JQUERY

?

Un **FRAMEWORK** basé sur le langage Javascript, spécialisé dans la **MANIPULATION DE DOM.**

Facilite le développement et permet de s'affranchir d'une partie des problèmes de compatibilité cross browser.

jQuery **EST** Javascript : il ne dispense donc pas de connaître le langage.

CDN

L'utilisation d'un Content Delivery Network est une bonne pratique.
Mais il faut prévoir le cas où il est indisponible.

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js'
<script>window.jQuery || document.write('<script src="path/to/local/jquer
```

CDN officiel : http://code.jquery.com/

Quelle version ? jQuery 1.x si IE < 9 ; jQuery 2.x sinon.

LA FONCTION $

```
jQuery();
$();
```

Accepte des paramètres et retourne une "collection jQuery" d'éléments DOM.

Doc : http://api.jquery.com/jquery/

NO CONFLICT

Plusieurs frameworks JS utilisent le symbole $.
On peut indiquer à jQuery de ne pas s'approprier ce symbole.

```
jQuery.noConflict(); // libère le $
$; // ne désigne plus jQuery
jQuery; // ok
```

Autre solution : éviter d'utiliser plusieurs frameworks ;-)

DOCUMENT READY

Pour manipuler le DOM, il faut attendre qu'il soit chargé...

```
// le classique
$(document).ready(function(){
    // ...
});

// et son alias
$(function(){
    // ...
});
```

```
$(window).load(function(){
    // ...
});
```

SÉLECTION D'ÉLÉMENTS DOM

Comme son nom l'indique (pas trop),
jQuery a été conçu pour utiliser des sélecteurs CSS :

```
// Les éléments "label"
$("label");

// Les éléments qui ont l'id "user-form"
$("#user-form");

// Les éléments qui ont la class "submit"
var submits = $(".submit");

// Le deuxième élément qui a la classe "submit"
submits.eq(1);
```

```
$("h2");
$("aside#right-col.blue");
$("li a");
$("li > a"); // les a descendants directs de li
$("img + p"); // les p immédiatement précédés de img
```

PARCOURIR LE DOM

Se déplacer par rapport à un objet jQuery :

```
var element = $("div").eq(0); // un sélecteur quelconque

element.find(selector); // ses enfants sans limite de profondeur
element.children(selector); // ses enfants immédiats
element.parent(selector); // son parent
element.parents(selector); // son parent et ses ancêtres
element.closest(selector); // son parent ou ancêtre le plus proche
element.siblings(selector); // ses frères
element.next(selector); // son frère suivant
element.prev(selector); // son frère précédent
```

Où `selector` représente un sélecteur CSS comme vu précédemment.

ÉCOUTEURS D'ÉVÈNEMENTS

Avec jQuery et la fonction `click` :

```
var link = $("a:eq(0)");

link.click(function(e){
    e.preventDefault();
    alert("Hello, world!");
});
```

Avec jQuery et la fonction `on` :

```
var link = $("a:eq(0)");

link.on("click", function(e){
    e.preventDefault();
    alert("Hello, world!");
});
```

```
link.on("click focus", function(e){});

link.on({
    click: function(e){},
    keyup: function(e){}
});
```

ÉCOUTEURS D'ÉVÈNEMENTS

La fonction `on` de jQuery pour des éléments qui ne sont pas encore dans le DOM :

```
var list = $("ul:eq(0)"), // Cette balise ul est présente au chargement d
    links = $("a", list); // Les liens contenu dans cette liste

// Écoute les liens présents au document ready
links.on("click", function(e){
    // ...
});

// Écoute aussi les liens qui ont été ajoutés dynamiquement après le docu
list.on("click", "a", function(e){
    // ...
});
```

Dans le deuxième exemple, `list` fait office de contexte. Éviter d'utiliser `$(document)` comme contexte.

Attention, dans cet exemple les liens présents au document ready déclencheront les deux callbacks.

ÉCOUTEURS D'ÉVÈNEMENTS

- La fonction de callback fournit toujours l'évènement, noté `e` par convention.
- Dans le callback, `this` désigne l'élément courant, et `$(this)` sa version jQuery.
- Attention (notamment dans les boucles) à ne pas définir plusieurs fois les mêmes écouteurs car ils se cumulent.

Pour décrocher un écouteur :

```
link.off("click");
```

DÉCLENCHER UN ÉVÈNEMENT

```
link.click(); // JS brut
link.trigger("click"); // jQuery
```

Par sécurité, tous les évènements ne peuvent pas être déclenchés depuis le code.

ÉVÈNEMENTS PERSONNALISÉS

Avec jQuery, il n'y a rien de spécial à faire !

