

# Jquery

Cognitic Sprl

JQuery

# **PREMIERS PAS**

# Premiers Pas - Intérêt de JQuery

La bibliothèque jQuery fournit une couche d'abstraction générique pour les scripts web classiques.

Les fonctionnalités standard permettent de répondre aux besoins suivants :

## □ Accéder aux éléments d'un document.

Sans l'aide d'une bibliothèque JavaScript, il faut écrire de nombreuses lignes de code pour parcourir l'arborescence du *DOM (Document Object Model)* et localiser des parties spécifiques de la structure d'un document HTML. jQuery fournit un mécanisme de sélection robuste et efficace qui permet de retrouver n'importe quels éléments d'un document afin de les examiner ou de les manipuler

# Premiers Pas - Intérêt de JQuery

- **Modifier l'aspect d'une page web**

CSS propose une solution puissante pour modifier le rendu des documents, mais elle montre ses limites lorsque les navigateurs web ne respectent pas les mêmes standards. Avec jQuery, les développeurs peuvent contourner ce problème en se fondant sur une prise en charge identique des standards quels que soient les navigateurs. Par ailleurs, la bibliothèque permet de modifier les classes ou les propriétés de style appliquées à une partie du document, même après que la page ait été affichée.

- **Altérer le contenu d'un document.**

jQuery ne se borne pas à des changements cosmétiques mais permet de modifier le contenu du document. Du texte peut être changé, des images peuvent être insérées ou interverties, des listes être réordonnées, l'intégralité de la structure du contenu HTML peut être revue et étendue. Toutes ces possibilités sont permises par une *API (Application Programming Interface)* simple d'emploi.

# Premiers Pas - Intérêt de JQuery

- **Répondre aux actions de l'utilisateur**

Même les comportements les plus élaborés et les plus puissants ne sont d'aucune utilité si leur exécution n'est pas contrôlée. La bibliothèque jQuery propose une solution élégante pour intercepter une grande variété d'événements, comme un clic sur un lien, sans avoir à mélanger des gestionnaires d'événements au code HTML. De plus, cette API de gestion des événements permet de passer outre les incohérences des navigateurs, qui constituent une véritable nuisance pour les développeurs web.

- **Animer les modifications d'un document**

Pour la bonne mise en œuvre des comportements interactifs, le concepteur doit également fournir un retour visuel à l'utilisateur. La bibliothèque jQuery apporte son aide en proposant des animations, comme les effets de fondu et de volet, ainsi qu'une boîte à outils pour en construire de nouvelles.

# Premiers Pas - Intérêt de JQuery

- **Récupérer des informations à partir d'un serveur sans actualiser la page**

Ce type de code, connu sous le nom *AJAX (Asynchronous JavaScript And XML)*, aide les développeurs web à créer un site réactif proposant des fonctionnalités riches. La bibliothèque JQuery permet de retirer de ce code les complexités propres aux navigateurs et de laisser les développeurs se concentrer sur l'aspect serveur.

- **Simplifier les tâches JavaScript courantes**

Outre les fonctionnalités de JQuery orientées document, la bibliothèque apporte des améliorations aux constructions JavaScript de base, comme les itérations et la manipulation des tableaux.

# Premiers Pas – Mise en place

jQuery étant une librairie côté client, sa mise en place est extrêmement simple.

Dans un premier temps, il faut télécharger jQuery ☐ <http://jquery.com/>



Deux version disponibles :

- ☐ production (production) de 32 KB
- ☐ Développement (development) de 247 KB.

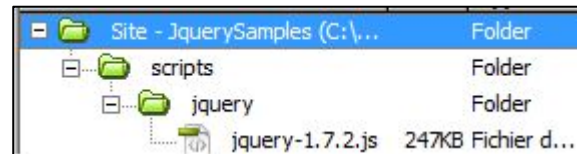
Cette dernière est bien adaptée à notre étude car elle permet au plus curieux de jeter un oeil sur le code utilisé par jQuery.

La version de

production est utilisée une fois le travail de développement terminé et vos pages mises en ligne.

# Premiers Pas – Mise en place

Afin de bien structurer le site utilisant jquery, nous veillerons à inclure le script téléchargé dans un dossier dédié.



Ensuite, il nous suffira de lier la librairie dans la page avant l'utiliser.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>JQuery Sample</title>
<script language="javascript" type="application/javascript" src="scripts/jquery/jquery-1.7.2.js"></script>
</head>

<body>
</body>
</html>
```

Il est également possible d'utiliser directement la librairie hébergée par Google dans l'Ajax Librairies Api

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/latest/jquery.min.js"></script>
</head>
```



# Quelques notions théoriques

Les scripts JQuery s'articulent autour d'un modèle unique.

```
<script>
$(function()
{
    //contenu exécuté lors du chargement du document
});
</script>
```

Il est, cependant , préférable d'attendre que le document soit prêt avant d'exécuter la fonction.

```
<script>
$(document).ready(function()
{
    //contenu exécuté lors du chargement du document
});
</script>
```

La particularité de la fonction ready est de charger les éléments DOM dès que ceux-ci sont disponibles.

En javascript classique, on utilise le classique *window.onload= function* qui attend que la page et tous ces éléments soient complètement chargés . Cela peut-être long (ex: images).

# Quelques notions théoriques

Jquery se base de façon native sur les éléments du DOM.

## Avantages :

- ⇒ Tous les éléments de la page sont incluse dans un objet que les sélecteurs, méthodes et fonctions de Jquery pourront manipuler
- ⇒ Le code html est dépouillé de toutes mentions Javascript comme par exemple `<a href="javascript:void(0);" >lien</a>`. Tout le code javascript se trouve dans une partie séparée de la page html ou dans un fichier externe (séparation contenu/présentation)

# Une première appli

## 1) Le fichier html

<b>Chapitre 1</b>	<pre>&lt;html xmlns="http://www.w3.org/1999/xhtml"&gt; &lt;head&gt; &lt;meta http-equiv="Content-Type" content="text/html; charset=utf-8" /&gt; &lt;title&gt;Première application&lt;/title&gt; &lt;style&gt; body{margin:0px; font :12px; font-family:Verdana, Geneva, sans-serif;} .menu{width:150px;} .menu_chapitre { padding:5px; cursor:pointer; position:relative; margin:1px; font-weight:bold; background: #9cf; border: 1px solid #000;} .menu_point a { display:block; color:black; background-color: white; padding-left:30px; text-decoration: none;} .menu_point a:hover { color:black; text-decoration: underline;} &lt;/style&gt; &lt;/head&gt; &lt;body&gt; &lt;div&gt;   &lt;div class="menu"&gt;     &lt;p class="menu_chapitre"&gt;Chapitre 1&lt;/p&gt;     &lt;div class="menu_point"&gt;       &lt;a href="#"&gt;Point 1&lt;/a&gt;       &lt;a href="#"&gt;Point 2&lt;/a&gt;       &lt;a href="#"&gt;Point 3&lt;/a&gt;     &lt;/div&gt;     &lt;p class="menu_chapitre"&gt;Chapitre 2&lt;/p&gt;     &lt;div class="menu_point"&gt;       &lt;a href="#"&gt;Point 1&lt;/a&gt;       &lt;a href="#"&gt;Point 2&lt;/a&gt;       &lt;a href="#"&gt;Point 3&lt;/a&gt;     &lt;/div&gt;     &lt;p class="menu_chapitre"&gt;Chapitre 3&lt;/p&gt;     &lt;div class="menu_point"&gt;       &lt;a href="#"&gt;Point 1&lt;/a&gt;       &lt;a href="#"&gt;Point 2&lt;/a&gt;       &lt;a href="#"&gt;Point 3&lt;/a&gt;     &lt;/div&gt;   &lt;/div&gt; &lt;/div&gt; &lt;/body&gt; &lt;/html&gt;</pre>
Point 1	
Point 2	
Point 3	
<b>Chapitre 2</b>	
Point 1	
Point 2	
Point 3	
<b>Chapitre 3</b>	
Point 1	
Point 2	
Point 3	

# Une première appli

## 2) Ajout de la librairie et rendre invisible les points

```
<script language="javascript" type="application/javascript" src="scripts/jquery/jquery-1.7.2.js"></script>
<script>
$(document).ready(function()
{
    $("div.menu_point").hide();
});
</script>
```

Chapitre 1

Chapitre 2

Chapitre 3

## 3) Déroulement du sous-menu

```
<script language="javascript" type="application/javascript" src="scripts/jquery/jquery-1.7.2.js"></script>
<script>
$(document).ready(function()
{
    $("div.menu_point").hide();
    $("p.menu_chapitre").click(function() {
        $(this).next("div.menu_point").slideDown(300).siblings("div.menu_point").slideUp("slow");
    });
});
</script>
```

JQuery

# LES SÉLECTEURS

# Les sélecteurs

Les sélecteurs sont l'un des aspects le plus important de la librairie jQuery.

Ceux-ci utilisent une syntaxe qui n'est pas sans rappeler celle des feuilles de style CSS. Ils permettent aux concepteurs d'identifier rapidement et aisément n'importe quel élément de la page et d'y appliquer les méthodes spécifiques à jQuery

Le site suivant permet de se rendre compte de la puissance des sélecteurs :

<http://www.codylindley.com/jqueryselectors/>

## Exemples de sélecteurs :

- `$('#header')` ☐ sélectionne l'objet html ayant pour id "header"
- `$('.container')` ☐ sélectionne tous les objets ayant la classe "container"
- `$('a')` ☐ sélectionne l'ensemble des liens de la page
- `$('#maListe li')` ☐ sélectionne l'ensemble des 'li' de la liste #maListe
- `$('input:reset')` ☐ sélectionne tous les boutons de type "reset"

# Les sélecteurs

## SELECTORS

#id, tag, .class, *	E[@attr]	
elm1, elm2, elmN	E[@attr=val]	
ancestor descendant	E[@attr^=val] (begins)	
parent > child	E[@attr\$=val] (ends)	
parent/child	E[@attr*=val] (contains)	
prev + next	E[@attr=val][@attr=val] (both)	
prev ~ siblings		
	:nth-child( index )	
:first	:first-child	
:last	:last-child	
:not( selector )	:only-child	:input
:even		:text
:odd	:enabled	:password
:eq( index )	:disabled	:radio
:gt( index )	:checked	:checkbox
:lt( index )	:selected	:submit
		:image
:contains( text )	:hidden	:reset
:empty	:visible	:button
:has( selector )	:header	:file
:parent	:animated	:hidden

# Les sélecteurs

## A. Sélection par l'identifiant

### **#identifiant**

Sélectionne l'élément (unique) spécifié par l'attribut id="identifiant".

`$("#box")` : sélectionne l'élément dont l'id est box.

C'est la notation jQuery de `getElementById` du JavaScript classique.

### **Remarque :**

Rappelons que cet identifiant doit être unique dans la page. Si par erreur, ce n'était pas le cas, jQuery reprend le premier élément doté de cet identifiant.

Exemple : [deux.html](#)

Paragraphe 1

Paragraphe 2

Paragraphe 3



# Les sélecteurs

## B. Sélection par le nom de la balise

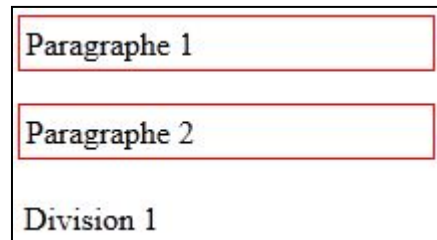
### élément

Sélectionne tous les éléments (ou balises) dont le nom est spécifié.

