&#8217;m about to lose my
4   cool.
5 </div>').on('click', function(event) {
6   $('<strong>Warning!</strong> I&#8217;m about to lose my
7   cool.
8 </div>').remove();
9 });
```

Méthodes d'événement

jQuery fournit un bon nombre de méthodes, qui permettent l'enregistrement des comportements des utilisateurs qui interagissent avec le navigateur. Ces méthodes enregistrent un bon nombre d'événements, les plus populaires, mais sans s'y limiter, le navigateur, les formulaires, le clavier et les événements de la souris. Les méthodes les plus populaires comprennent:

Evénements navigateur

- `.resize()`
- `.scroll()`

Chargement des documents

- `.ready()`

Gestionnaire d'événements

- `.off()`
- `.on()`
- `.one()`
- `jQuery.proxy()`
- `.trigger()`
- `.triggerHandler()`
- `.unbind()`
- `.undelegate()`

Objet de l'événement

- `event.currentTarget`
- `event.preventDefault()`
- `event.stopPropagation()`
- `event.target`
- `event.type`

Événements Formulaire

- `.blur()`
- `.change()`
- `.focus()`
- `.select()`
- `.submit()`

Événements clavier

- `.focusin()`
- `.focusout()`
- `.keydown()`
- `.keypress()`
- `.keyup()`

Événements de la souris

- `.click()`
- `.dblclick()`
- `.focusin()`
- `.focusout()`
- `.hover()`
- `.mousedown()`
- `.mouseenter()`
- `.mouseleave()`
- `.mousemove()`
- `.mouseout()`
- `.mouseover()`
- `.mouseup()`

Effets

À côté des événements, jQuery fournit également une poignée d'effets personnalisables. Ces effets se font de différentes façons, y compris les méthodes d'événements pour afficher et masquer le contenu, faire des transitions avec par exemple un effet fondu, ou glissant vers le haut et

vers le bas. Tous ces éléments sont prêts à utiliser des méthodes et peuvent être personnalisés.

Chaque méthode a sa propre syntaxe il est donc préférable de faire référence à la [documentation JQuery sur ces effets](#) pour la syntaxe spécifique autour de chaque méthode.

Le plus souvent cependant, les effets acceptent généralement une durée, un type d'effet et la possibilité de spécifier une fonction de rappel.

jQuery et animations CSS

Les animations personnalisées de différentes propriétés CSS peuvent être accomplies dans jQuery, bien que ce soit un peu moins pertinent que d'utiliser CSS peut désormais gérer lui-même des animations. Les animations CSS offrent de meilleures performances d'un point de vue de traitement du navigateur et seront préférés lorsque cela est possible. Les effets d'animation jQuery, notamment avec l'aide de Modernizr, peuvent faire une solution de repli pour tous les navigateurs qui ne supportent pas les animations CSS même si cela commence à se raréfier...

Durée d'effet

L'utilisation de la méthode `.show()`, par exemple, le premier paramètre disponible à passer éventuellement dans le procédé est la durée, qui peut être réalisée en utilisant une valeur clé ou en millisecondes. Le mot-clé `slow` correspond par défaut à 600 millisecondes, alors que le mot-clé `fast` lui est égale à 200 millisecondes. L'utilisation d'une valeur de mot-clé est courante, mais les valeurs de milliseconde peut également être transmis directement en paramètre. Les valeurs de mots-clés doivent être mise entre quote (') alors que les valeurs en millisecondes ne le font pas.

```
1 $('.error').show();  
2 $('.error').show('slow');  
3 $('.error').show(500);
```

Effet Easing

En plus de définir la durée pendant laquelle un effet a lieu, le déroulement ou la vitesse à laquelle une animation progresse à différents moments dans l'animation, peut également être réglée. Par défaut, jQuery a deux valeurs de mots clés pour faciliter cela, la valeur par défaut est `swing`, la valeur alternative étant `linear`. La valeur par défaut la `swing` commence l'animation à un rythme lent, prenant de la vitesse lors de l'animation, et

puis ralentit à nouveau avant la fin. La valeur `linear` exécute l'animation à un rythme constant pendant toute la durée.