```
var link = $("a:eq(0)");

link.on("bored", function(e){
    alert("Ehhh, I'm getting bored! :-(");
});

setTimeout(function(){
    link.trigger("bored");
}, 3600*1000); // 1h
```

En JS brut c'est plus compliqué : objet `CustomEvent`, fonction `dispatchEvent`.

COLLECTIONS

```
var elements = $("h2");

elements.length; // Une collection est un tableau évolué

// Convertit la collection jQuery en tableau d'objets DOM
elements.get(); // Tableau simple

elements.doSomething();

elements.each(function(index, element){
    console.log(this === element); // true
    console.log($(this) === $(element)); // true
});
```

PSEUDO SÉLECTEURS

Divers

```
$(":empty");
$(":first");
$(":first-child");
$(":last-child");
$(":last");
$(":nth-child()");
$(":only-child");
$(":even");
$(":odd");
```

PSEUDO SÉLECTEURS

Divers

```
$(":eq()");
$(":gt()");
$(":lt()");
$(":header"); // titres de h1 à h6
$(":hidden");
$(":visible");
$(":not()");
```

PSEUDO SÉLECTEURS

Formulaires

```
$(":input"); // input, textarea, select et button
$(":button"); // button
$(":checkbox"); // checkbox
$(":checked"); // checked (checkboxes)
$(":selected"); // selected (selects)
$(":radio"); // type radio
$(":submit"); // type submit (input ou button)
$(":text"); // type text
$(":password"); // type password
$(":focus"); // éléments qui ont le focus
$(":enabled"); // éléments activés
$(":disabled"); // éléments désactivés (grisés)
```

SÉLECTEURS D'ATTRIBUTS

Les plus courants.

```
$('[href="http://www.umanit.fr"]'); // égal
$('[href!="http://www.umanit.fr"]'); // pas d'attribut href ou différent
$('[href*="https"]'); // contient
$('[href^="https"]'); // commence par
$('[href$=".fr"]'); // se termine par
```

SÉLECTEURS (EN TOUS GENRES)

Possibilité de les cumuler.

```
$(":header:not(h1)");
$("tr:even");
$("tr:not(tr:last)");
$("td:gt(4):lt(10)");
```

Et plusieurs façons d'écrire la même chose.

```
$('a[href^="http"][href$=".fr"]:not(:visible)');
$('a[href^="http"][href$=".fr"]:hidden');
```

L'ATTRIBUT DATA-*

Introduit en HTML5, il permet aux éléments du DOM de stocker des informations destinées à être utilisées par JS.

Il n'est pas interprété par les navigateurs comme faisant partie du contenu de la page.

```
<ul>
    <li><img src="path/to/img" alt="Le passage Pommeraye" data-slide-id="
    <li><img src="path/to/img" alt="La grue Titan" data-slide-id="1"></li
    <li><img src="path/to/img" alt="La Tour LU" data-slide-id="2" data-cu
</ul>
```

```
var currentSlide = $("li img[data-current-slide]"),
    currentSlideId = currentSlide.data('slide-id');
```

L'ATTRIBUT DATA-*

Bonne pratique pour cibler un bouton d'action :

```
var singButton = $('[data-trigger="sing"]');
```

Plus propre que de se baser sur le CSS et évite ce genre de choses :

```
var singButton = $('.rightcol > a.full-width');
```

Si la structure de la page ou le CSS change, le code ne fonctionnera plus !

PLUGIN

```
(function($) {
    $.fn.myCustomFunction = function(a, b, c) {
        // ...
    };
})(jQuery);
```

```
var a,
    b,
    c;
$("a").myCustomFunction(a, b, c);
```