`$("div")` : sélectionne toutes les divisions `<div>` de la page.

C'est la notation jQuery de `getElementsByName` du JavaScript classique.

Exemple : [trois.html](#)



# Les sélecteurs

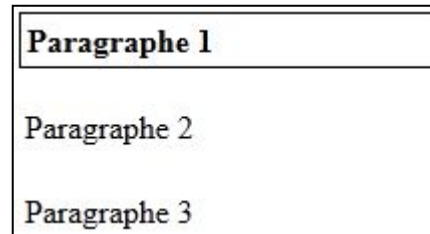
## C. Sélection par la classe

### **.classe**

Sélectionne tous les éléments (ou balises) avec la classe spécifiée.

`$(".texte")` : sélectionne tous les éléments dotés de l'attribut `class="texte"`.

Exemple : [quatre.html](#)



Pour obtenir ce résultat, nous avons utilisé `$(".gras").css("border", "1px solid black");`

Nous aurions pu également écrire `$("p.gras").css("border", "1px solid black");`

Dans notre deuxième cas, jquery va d'abord récupérer les balises `<p>` et puis sélectionner celles qui ont la classe *gras*

# Les sélecteurs

La notation avec plusieurs classes est envisageable.

`$(".classe1.classe2")` sélectionne tous les éléments qui ont comme nom de classe classe1 et classe2.

Le sélecteur étoile `*` permet de sélectionner tous les éléments :

`$("*").css("border", "1px solid black");`

jQuery permet d'associer plusieurs sélecteurs:

`$("div,span,p.nom_classe").css("border", "1px solid black");`

# Les sélecteurs

## □ Les sélecteurs hiérarchiques

La notation DOM avec ses parents, descendants, enfants, frères et sœurs (siblings) est maintenant bien ancrée dans l'écriture du JavaScript.

### A. Sélection des descendants

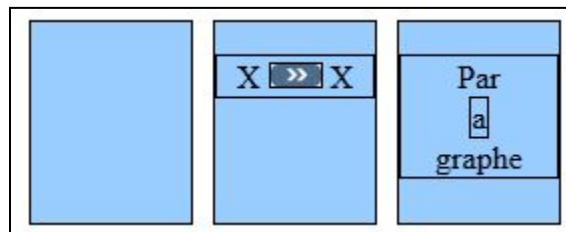
#### **Ascendant Descendant**

Sélectionne tous les descendants de l'élément noté "Descendant" par rapport à l'élément parent noté "Ascendant".

`$("#box p")` : sélectionne tous les descendants de `<p>` contenus dans l'élément parent identifié par `box`.

Les descendants peuvent être les enfants, les petits-enfants et les arrière-petits-enfants.

Exemple : [descendant.html](#)



# Les sélecteurs

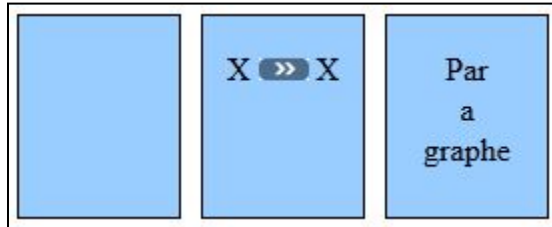
## B. Sélection des enfants

### **Parent > Enfant**

Sélectionne tous les éléments notés par "Enfant" qui sont les enfants directs de l'élément parent noté "Parent".

`$("#box > p")` : sélectionne tous les enfants immédiats de `<p>` contenus dans l'élément parent identifié par `box`.

Exemple : [parentEnfant.html](#)



### Remarque :

Seuls les enfants directs sont sélectionnés

# Les sélecteurs

## C. Sélection des frères suivants

### **Précédent ~ Frères**

Cible les éléments frères situés après l'élément identifié par le sélecteur "Précédent" et qui répondent au sélecteur "Frères".

`$("#prev ~ div")` trouve toutes les divisions `<div>` qui sont frères après l'élément avec `#prev` comme identifiant.

Exemple : [frere.html](#)

•	Item 1
•	Item 2
•	Item 3
•	Item 4
•	Item 5

# Les sélecteurs

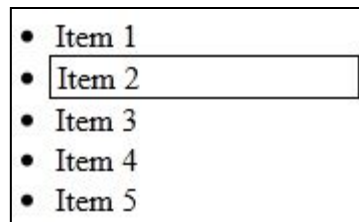
## D. Sélection de l'élément suivant

### **Précédent + Suivant**

Cible l'élément immédiatement suivant situé après l'élément identifié par le sélecteur "Précédent" et qui répond au sélecteur "Suivant".

`$("#prev + div")` trouve la division `<div>` qui suit l'élément avec `#prev` comme identifiant.

Exemple [precedentSuivant.html](#)



JQuery

# LES SÉLECTEURS DE FORMULAIRES



# Les sélecteurs de formulaires

## **:input**

Sélectionne tous les éléments de formulaires (input, textarea, select et boutons).

## **:text**

Sélectionne tous les éléments de formulaires de type ligne de texte.

## **:password**

Sélectionne tous les éléments de formulaires de type mot de passe.

## **:radio**

Sélectionne tous les éléments de formulaires de type boutons radio.

## **:checkbox**

Sélectionne tous les éléments de formulaires de type boutons de cases à cocher.

## **:submit**

Sélectionne tous les éléments de formulaires de type submit.

## **:image**

Sélectionne tous les éléments de formulaires de type image.

## **:reset**

Sélectionne tous les éléments de formulaires de type reset.

# Les sélecteurs de formulaires

## **:button**

Sélectionne toutes les balises <button> et tous les éléments de formulaires de type button.

## **:file**

Sélectionne tous les éléments de formulaires de type fichier.

## **:hidden**

Sélectionne tous les éléments de formulaires de type hidden.

JQuery

# LES FILTRES

# Les filtres JQuery de base

Tous les éléments du DOM étant répertoriés par jQuery, il devient facile de filtrer des éléments déterminés comme le premier, le dernier, etc...

## 1. Le premier élément

**:first**

Sélectionne la première instance d'un élément.

`$("li:first")` : sélectionne le premier élément de liste `<li>` du document.

### Remarque :

Le sélecteur `:first` sélectionne un seul élément tandis que `:first-child` peut en sélectionner plusieurs, soit un pour chaque parent.

Exemple [firstElement.html](#)

1	2	3
4	5	6
7	8	9
10	11	12

# Les filtres JQuery de base

## 2. Le dernier élément

**:last**

Sélectionne la dernière instance d'un élément.

`$("li:last")` : sélectionne le dernier élément de liste `<li>` du document.

Exemple [last.html](#)

1	2	3
4	5	6
7	8	9
10	11	12

# Les filtres JQuery de base

## 3. Les éléments pairs

**:even**

Sélectionne les éléments pairs selon un index commençant à 0.

`$("tr:even")` : sélectionne les lignes d'index JavaScript 0, 2, 4 soit les lignes 1, 3, 5 à l'écran.

Exemple [even.html](#)

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

# Les filtres JQuery de base

## 4. Les éléments impairs

**:odd**

Sélectionne les éléments impairs selon un index commençant à 0.

`$("tr:odd")` : sélectionne les cellules d'index JavaScript 1, 3, 5

Exemple [odd.html](#)

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

# Les filtres JQuery de base

## 5. Un élément déterminé

### **:eq(index)**

Sélectionne l'élément déterminé par la valeur de l'index.

Comme les index JavaScript débutent à zéro, le sélecteur :eq(0) sélectionne donc le premier élément, :eq(1) le second élément et ainsi de suite.

\$("#td:eq(2)") : sélectionne la troisième cellule d'un tableau.

Exemple [eq.html](#)

- Item de liste 1
- Item de liste 2
- Item de liste 3
- Item de liste 4
- Item de liste 5



# Les filtres JQuery de base

## 6. Les éléments suivants

**:gt(index)**

Sélectionne les éléments avec une valeur d'index supérieure (greater than) à la valeur fournie en paramètre.

`$("a:gt(1)")` : sélectionne tous les liens de la page en commençant par le troisième (soit après le second élément).

Exemple [gt.html](#)

- Item de liste 1
- Item de liste 2
- Item de liste 3
- Item de liste 4
- Item de liste 5

# Les filtres JQuery de base

## 7. Les éléments précédents

**:lt(index)**

Sélectionne les éléments avec une valeur d'index inférieure (less than) à la valeur fournie en paramètre.

`$("p:lt(3)")` : sélectionne tous les paragraphes situés avant le quatrième, soit les trois premiers paragraphes.

Exemple [lt.html](#)

- Item de liste 1
- Item de liste 2
- Item de liste 3
- Item de liste 4
- Item de liste 5

# Les filtres JQuery de base

## 8. La balise de titre

### **:header**

Retourne tous les éléments qui sont des balises de titre comme <h1>, <h2>, <h3>, etc...

`$(":header")` : sélectionne toutes les balises de titre de la page.

Exemple [header.html](#)

<b>Titre de niveau 1</b>
Contenu
<b>Titre de niveau 2</b>
Contenu
<b>Titre de niveau 3</b>
Contenu

# Les filtres JQuery de base

## 9. Exclusion d'un élément

### **:not(sélecteur)**

Exclut de la sélection tous les éléments qui répondent au critère spécifié par le sélecteur.

`$("div:not(#box)")` : sélectionne toutes les divisions sauf celle identifiée par box.

Exemple [not.html](#)

- Item de liste 1
- Item de liste 2
- Item de liste 3
- Item de liste 4
- Item de liste 5

# Les filtres JQuery de base

- **Les filtres enfants**

1. **Le premier enfant**

**:first-child**

Sélectionne tous les éléments qui sont le premier enfant de leur parent.

`$("ul:first-child")` : sélectionne le premier enfant (soit le premier élément de liste) de la liste non ordonnée `<ul>`.

2. **Le dernier enfant**

**:last-child**

Sélectionne tous les éléments qui sont le dernier enfant de leur parent.

`$("ul:last-child")` : sélectionne le dernier enfant (soit le dernier élément de liste) de la liste non ordonnée `<ul>`.

# Les filtres JQuery de base

## 3. Le énième enfant

### **:nthchild(index)**

Sélectionne les éléments qui sont le énième enfant de leur parent. La position est fournie par le paramètre index.

#### Remarque :

Au contraire de nombreux index en jQuery, l'index ne débute pas ici à 0 mais à 1.

`$("ul li:nth-child(2)")` : sélectionne le second élément `<li>` de la liste `<ul>`.

Revenons à cette fameuse exception concernant l'index.

La plupart des index utilisés par jQuery font appel à des fonctions natives de JavaScript et respectent la convention de débiter les index à 0.

Le sélecteur `:nth-child`, sélecteur spécifique à jQuery, est strictement dérivé des spécifications CSS. La valeur d'index 1 signifie donc bien qu'il s'agit du premier élément.

Exemple [nthchild.html](#)

```
<script type="text/javascript">
$(document).ready(function(){
  $("li:nth-child(2)").css("background", "#9cf");
});
</script>
<style type="text/css">
```

- Item de liste 1
- Item de liste 2
- Item de liste 3
- Item de liste 4
- Item de liste 5

# Les filtres JQuery de base

## 4. Les enfant pairs et impairs

Variante du sélecteur précédent. Le sélecteur nth-child peut sélectionner les éléments pairs et impairs.

**:nthchild(even ou odd)**

Sélectionne les éléments qui sont les nièmes enfants pairs (even) ou impairs (odd) de leur parent.

Comme :nth-child dont il est une variante, l'index débute à 1.