```
1 $('.error').show( 'slow', 'linear' );  
2 $('.error').show( 500, 'linear' );
```

jQuery UI

Les deux valeurs d'accélération qui viennent avec jQuery peuvent être étendues par l'utilisation de différents modules d'extension, qui peuvent offrir des valeurs supplémentaires. L'un des plug-ins est le plus populaire est [jQuery UI](#).

En plus de nouvelles valeurs de déroulement jQuery UI fournit également une poignée d'autres interactions, des effets, des widgets et d'autres ressources utiles.

Effet de “callback”

Quand une animation est terminée, il est possible d'exécuter une autre fonction, appelée fonction de callback. Cette fonction doit être placée après la durée du déroulement, si elle existe. A l'intérieur de cette fonction de nouveaux événements ou effets peuvent être placés, chacun suivant leur propre syntaxe requise.

```
1 $('.error').show( 'slow', 'linear', function(event) {  
2     $('.error .status').text( 'Continue' );  
3 });
```

Syntaxe des effets

Comme mentionné précédemment, chaque méthode d'effet a sa propre syntaxe qui se trouve dans la [documentation des effets jQuery](#). La durée, de déroulement et les paramètres de rappel décrits ici sont communs, mais non disponible sur toutes les méthodes. Il est préférable d'examiner la syntaxe d'une méthode pour pouvoir l'utiliser pleinement.

Demo effets <https://codepen.io/shayhowe/pen/qciLt>

En prenant les mêmes événements démo ci-dessus, la méthode `.remove()` est maintenant utilisée dans le cadre d'une fonction de callback sur la méthode `.fadeOut()`. L'utilisation de la méthode `.fadeOut()` fait que le message d'alerte va progressivement disparaître plutôt que de

disparaître rapidement et il sera alors retiré du DOM une fois l'animation terminée.

HTML

```
1 <div class="notice-warning">
2   <div class="notice-close">&#215;</div>
3   <strong>Warning!</strong> I&#8217;m about to lose my
4   cool.
5 </div>
```

JavaScript

```
1 $('<strong>Warning!</strong> I&#8217;m about to lose my
2   cool.').appendTo('div').fadeOut('slow',
3   function(event) {
4     $(this).remove();
5   });
6
```

Effets de base

- .hide()
- .show()
- .toggle()

Effets personnalisés

- .animate()
- .clearQueue()
- .delay()
- .dequeue()
- jQuery.fx.interval
- jQuery.fx.off
- .queue()
- .stop()

Effets évanouissements

- .fadeIn()
- .fadeOut()
- .fadeTo()
- .fadeToggle()

Effets de glissement

- `.slideDown()`
- `.slideToggle()`
- `.slideUp()`

Démo Slide <https://codepen.io/shayhowe/pen/xGBfe>

HTML

```
1 <div class="panel">
2   <div class="panel-stage"></div>
3   <a href="#" class="panel-tab">Afficher
4   <span>&#9660;</span></a>
5 </div>
```

JavaScript

```

1  $(' .panel-tab').on('click', function(event){
2      event.preventDefault();
3      $(' .panel-stage').slideToggle('slow',
4      function(event){
5          if($(this).is(':visible')){
6              $(' .panel-tab').html('Close
7              <span>#9650;</span>');
8          } else {
9              $(' .panel-tab').html('Open
10             <span>#9660;</span>');
11         }
12     });
13 });

```

Demo tabs <https://codepen.io/shayhowe/pen/klAvu>

HTML

```
1 <ul class="tabs-nav">
2   <li><a href="#tab-1">Notice</a></li>
3   <li><a href="#tab-2">Prix</a></li>
4 </ul>
5 <div class="tabs-stage">
6   <div id="tab-1">Lorem ipsum dolor sit amet,
7   consectetur adipiscing elit. Phasellus nec neque nisi,
8   dictum aliquet lectus.</div>
9   <div id="tab-2">Phasellus pharetra aliquet viverra.
```



```
Donec scelerisque tincidunt diam, eu fringilla urna  
auctor at.</div>  
</div>
```

JavaScript

```
1 // Show the first tab by default  
2 $('.tabs-stage div').hide();  
3 $('.tabs-stage div:first').show();  
4 $('.tabs-nav li:first').addClass('tab-active');  
5  
6 // Change tab class and display content  
7 $('.tabs-nav a').on('click', function(event){  
8     event.preventDefault();  
9     $('.tabs-nav li').removeClass('tab-active');  
10    $(this).parent().addClass('tab-active');  
11    $('.tabs-stage div').hide();  
12    $($ (this).attr('href')).show();  
13 });
```

Ressources et liens

- [jQuery Documentation](#)
- [jQuery UI](#)