\$("#table tr:nth-child(odd)") : sélectionne les lignes impaires <tr> du tableau.

Exemple [enfantpairimpair.html](#)

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

# Les filtres JQuery de base

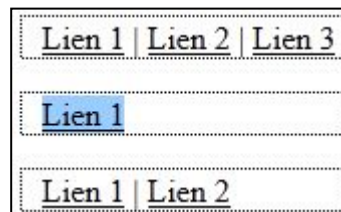
## 5. Les enfant uniques

### :only-child

Sélectionne tous les éléments qui sont enfant unique de leur parent. Si le parent a plusieurs enfants, aucune sélection n'est effectuée.

`$("div button:only-child")` : retrouve les boutons (balise `<button>`) qui n'ont pas de frère(s) dans toutes les div rencontrées.

Exemple [enfantunique.html](#)





# Les filtres JQuery de base

- **Les filtres de contenu**

1. **Un texte donné**

**:contains(texte)**

Sélectionne les éléments qui contiennent un texte ou un fragment de texte fourni en argument.

`$("div:contains('Eni')")` : sélectionne les divisions qui contiennent le texte "Eni".

**Remarque**

l'argument texte est sensible à la casse (*case sensitive*).

Exemple [contains.html](#)

Entreprise Cognitic Sprl
Spécialiste en informatique
En Belgique

# Les filtres JQuery de base

## 2. Un contenu vide

### **:empty**

Sélectionne tous les éléments qui n'ont pas d'enfants (inclus les nœuds de texte).

`$("div:empty")` : sélectionne les div vides.

Exemple [empty.html](#)

1	2	
	5	
7		9

## 3. Le parent

### **:parent**

Sélectionne les éléments qui sont parent, c'est-à-dire qui ont des éléments enfants, inclus les nœuds de texte.

`$("div:parent")` : sélectionne les divisions qui ont des éléments enfants.

Exemple [parent.html](#)

1	2	
	5	
7		9

# Les filtres JQuery de base

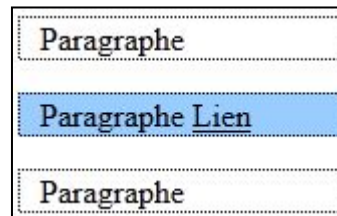
## 4. Un sélecteur particulier

### :has(sélecteur)

Sélectionne les éléments qui contiennent le sélecteur transmis en argument.

`$("div:has(p)")` : sélectionne les div qui contiennent un ou des paragraphe(s).

Exemple [has.html](#)



# Les filtres JQuery de base

- **Les filtres d'attribut**

Les attributs sont nombreux dans le langage Html et Xhtml. (*title, alt, src, href, width, style, etc...*)

## 1. L'attribut

### [attribut]

Sélectionne les éléments dotés de l'attribut spécifié.

`$("div[id]")` : sélectionne les éléments qui ont un attribut id.

Exemple [attribut.html](#)



# Les filtres JQuery de base

## 2. L'attribut avec valeur

### [attribut=valeur]

Sélectionne les éléments dotés d'un attribut avec une valeur déterminée.

`$("input[name='newsletter']")` : sélectionne l'élément de formulaire `<input>` avec un attribut `name="newsletter"`.

Exemple [attributvaleur.html](#)

- *Item de liste 1*
- *Item de liste 2*
- **Item de liste 3**
- *Item de liste 4*
- **Item de liste 5**

## 3. L'attribut qui n'a pas une certaine valeur

### [attribut!=valeur]

Sélectionne les éléments qui n'ont pas l'attribut spécifié et ceux qui ont l'attribut spécifié mais pas avec la valeur indiquée. **Valeur est case sensitive.**

`$("input[name!=newsletter]")` : sélectionne les éléments de formulaire `<input>` qui n'ont pas d'attribut `name` et ceux qui ont un attribut `name` avec une autre valeur que `newsletter`.

Exemple [attributnotvaleur.html](#)

- Lien 1
- Lien 2
- Lien 3
- Lien 4
- Lien 5

# Les filtres JQuery de base

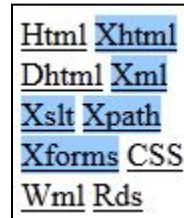
## 4. L'attribut dont la valeur commence par

**[attribut^=valeur]**

Sélectionne les éléments qui possèdent l'attribut spécifié et dont la valeur commence par les caractères indiqués.

`$("input[name^='news']")` : sélectionne les éléments de formulaire `<input>` avec l'attribut `name` et dont la valeur de celui-ci commence par les caractères "news".

Exemple [commencevaleur.html](#)



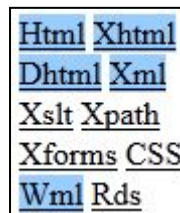
## 5. L'attribut dont la valeur finit par

**[attribut\$=valeur]**

Sélectionne les éléments qui possèdent l'attribut spécifié et dont la valeur se termine par les caractères indiqués.

`$("input[name$='letter']")` : sélectionne les éléments de formulaire `<input>` avec l'attribut `name` et dont la valeur de celui-ci se termine par les caractères "letter".

Exemple [finitvaleur.html](#)



# Les filtres JQuery de base

## 6. L'attribut dont la valeur contient

**[attribut\*=valeur]**

Sélectionne les éléments qui possèdent l'attribut spécifié et dont la valeur comporte les caractères indiqués.

`$("input[name*='slet']")` : sélectionne les éléments de formulaire `<input>` avec l'attribut `name` et dont la valeur de celui-ci comporte les caractères "slet".

Exemple [contientvaleur.html](#)

<u>Html</u>	<u>Xhtml</u>
<u>Dhtml</u>	<u>Xml</u>
<u>Xslt</u>	<u>Xpath</u>
<u>Xforms</u>	<u>CSS</u>
<u>Wml</u>	<u>Rds</u>

# La recherche d'éléments

## A. Trouver une séquence d'éléments

**slice(position de départ,[position de fin])**

Extrait une séquence parmi les éléments de la recherche.


- ✓ Position de départ (entier) : indique la position du premier élément de la séquence. Cet entier peut être négatif. Dans ce cas, la sélection débute à partir de la fin de la sélection initiale.
- ✓ Position de fin (entier) (facultatif) : indique la position (non comprise, strictement inférieur) du dernier élément de la séquence.

### Remarque

L'index des positions commence à 0.

Exemple [slice.html](#)

Trouver les cellules de 4 à 9		
1	2	3
4	5	6
7	8	9
10	11	12



Trouver les cellules de 4 à 9		
1	2	3
4	5	6
7	8	9
10	11	12



# La recherche d'éléments

## B. Trouver un élément selon un critère

### **is(expression)**

Indique si la sélection répond à un critère déterminé par l'expression. Renvoie true ou false.

✓ expression (chaîne de caractères) : expression correspondant au critère à vérifier.

Cette méthode est très pratique pour vérifier si un checkbox est coché

□ `$("#chkImportant").is(":checked")`

Ou encore pour vérifier si la cible d'un événement est un élément particulier

□ `$(event.target).is("li")`

Exemple [is.html](#)

# Former un tableau d'éléments

## □ **map(fonction de rappel)**

Renvoie un tableau d'éléments (Array) résultant d'une action sur un ensemble d'éléments.

Chaque ligne du tableau est le retour de la fonction appliquée à un élément.

✓ fonction de rappel (callback) : fonction appliquée aux éléments ciblés.

Exemple [map.html](#)

### **Remarque**

**Dans l'exemple, nous utilisons également les méthodes :**

**Get** □ permet d'accéder à tous les éléments du formulaires.

**Join** □ convertit le tableau Array en une chaîne de caractères composée de tous les éléments

# Exercice

- **Exercice 1**

Soit une liste de variétés de fruits. Au choix d'un élément de formulaire de type <select>, le script ne retiendra que les variétés d'un fruit spécifique (pomme, poire, raisin ou fraise).



Sélectionnez la catégorie : Toutes ▼

Filtrer la liste

- Arine
- Perlette
- Marjolaine
- Alexandrine
- Charlotte
- Violette
- Choupette
- Cardinal
- Mamie
- Pérouille
- Calmeria

The image shows a web form with a dropdown menu labeled 'Sélectionnez la catégorie :' and a button labeled 'Filtrer la liste'. Below the button is a list of ten fruit varieties, each preceded by a bullet point. The varieties are: Arine, Perlette, Marjolaine, Alexandrine, Charlotte, Violette, Choupette, Cardinal, Mamie, Pérouille, and Calmeria.

JQuery

# **LA MODIFICATION DES ÉLÉMENTS**

# La modification des éléments

Jquery ne se contente pas de permettre la sélection d'élément mais permet également de modifier ceux-ci dynamiquement

## A. Ajouter ou supprimer une classe

### □ **addClass(classe)**

Ajoute la classe spécifiée à tous les éléments sélectionnés.

`$("p:last").addClass("selected")` : ajoute la classe selected au dernier paragraphe.

Cette méthode retourne un objet jQuery.

#### Remarque

Cette méthode ne remplace pas une classe, elle ajoute simplement une classe.

Il est possible d'ajouter plus d'une classe à la fois. Elles sont notées les unes à la suite des autres, séparées par un espace □ `addClass(classe1 classe2 classe3)`.

# La modification des éléments

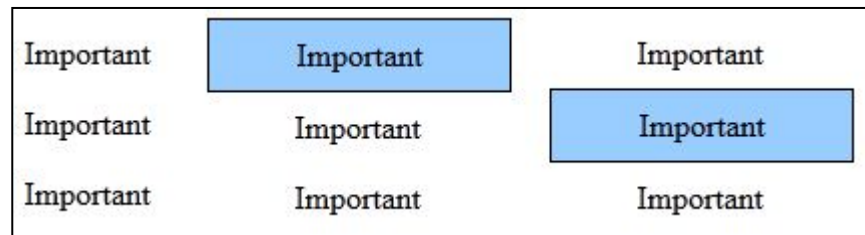
## □ **removeClass(classe)**

Supprime la classe spécifiée à tous les éléments sélectionnés.

`$("#p:last").removeClass("selected")` : supprime la classe selected au dernier paragraphe.

Exemple [addRemoveClass.html](#)

```
<script type="text/javascript">
$(document).ready(function() {
    $("div").mouseover(function() {$(this).addClass("arriereplan");});
    $("div").mouseout(function() {$(this).removeClass("arriereplan");});
});
</script>
```



Si nous désirons vérifier l'existence d'une classe □ `:hasClass(class)`

# La modification des éléments

## B. Basculer entre deux classes

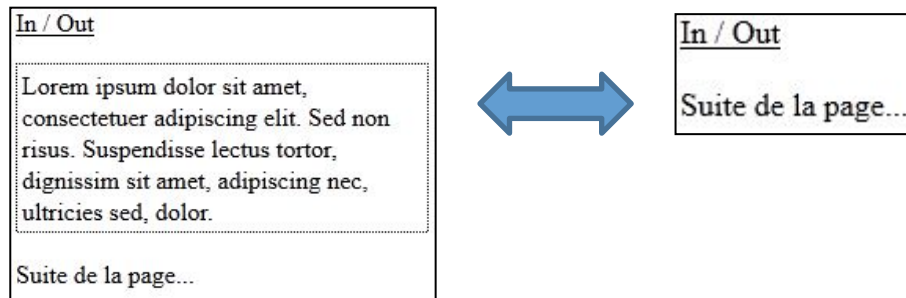
La librairie jQuery propose plusieurs méthodes qui permettent de déclencher tantôt une action, tantôt une autre. Cet effet de permutation est repris sous le terme de *toggle*.

### □ **toggleClass(classe)**

Ajoute la classe spécifiée si elle n'est pas présente, retire la classe spécifiée si elle est présente.

`$(p).toggleClass("classe1")` : applique la classe classe1 aux paragraphes si elle n'est pas présente. Si elle l'est, enlève la classe classe1.

Exemple [toggleClass.html](#)



# La modification des éléments

## C. Récupérer la valeur d'un attribut

Cette méthode jQuery correspond à `getAttribute()` du JavaScript classique.

**attr(nom de l'attribut)**

Accède à la valeur de l'attribut mentionné.

Cette méthode est assez utile pour retrouver la valeur d'un attribut de l'élément sélectionné ou du premier élément sélectionné s'il y en a plusieurs. `$("a").attr("title")` : récupère la valeur de l'attribut `title` du premier lien rencontré.

### Remarque

Si l'élément n'a pas d'attribut répondant à ce nom, la valeur *undefined* est retournée.

Exemple [getAttr.html](#)

jQuery est un framework JavaScript libre.

Le style du `<span>` est :



jQuery est un framework JavaScript libre.

Le style du `<span>` est :

fontstyle: italic



# La modification des éléments

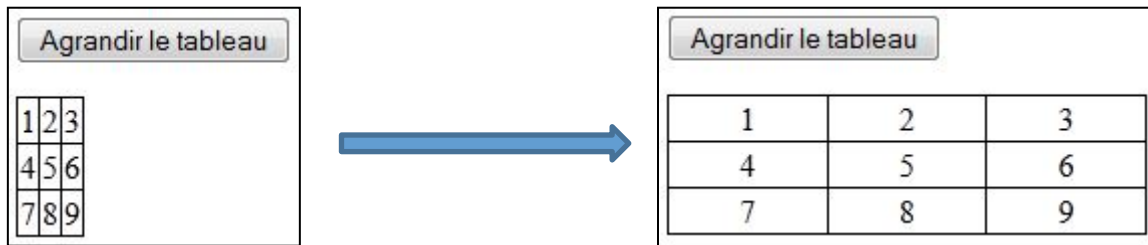
## D. Ajouter un attribut et sa valeur

**attr(attribut, valeur)**

Assigne une paire attribut/valeur à tous les éléments concernés.

`$("#photo").attr("alt", "Parc éoliennes")` : assigne à l'élément identifié par `#photo`, l'attribut `alt="Parc éoliennes"`.

Exemple [AddValAttr.html](#)



# La modification des éléments

## E. Ajouter plusieurs attributs et leurs valeurs

### **attr({propriétés})**

Permet d'assigner un ensemble de paires attribut/valeur aux éléments sélectionnés.

Les différentes propriétés sont séparées par une virgule.

`$("img").attr({ src: "hat.gif", alt: "Logo jQuery!" })` : assigne les attributs src et alt aux images.

Exemple [attrprop.html](#)



Pour supprimer un attribut ☐ **removeAttr(nom de l'attribut)**

# La modification des éléments

## F. Ajouter des éléments à la sélection

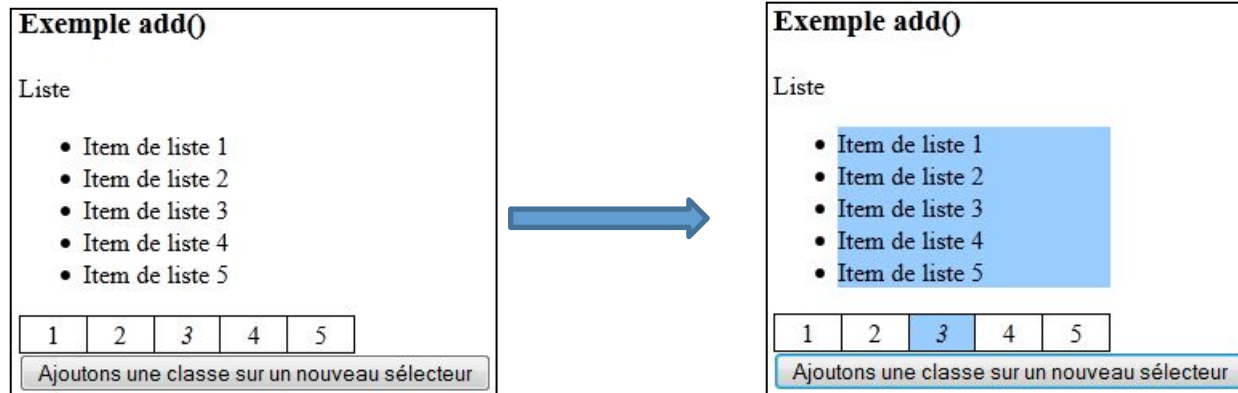
### **add(sélecteur ou élément(s) ou Html)**

Ajoute des éléments, fournis en argument, à l'ensemble des éléments sélectionnés dans la recherche.

Les paramètres peuvent être :

- ✓ un sélecteur jQuery : `$("p").add("span")` ;
- ✓ un ou des éléments : `$("p").add(document.getElementById("a"))` ;
- ✓ du code Html : `$("p").add("<span>Again</span>")`

Exemple [add.html](#)



JQuery

# LES FEUILLES DE STYLES

# Les feuilles de style et JQuery

JQuery permet la modification dynamique des propriétés de style CSS de façon très simple

## 1. La méthode CSS

Il existe trois façons d'utiliser cette méthode

### A. **css(nom)**

Permet d'accéder à une propriété de style du premier élément trouvé.

Le nom est une chaîne de caractères qui reprend la propriété de style à accéder.

Exemple [CssAccess.html](#)

#### **Remarque :**

Depuis la version 1.3, la couleur des bordures doivent être récupérées indépendamment

`$(this).css("border-left-color");` ☐ ok

`$(this).css("border-color");` ☐ ko

# Les feuilles de style et JQuery

## B. `css({propriété de style})`

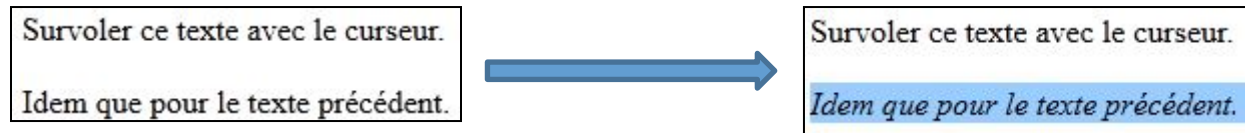
Modifie les propriétés de style d'un élément donné en utilisant la notation CSS clé/valeur pour les propriétés de style à transformer.

```
$("p").css({ color: "red", background: "blue" });
```

Il faut remarquer la présence des accolades habituelles pour les déclarations de style.

La notation JavaScript (CamelCase) peut également être adoptée soit *backgroundColor* au lieu de *background-color*.

Exemple [CssMod.html](#)



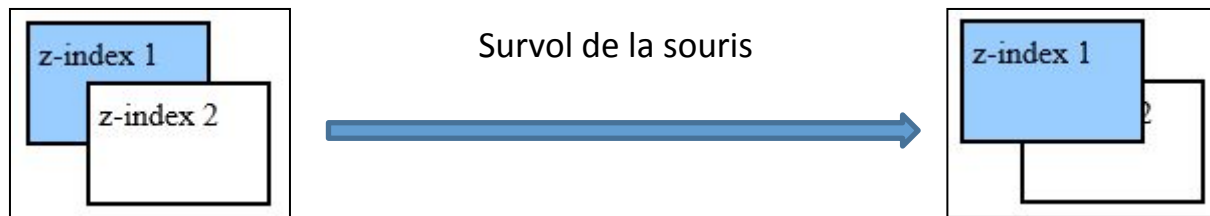
# Les feuilles de style et JQuery

## C. `css(clé,valeur)`

Modifie les propriétés de style d'un élément donné en utilisant la notation clé/valeur pour les propriétés de style à transformer.

- clé (chaîne de caractère) correspond au nom de la propriété de style à modifier.
- valeur (chaîne de caractère ou nombre) est la nouvelle valeur de la propriété. Si un nombre est spécifié, jQuery le converti automatiquement en pixel.

Exemple [CssCleValeur.html](#)



JQuery

# **GESTIONNAIRE D'ÉVÉNEMENTS**



# Les gestionnaires d'événements

La librairie jQuery met à disposition des gestionnaires d'événements assez similaires à ceux du JavaScript traditionnel.

Ainsi au *onmouseover* de JavaScript correspond *mouseover* en jQuery. Le préfixe "on" a simplement disparu.

## □ **click()**

Associe une fonction à l'événement clic des éléments de la sélection.

Exemple :

```
$('#input').click(function(){...});
```

## □ **dblclick( )**

Associe une fonction à l'événement double clic des éléments de la sélection.

Exemple :

```
$('#input').dblclick(function(){...});
```

# Les gestionnaires d'événements

## □ **focus()**

Associe une fonction à l'événement au focus des éléments spécifiés.

Exemple :

```
$('#input').focus(function(){...});
```

## □ **blur()**

Déclenche l'événement qui se produit lorsque l'élément perd le focus. Cela a pour effet de déclencher toutes les fonctions associées à cet événement pour les éléments sélectionnés..

Exemple :

```
$('#input').blur();
```

## □ **scroll()**

Associe une fonction à l'utilisation de la barre de défilement d'un élément.

Exemple :

```
$("#textarea").scroll(function () {...});
```

# Les gestionnaires d'événements

## ❑ **mousedown()**

Associe une fonction pour les éléments sélectionnés lorsque l'utilisateur presse une touche de la souris..

Exemple :

```
$('#input'). mousedown(function(){...});
```

## ❑ **mouseup()**

Associe une fonction pour les éléments de la sélection lorsque l'utilisateur relâche le bouton de la souris.

Exemple :

```
$('#input'). mouseup(function(){...});
```

## ❑ **mousemove()**

Associe une fonction lorsque l'utilisateur bouge le curseur de la souris.

Exemple :

```
$("#textarea").mousemove(function () {...});
```

### **Remarque**

Détecter le moindre mouvement du curseur est un événement intéressant mais il faut être conscient que cela entraîne une forte consommation des ressources du système

# Les gestionnaires d'événements

## □ **mouseover()**

Associe une fonction lorsque l'utilisateur positionne le curseur de la souris audessus d'un élément.

Exemple :

```
$('#input').mouseover(function(){...});
```

## □ **mouseout()**

Associe une action à l'événement lorsque le curseur de la souris quitte un élément..

Exemple : `$('#input').mouseout(function(){...});`

## □ **mouseenter()**

Exécute une fonction lorsque le pointeur entre dans un élément.

Exemple : `$("#textarea").mouseenter(function () {...});`

## □ **mouseleave()**

Déclenche une fonction lorsque le pointeur quitte un élément.

Exemple : `$("#textarea").mouseleave(function () {...});`

# Les gestionnaires d'événements

## Remarque

mouseover, hérité du JavaScript onmouseover, propage l'événement à la hiérarchie des objets de la page.  
(propagation des événements ou débordement des événements)

Au contraire, mouseenter évite cette propagation de l'événement.

Idem pour mouseout et mouseleave

[Exemple : propagation.html](#)

# Les gestionnaires d'événements

Il existe également des événements ciblés pour les formulaires

- **submit()**  
Déclenche l'événement d'envoi du formulaire.  
Exemple : `$("#form").submit();`
- **change()** : déclenche un événement lorsqu'un contrôle de formulaire est modifié, par exemple lorsqu'une case à cocher de formulaire est activée.
- **error()** : cet événement est déclenché lorsqu'une erreur survient dans le script.
- **keydown()**, **keyup()** et **keypress()** : déclenchent un événement lorsqu'une touche du clavier est pressée (vers le bas), lorsqu'une touche du clavier est relâchée (vers le haut) et lorsqu'un caractère est encodé.
- **load()** et **unload()** : se produit lorsque la page en cours est chargée ou quittée.
- **resize()** : associe un événement lorsque la taille d'un élément, généralement la fenêtre du navigateur, est modifiée.
- **select()** : se produit lorsque l'utilisateur sélectionne un texte (ou une partie de celui-ci). Parfois appliqué aux champs de formulaire du type ligne de texte ou zone de texte (textarea).

JQuery

# EVÉNEMENTS AVANCÉS

# Gestionnaires d'événements avancés

## 1. Lier un événement à un objet

La méthode *bind()* est plus flexible et plus puissante que les événements spécifiques comme *click()* ou *mouseover()*,

La méthode permet non seulement d'affecter un ou plusieurs événements à un objet jQuery sur lequel sera exécutée une fonction passée en paramètre mais également de transmettre des données à cette fonction.

Ainsi un *clic* sur un *lien* ou le *survol* d'une image peut attribuer des informations différentes au gestionnaire d'événement.

La fonction liée à l'événement pourra alors s'exécuter différemment selon le contexte fourni par les données.

### **bind(événement, [données], fonction)**

- événement (chaîne de caractères) : l'événement associé. Si plusieurs événements sont spécifiés, ils seront simplement séparés par un espace.
- données (optionnel) : les données éventuellement fournies à la fonction.
- fonction : le code à exécuter au déclenchement de l'événement.



# Gestionnaires d'événements avancés

## 2. Exécuter une fonction une seule fois

**one(événement,[données],fonction)**

Associe une fonction à un événement donné. La différence avec la fonction bind() est que la fonction associée à l'événement ne sera exécutée au maximum 1! pour chaque élément de la sélection.

- événement (chaîne de caractères) : type d'événement concerné.
- données [optionnel] : données supplémentaires à passer au gestionnaire d'événements.
- fonction : fonction à associer à l'événement

Exemple [one.html](#)



# Gestionnaires d'événements avancés

## 3. Déclencher un événement particulier

### **trigger(événement)**

Déclenche un événement particulier pour les éléments de la sélection. Cela va aussi déclencher l'action par défaut du navigateur pour ce type d'événement (si elle existe).

Par exemple, utiliser le type d'événement 'submit' dans la fonction va aussi déclencher l'envoi du formulaire par le navigateur.

Cette action par défaut peut être empêchée en retournant "false" dans une des fonctions associées à l'événement pour un des éléments de la sélection.

Exemple : [trigger.html](#)

# Gestionnaires d'événements avancés

## 4. Déléguer un événement

### **live(événement, fonction)**

Attache une fonction à un événement donné pour tous les éléments existants et futurs trouvés.

```
$("#p").live("click", function(){alert( $(this).text() );});
```

#### **Remarque**

jQuery fonctionne sur les éléments existant au chargement du DOM. Si un

événement génère un nouvel élément, votre fonction n'agira pas sur celui-ci

La méthode `live()` permet de contourner le problème et rappelle la fonction définie après son exécution.

Ce qui a pour conséquence de pouvoir agir sur les nouveaux éléments d'une page.

La méthode **die(événement, fonction)** annule `live()` en arrêtant la fonction définie en paramètre.

Exemple

# Exercices

- **Exercice 2**

**Redimensionner la taille des caractères**

Nous allons permettre à l'utilisateur de modifier, à son gré, la taille de la police de caractères pour un meilleur confort de lecture.

# Exercices

- **Exercise 3**



JQuery

# **EVÉNEMENTS PROPRES À JQUERY**

# Méthodes propres à JQuery

## □ **hover(fonction 1, fonction 2)**

La méthode `hover()` de jquery lie deux événements fréquemment utilisés à l'élément sélectionné ; l'un lorsque le pointeur survole et l'autre lorsqu'il le quitte.

Cette méthode s'effectue donc en deux temps. Quand le curseur se situe au-dessus d'un élément déterminé, la première fonction passée en paramètre est exécutée. Lorsque le curseur sort du cadre de l'élément, la seconde fonction est exécutée.

```
$("#p").hover(function(){
    $(this).addClass("hover");
},
function(){
    $(this).removeClass("hover");
});
```

Exemple [hover.html](#)

# Méthodes propres à JQuery

## □ **toggle (fonction 1, fonction 2, etc.)**

Permet de basculer d'une fonction à l'autre à chaque clic sur les éléments de la sélection.

Dès que l'événement clic est réalisé, la première fonction est exécutée. Au clic suivant, la seconde sera exécutée. Et

ainsi de suite.

```
$("#p").toggle(function(){
    $(this).addClass("selected");
},
function(){
    $(this).removeClass("selected");
});
```



JQuery

# **DIMENSIONS ET POSITIONS**

# Le dimensionnement

## □ **Height()**

Renvoie la hauteur, exprimée en pixels, d'un élément

## □ **Height(valeur)**

Assigne une hauteur aux éléments spécifiés. Si aucune unité n'est spécifiée (comme em ou %), l'unité px sera choisie par défaut.

## □ **Width()**

Renvoie la largeur, exprimée en pixels, d'un élément

## □ **Width(valeur)**

Assigne une largeur aux éléments spécifiés. Si aucune unité n'est spécifiée (comme em ou %), l'unité px sera choisie par défaut.

Ces méthodes nous permettent de récupérer les dimensions des div, span, images, etc...

Il existe encore d'autres méthodes permettant de nous renvoyer les informations concernant les dimensions intérieures/extérieures des objets

# Le dimensionnement

- **innerHeight()**  
retourne la hauteur intérieure (la bordure exclue mais le padding inclus) du premier élément trouvé dans la sélection
- **innerWidth()**  
récupère la largeur intérieure (la bordure exclue mais le padding inclus) du premier élément trouvé répondant à la sélection.
- **outerHeight()**  
retourne la hauteur extérieure (la bordure incluse et le padding par défaut) pour le premier élément trouvé répondant à la sélection
- **outerWidth()**  
retourne la largeur extérieure (inclut la bordure et le padding par défaut) pour le premier élément trouvé répondant à la sélection.

**Remarque :**

Ces méthodes fonctionnent pour les éléments visibles et non visibles.

Exemple [dimension.html](#)

# Le positionnement

- **position()**

Renvoie la valeur top et left de la position d'un élément relative à son élément parent

- **offset()**

Renvoie la valeur top et left de la position d'un élément relative au document.

- **scrollTop(valeur)**

Modifie le décalage (en pixels) entre le bord supérieur du document (top) et l'élément sélectionné, en prenant la valeur passée en argument.

- **scrollLeft(valeur)**

Modifie le décalage (en pixels) entre le bord gauche du document (left) et l'élément sélectionné, en prenant la valeur passée en argument.

Exemple [position.html](#)

# Les fonctions en JQuery

Afin de pouvoir réutiliser du code, il est important de pouvoir créer ses propres fonctions.

La syntaxe est assez simple

- ✓ `function mafonction(p1,p2){ //corps de la fonction nommée }`
- ✓ `function(p1,p2){//corps de la fonction anonyme }`

L'appel est également simple

- ✓ `mafonction() ;`
- ✓ `mafonction(p1);`

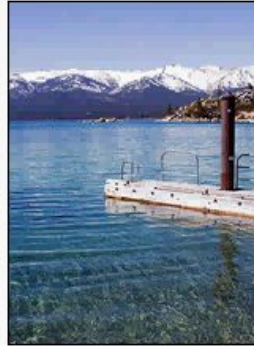
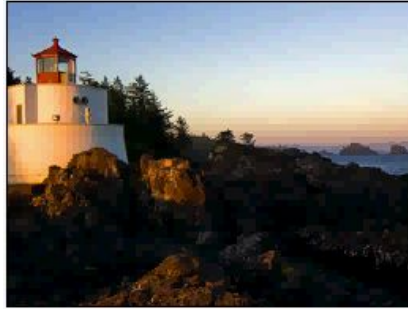
## Remarque :

`var a= mafonction ; // a contient la fonction (appel par a(p1) ;`  
`var a=mafonction() ; // a contient le résultat de l'appel de la fonction`

# Exercices

- **Exercice 4**

Effet de Zoom avec légende



# Exercices

- **Exercice 5**

jQuery est un framework JavaScript libre qui porte sur l'interaction entre le Html et le JavaScript (comprenant le DOM et l'AJAX ).

Source : Wikipedia

JQuery

# LES ANIMATIONS



# Les Animations

Les animations visuelles font partie prenante du JavaScript. Utilisées à bon escient et avec modération, elles peuvent donner plus de force à un élément du contenu. Le Web 2.0 les a d'ailleurs largement adoptées.

Le framework jQuery propose une série d'effets visuels faciles à encoder et compatible avec nos différents navigateurs.

Il est également possible de créer ses propres animations.

## A. Afficher et cacher

### **show(vitesse, fonction de rappel)**

Affiche un élément sélectionné (pour autant que celui-ci soit caché).

L'animation modifie dynamiquement la hauteur, la largeur et l'opacité de l'élément. Depuis la spécification jQuery 1.3, les marges externes et internes sont également modifiées pour obtenir un effet plus fluide.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

# Les Animations

## **hide(vitesse, fonction de rappel)**

Cache un élément sélectionné (pour autant que celui-ci soit visible).

L'animation modifie dynamiquement la hauteur, la largeur et l'opacité de l'élément. Depuis la spécification jQuery 1.3, les marges externes et internes sont également modifiées pour obtenir un effet plus fluide.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

Exemple [ShowHide.html](#)

# Les Animations

## B. Glisser verticalement

Les fonctions `slideDown()` et `slideUp()` permettent de jouer dynamiquement sur la hauteur d'un élément, généralement une division `<div> ... </div>`.

### **slideDown(vitesse, fonction de rappel)**

Fait glisser vers le bas (down) un élément sélectionné.

L'animation modifie seulement la hauteur.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

### **slideUp(vitesse, fonction de rappel)**

Fait glisser vers le haut (up) un élément sélectionné.

L'animation modifie seulement la hauteur.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

Exemple [SlideUpDown.html](#)

# Les Animations

## C. Les fondus

### **fadeIn(vitesse, fonction de rappel)**

Fait apparaître l'élément sélectionné selon un effet de fondu.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

### **fadeOut(vitesse, fonction de rappel)**

Fait disparaître l'élément sélectionné selon un effet de fondu.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

#### **Remarque**

Cette animation est obtenue en ajustant uniquement l'opacité. L'élément sélectionné doit ainsi déjà avoir une largeur et une hauteur spécifiée.

Exemple [FadeInFadeOut.html](#)

# Les Animations

## D. D'un glissé à l'autre

### **slideToggle(vitesse, fonction de rappel)**

Cette fonction fait glisser vers le bas un élément qui est en état "Up" et fait glisser vers le haut un élément qui est en état "Down".

L'animation modifie seulement la hauteur.

- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ fonction de rappel (callback) (optionnel) : fonction à exécuter à la fin de l'effet.

# Les Animations

## D. Créer ses propres animations

### **animate(paramètres, vitesse, easing, fonction de rappel)**

Chaque paramètre de l'objet représente une propriété sur laquelle portera l'animation (exemple: *height*, *top* ou *opacity*).

La valeur associée à la clé indique comment la propriété sera animée. Si la valeur est un nombre, le style de la propriété passera de sa valeur actuelle à la valeur spécifiée. Si la valeur *hide*, *show* ou *toggle* est spécifiée, une animation par défaut sera construite pour cette propriété.

- ✓ paramètres : conteneurs d'attributs de style que vous souhaitez animer et à quelle valeur.
- ✓ vitesse (optionnel) : chaîne de caractères représentant une des trois vitesses prédéfinies ('slow', 'normal' ou 'fast') ou le nombre en millisecondes correspondant à la durée de l'effet.
- ✓ easing (optionnel) : nom de l'effet customisé que vous souhaitez utiliser (plugin requis).
- ✓ fonction de rappel (optionnel) : fonction à exécuter à la fin de l'animation.

#### **Remarque**

Les propriétés devront être spécifiées selon la notation JavaScript (CamelCase) soit par exemple `marginLeft` au lieu de la notation CSS soit `margin-left`.

# Les Animations

## Exemples

```
animate( {fontSize:"24px", left:300, width: "200px", opacity: 0.5} , 1000 )
```

```
$("p").animate({ height: 'toggle', opacity: 'toggle' }, "slow");
```

[!\[\]\(9dfdaff1d86ba3c1f8353b4d1b61b8c5\_img.jpg\) monAnimation.html](#)

# Exercices

- **Exercice 6**

- Chapitre 1
- Chapitre 2
-  Chapitre 3
-  Chapitre 4
- Chapitre 5



# Exercices

- Exercice 7

Formation Web	▼
Formation OO	▼
Formation SharePoint	▼
Formation BD	▼

# Exercices

- **Exercice 8**

Pour cet exercice, vous aurez besoin du plugin se trouvant à l'adresse

<http://www.prototype.com/scripts/jquery/backgroundPosition/>



JQuery

# TRACAGE

# Tracage

En permettant au concepteur d'accéder à chacun des éléments de la page, le DOM a relancé le JavaScript. Mais il faut bien admettre que le traçage de parents à enfants et autres frères n'est pas toujours très aisé à mettre en place.

En outre, une simple modification entraîne souvent une réécriture complète du code. La librairie jQuery remédie grandement à ces inconvénients par ses nombreux sélecteurs et par des méthodes spécifiques pour traverser et manipuler les éléments du DOM.

## □ **children()**

Récupère un groupe d'éléments contenant les enfants immédiats de chacun des éléments concernés par la sélection.

## □ **parent()**

Récupère un groupe d'éléments contenant les parents immédiats de chacun des éléments concernés par la sélection

## □ **parents()**

Récupère un groupe d'éléments contenant tous les parents de chacun des éléments concernés par la sélection

# Tracage

- **siblings()**

Retourne la liste des frères immédiats de chaque élément de la sélection

- **prev()**

Retourne le frère immédiat précédent de chaque élément de la sélection.

- **prevAll()**

Retourne les frères immédiats précédents de chaque élément de la sélection.

- **next()**

Retourne le frère immédiat suivant de chaque élément de la sélection

- **nextAll()**

Retourne les frères immédiats suivants de chaque élément de la sélection.

# Tracage

## □ **contents()**

Trouve tous les nœuds enfants situés dans les éléments de la sélection (incluant les nœuds texte).

Si l'élément spécifié est une balise `<iframe>`, *contents()* trouve le contenu du document

## □ **closest(sélecteur)**

Retourne l'ensemble d'éléments contenant le parent le plus proche de l'élément sélectionné répondant au sélecteur, l'élément de départ inclus.

La méthode `closest()` vérifie d'abord si l'élément courant répond à l'expression spécifiée.

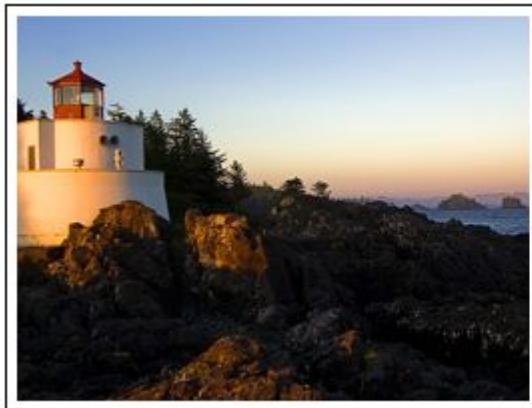
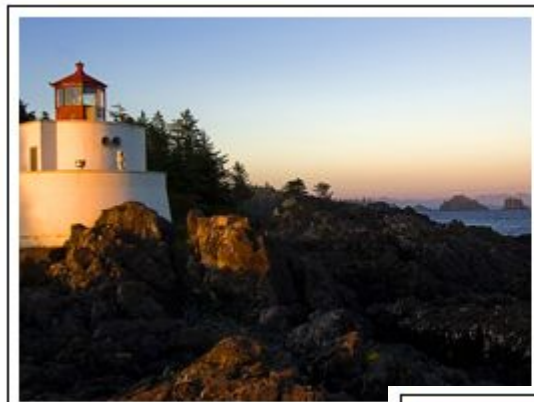
Dans l'affirmative, il retourne simplement l'élément spécifié. Sinon, il continue alors de traverser le document vers le haut, parent par parent, jusqu'à ce qu'il trouve un élément répondant à la condition de l'expression. Si aucun élément n'est trouvé, la méthode ne retourne rien.

# Tracage

## □ **find(expression ou sélecteur)**

Recherche les éléments descendants répondant aux conditions du sélecteur spécifié.

Exemple [loupe.html](#)



JQuery

# **MODIFIER/AJOUTER DES ÉLÉMENTS À LA VOLÉE**



# Modifier/Ajouter des éléments

## 1. Modifier le contenu

### □ **text()**

Récupère le contenu texte de l'élément concerné.

#### **Remarque :**

Cette méthode fonctionne pour les documents Html, Xhtml et XML.

### □ **text(valeur)**

Assigne un nouveau contenu texte (voir l'argument valeur) aux éléments concernés.

*`$("#div").text("les nouveaux éléments de texte")`*

insère au format texte, le contenu "les nouveaux éléments de texte" dans la balise <div>.

### □ **html()**

Récupère au format Html, le contenu de l'élément concerné.

#### **Remarque**

Cette méthode ne fonctionne pas pour les documents XML (à l'exception des documents Xhtml).

# Modifier/Ajouter des éléments

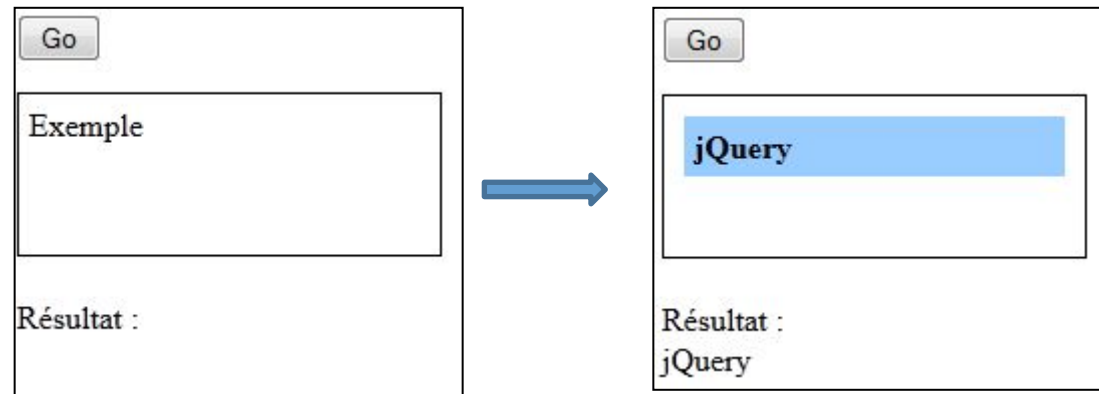
## □ `html(valeur)`

Assigne un nouveau contenu Html (voir l'argument valeur) aux éléments concernés.

### Remarque

1. Avec les méthodes `text(valeur)` et `html(valeur)`, le nouveau contenu écrase le contenu précédent.
2. Les méthodes `html()` de JQuery utilisent la propriété JavaScript `innerHTML`. Pour rappel, cette propriété, à l'origine propriétaire Internet Explorer, a depuis été adoptée par les autres navigateurs mais son **interprétation pose parfois des problèmes de compatibilité**.

Exemple [contenu.html](#)



# Modifier/Ajouter des éléments

## 2. Insérer à l'intérieur

### □ **append(contenu)**

Ajoute du contenu à la fin mais à l'intérieur de l'élément spécifié.

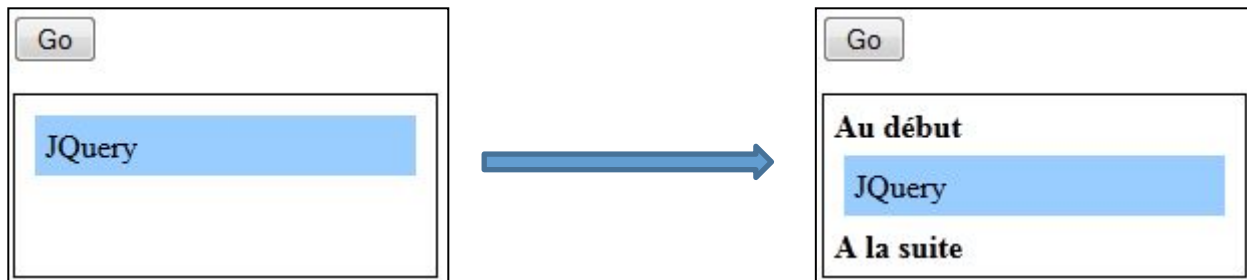
Le contenu peut être une chaîne de caractères, du Html ou un objet jQuery.

### □ **prepend(contenu)**

Ajoute du contenu au début mais à l'intérieur de l'élément spécifié.

Le contenu peut être une chaîne de caractères, du Html ou un objet jQuery

Exemple [insérerInterieur1.html](#)



# Modifier/Ajouter des éléments

## □ **appendTo()**

Ajoute des éléments spécifiés par le sélecteur A à la fin d'autres spécifiés par B selon l'expression `$(A).appendTo(B)`.

## □ **prependTo()**

Ajoute des éléments spécifiés par le sélecteur A au début d'autres spécifiés par B selon l'expression `$(A).prependTo(B)`

Exemple [insérerInterieur2.html](#)



# Modifier/Ajouter des éléments

## 3. Insérer à l'extérieur

### □ **after(contenu)**

Ajoute du contenu spécifié en argument après l'élément de la sélection..

Le contenu peut être une chaîne de caractères, du Html ou un objet jQuery.

### □ **before(contenu)**

Ajoute du contenu spécifié en argument avant chaque élément de la sélection.

Le contenu peut être une chaîne de caractères, du Html ou un objet jQuery

Exemple [insérerExterieur.html](#)



# Modifier/Ajouter des éléments

## 4. Entourer un élément

### **wrap (html ou élément)**

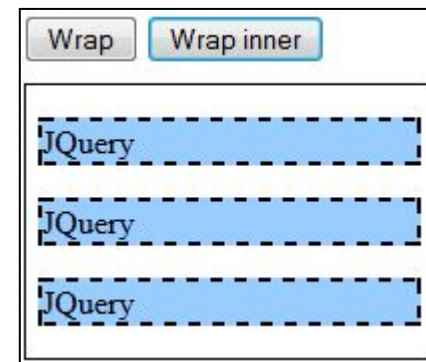
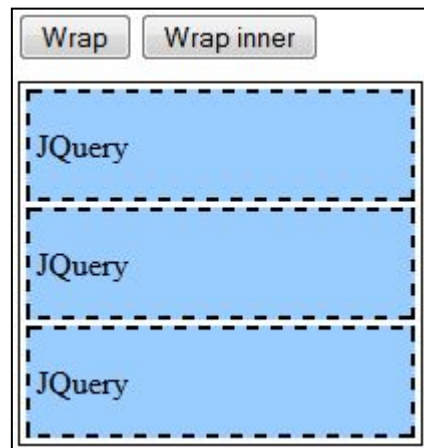
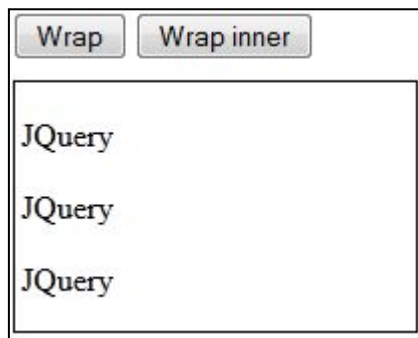
Entoure chaque élément sélectionné avec l'élément fourni en argument.

Cette procédure est très utile pour injecter une structure de code additionnelle dans un document sans modifier la sémantique originelle du document

### **wrapInner(html ou élément)**

Entoure les enfants d'un élément (les nœuds textes inclus) par un autre élément.

Exemple [entourer.html](#)



# Modifier/Ajouter des éléments

## 5. Remplacer un élément

### □ `replaceWith()`

Remplace l'élément courant par le contenu spécifié, sous forme de code Html ou d'objet DOM

#### Remarque

La méthode `html()` remplace le contenu de l'élément tandis que `replaceWith()` remplace l'élément lui-même.

Exemple [replacewith.html](#)



# Modifier/Ajouter des éléments

## 6. Supprimer un élément

### `remove()`

Supprime de l'arbre du DOM, tous les éléments répondant aux critères de sélection.

#### Remarque

Cette méthode ne supprime pas les éléments de l'objet jQuery, ce qui permet une utilisation ultérieure de ces éléments même si ceux-ci ne figurent plus dans le document

Exemple [remove.html](#)





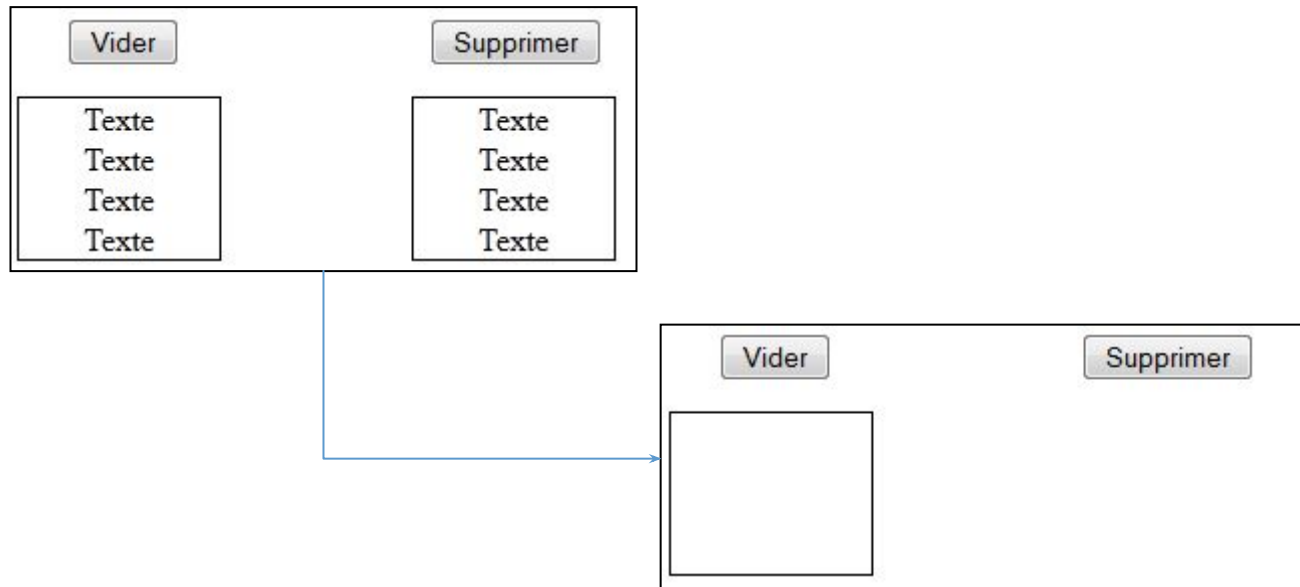
# Modifier/Ajouter des éléments

## 6. Vider un élément

### □ `empty()`

Supprime tous les noeuds enfants de l'élément sélectionné.

Exemple [methodeempty.html](#)



# Modifier/Ajouter des éléments

## 6. Copier un élément

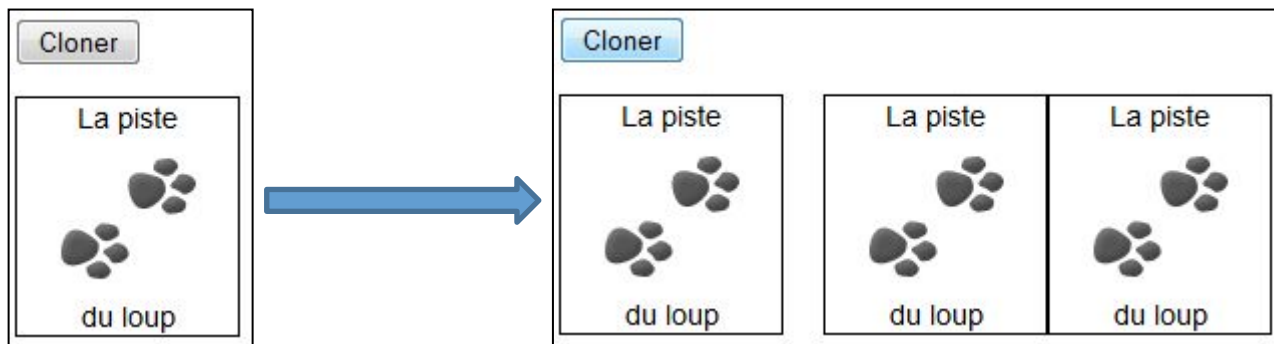
### □ `clone(paramètre)`

Copie (clone) les éléments DOM trouvés et les sélectionne.

Cette fonction est utile pour créer des copies d'éléments et les déplacer vers un autre endroit spécifié du DOM..

Paramètres (optionnel) : indiquez true si vous souhaitez cloner les gestionnaires d'événements associés à la sélection

Exemple [clone.html](#)



# Exercice

## 8. ToDo List

### Todo list

ToDo :  ☐ Importance haute

Ajouter

Reset

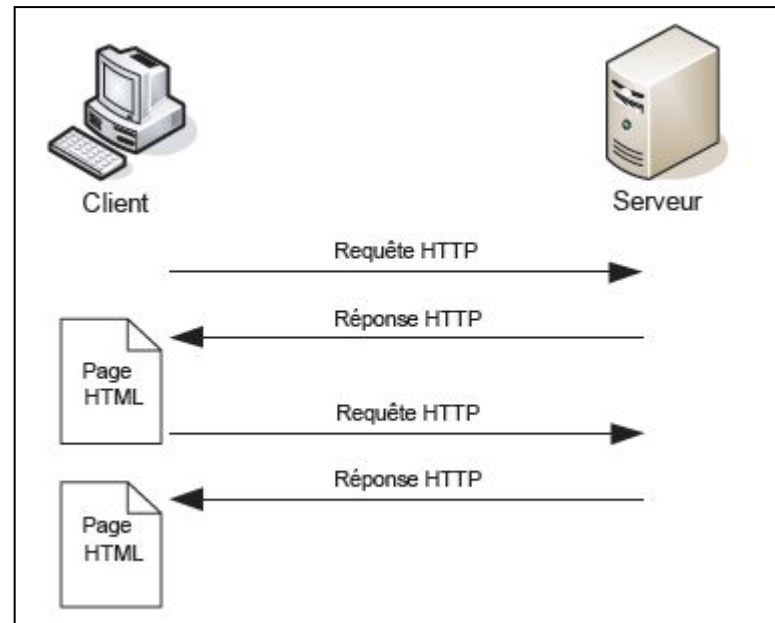
JQuery

**AJAX**

# Ajax

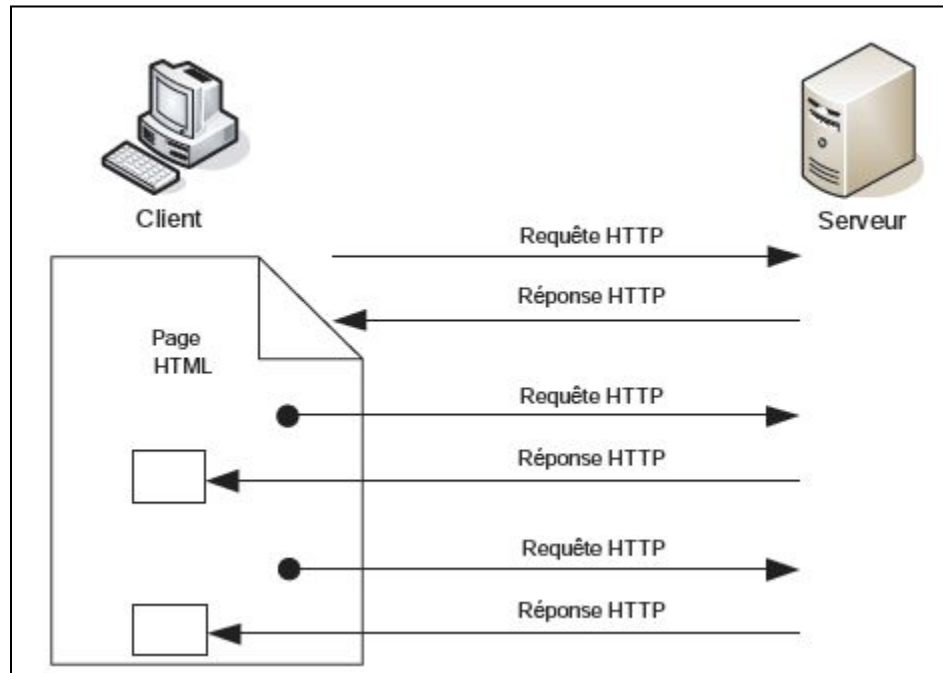
- **Qu'est-ce qu'Ajax ?**

Dans une application Web classique, lorsque l'utilisateur clique sur un lien ou valide un formulaire, le navigateur envoie une requête au serveur HTTP, qui lui retourne en réponse une nouvelle page, qui remplace purement et simplement la page courante



# Ajax

Avec Ajax, il est possible de ne mettre à jour qu'une partie de la page



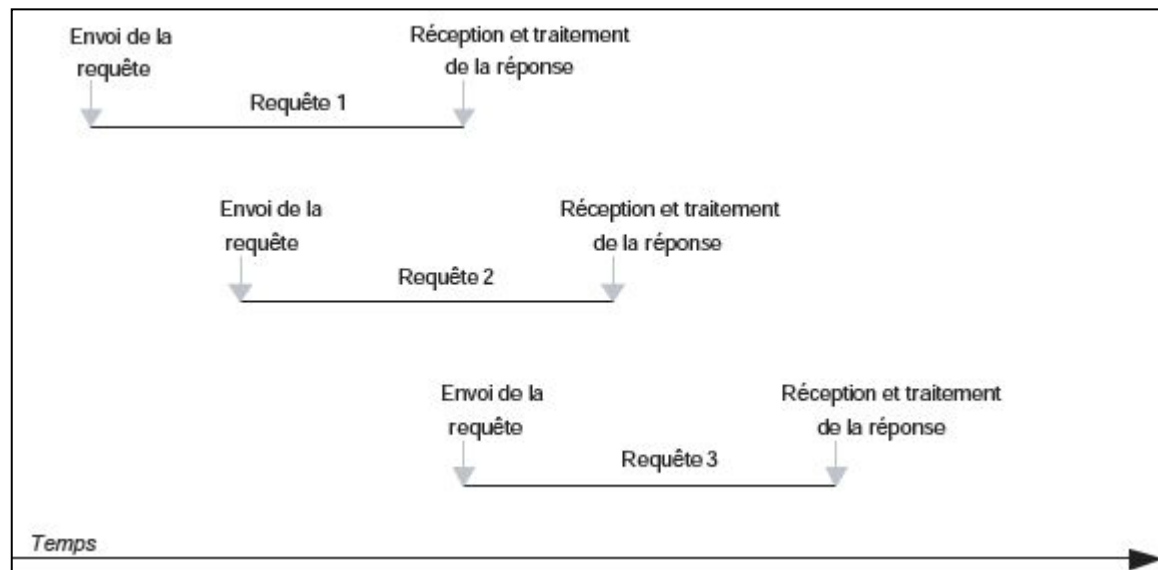
Les requêtes HTTP sont envoyées par une instruction JavaScript en réaction à une action de l'utilisateur. La réponse HTTP est également récupérée par JavaScript, qui peut dès lors mettre à jour la page courante, grâce à DOM et aux CSS, qui constituent ce qu'on appelle le HTML dynamique

# Ajax

- **Communication asynchrone avec le serveur**

La deuxième caractéristique d'Ajx est que la communication avec le serveur via JavaScript peut être asynchrone. La requête est envoyée au serveur sans attendre la réponse, le traitement à effectuer à la réception de celle-ci étant spécifié auparavant.

JavaScript se charge d'exécuter ce traitement quand la réponse arrive. L'utilisateur peut de la sorte continuer à interagir avec l'application, sans être bloqué par l'attente de la réponse, contrairement au Web classique. Cette caractéristique est aussi importante que la mise à jour partielle des pages.



# En résumé

Ajax est une technique qui fait usage des éléments suivants:

- HTML.
- CSS (Cascading Style-Sheet) pour la présentation de la page.
- JavaScript (EcmaScript) pour les traitements locaux, et DOM (Document Object Model) qui accède aux éléments de la page ou du formulaire ou aux éléments d'un fichier xml pris sur le serveur (avec la méthode getElementByTagName par exemple)...
  - L'objet XMLHttpRequest lit des données ou fichiers sur le serveur de façon asynchrone.
  - Si besoin, DOMparser intègre un document XML.
- PHP ou un autre langage de scripts peut être utilisé coté serveur.

Le terme "Asynchronous", asynchrone en français, signifie que l'exécution de JavaScript continue sans attendre la réponse du serveur qui sera traitée quand elle arrivera. Tandis qu'en mode synchrone, le navigateur serait gelé en attendant la réponse du serveur



# En résumé

- **Inconvénients d'Ajax**

1. Si JavaScript est désactivé, Ajax ne peut fonctionner. Il faut demander au lecteur de l'activer parmi les options du navigateur.
2. Si l'on charge les données à afficher de façon dynamique, elles ne font pas partie de la page et elles ne sont pas prises en compte par les moteurs de recherche.
3. L'aspect asynchrone fait que les modifications se font avec un délai (si le traitement sur le serveur est long), ce qui peut être déconcertant.

# AJAX – Mise en Œuvre -JS

- **En pure javascript**

Ajax utilise un modèle de programmation comprenant d'une part la présentation, d'autre part les événements.

Les fonctions JavaScript identifient les éléments de la page grâce au DOM et communiquent avec le serveur par l'objet XMLHttpRequest.

Pour recueillir des informations sur le serveur cet objet dispose de deux méthodes:

- open: établit une connexion.
- send: envoie une requête au serveur.

Les données fournies par le serveur seront récupérées :

- ✓ **responseXml**
- ✓ **responseText** de l'objet XMLHttpRequest

Il faut attendre la disponibilité des données, et l'état est donné par l'attribut **readyState** de XMLHttpRequest.

Les états de **readyState** sont les suivants (seul le dernier est vraiment utile):

- 0: non initialisé.
- 1: connexion établie.
- 2: requête reçue.
- 3: réponse en cours.
- 4: terminé.

# AJAX – Mise en Œuvre -JS

- **L'objet XMLHttpRequest**

Il permet d'interagir avec le serveur, grâce à ses méthodes et ses attributs.

- **Attributs**

**readyState** le code d'état passe successivement de 0 à 4 qui signifie "prêt". **status** 200 est ok 404 si la page n'est pas trouvée.

**responseText** contient les données chargées dans une chaîne de caractères.

**responseXml** contient les données chargées sous forme xml, les méthodes de DOM servent à les extraire.

**onreadystatechange** propriété activée par un évènement de changement d'état. On lui assigne une fonction.

- **Méthodes**

- **open**(mode, url, boolean)

- ✓ mode: type de requête, GET ou POST

- ✓ url: l'endroit où trouver les données, un fichier avec son chemin sur le disque.

- ✓ boolean: true (asynchrone) / false (synchrone).

- en option on peut ajouter un login et un mot de passe.

- **send**("chaîne") null pour une commande GET.

# AJAX – Mise en Œuvre -JS

- **Exemple de mise en place**

- Etape 1 :

```
if (window.XMLHttpRequest) // Objet de la fenêtre courant
{
    xhr = new XMLHttpRequest(); // Firefox, Safari, ...
}
else if (window.ActiveXObject) // Version Active
{
    xhr = new ActiveXObject("Microsoft.XMLHTTP"); // Internet Explorer
}
```

- Etape 2 : Attendre la réponse

```
if (xhr.readyState == 4) { // Reçu, OK } else { // Attendre... }
```

- Etape 3 : Effectuer la requête

```
xhr.open('GET', 'http://www.xul.fr/fichier.xml', true);
xhr.send(null);
```

# Ajax - JQuery

## 1. Les requêtes AJAX raccourcies

La méthode load() permet de charger d'une façon extrêmement simple, un fichier selon le procédé mis en place par Ajax.

### A. **load( url[, données],[, fonction])**

Charge le code Html (ou Xhtml) à partir d'un fichier donné et place celui-ci dans l'élément sélectionné.

- ✓ url : une chaîne de caractères contenant l'URL du fichier Html à charger.
- ✓ données (optionnel) : liste de paires de la forme clé/valeur qui seront envoyées en tant que données au serveur.
- ✓ fonction (optionnel) : la fonction qui doit être exécutée en cas de réussite de la requête. Par défaut, les données sont chargées dans l'élément sélectionné.

Exemple [load.html](#)

# Ajax - JQuery

## B. Charger qu'en cas de modification

### **loadIfModified( url[, données,][, fonction])**

Charge le code Html à partir d'un fichier donné et place celui-ci dans l'élément sélectionné s'il a été modifié depuis la dernière requête.

- ✓ url : une chaîne de caractères contenant l'URL du fichier Html à charger.
- ✓ données (optionnel) : liste de paires de la forme clé/valeur qui seront envoyées en tant que données au serveur.
- ✓ fonction (optionnel) : la fonction qui doit être exécutée en cas de réussite de la requête. Par défaut, les données sont chargées dans l'élément sélectionné.

### **Exemple**

`$("div").loadIfModified("test.htm")` : charge les données du fichier test.htm et ne place celles-ci dans la division `<div>` que si le fichier a subi des changements depuis la dernière requête.

# Ajax - JQuery

## C. Charger un script

**`$.getScript(url[, fonction])`**

Charge un script JavaScript du serveur en utilisant la méthode HTTP GET et exécute celui-ci.

- ✓ url : une chaîne de caractères qui indique l'adresse du script à charger.
- ✓ fonction (optionnel) : une fonction à exécuter si la requête est réussie

Exemple [getScript.html](#)

# Ajax - JQuery

## 2. La requête AJAX complète

Cette méthode permet d'effectuer une requête AJAX en maîtrisant, grâce aux nombreuses options disponibles, les différents paramètres et étapes de celle-ci.

### **ajax(options)**

Réalise une requête HTTP asynchrone (AJAX).

```
$.ajax({  
    url: "test.htm",  
    success: function(data )  
    {  
        $("#resultat").html(data);  
        $.log("Terminé");  
    },...  
});
```



# Ajax - JQuery

- ✓ **url** (obligatoire): une chaîne de caractères contenant l'adresse de la requête.
- ✓ **type** (optionnel) : une chaîne de caractères qui définit la méthode HTTP à utiliser pour la requête (GET ou POST). La valeur par défaut est GET. D'autres méthodes d'envoi HTTP peuvent être utilisées, comme PUT ou DELETE, mais celles-ci ne sont pas supportées par tous les navigateurs.
- ✓ **dataType** (optionnel) : une chaîne de caractères qui spécifie le format des données qui seront renvoyées par le serveur (xml, html, json ou script). Si rien n'est spécifié, jQuery utilisera le type MIME pour déterminer le format adéquat soit responseXML ou ResponseText. Les types disponibles sont :
  - "xml" : retourne un document XML qui pourra être traité par jQuery.
  - "html" : retourne du code Html au format texte.
  - "script" : évalue la réponse en JavaScript et retourne cette dernière au format texte.
  - "json" : évalue la réponse en JSON et retourne un objet JavaScript.
- ✓ **ifModified** (optionnel) : une valeur booléenne qui indique que le serveur doit vérifier si les données retournées sont différentes de la dernière requête avant de renvoyer le fichier avec succès. Par défaut, cette option vaut false.
- ✓ **timeout** (optionnel) : nombre de millisecondes après lequel la requête est considérée comme non réussie.

# Ajax - JQuery

- ✓ **global** (optionnel) : une valeur booléenne qui permet le déclenchement du gestionnaire d'évènements global d'AJAX. Par défaut, la valeur est true. Avec une valeur false, les déclenchements d'évènements de type ajaxStart() ou ajaxStop() sont ignorés.
- ✓ **beforeSend** (optionnel) : une fonction qui doit être exécutée avant l'envoi de la requête. Ceci permet de modifier l'objet XMLHttpRequest avant qu'il soit envoyé pour spécifier, par exemple, des entêtes HTTP personnalisées (pdf, png,...)
- ✓ **error** (optionnel) : une fonction qui doit être appelée en cas d'échec de la requête. La fonction dispose de trois arguments : l'objet XMLHttpRequest, une chaîne de caractères décrivant le type d'erreur rencontré et un objet d'exception, dans le cas où ce dernier a été généré.
- ✓ **success** (optionnel) : fonction à appeler si la requête s'exécute avec succès. Un seul argument est passé en paramètre soit les données retournées par le serveur.
- ✓ **complete** (optionnel) : fonction à exécuter lorsque la requête se termine. La fonction dispose de deux arguments : l'objet XMLHttpRequest et une chaîne de caractères décrivant le type de succès de la requête.
- ✓ **data** (optionnel) : données à envoyer au serveur. L'objet doit être formé de paires de la forme clé/valeur. Les données sont converties en chaîne de caractères (si elles ne le sont pas déjà). Voir l'option processData ci-après pour empêcher ce processus automatique.

# Ajax - JQuery

- ✓ **processData** (optionnel) : valeur booléenne qui indique si les données de l'option data doivent être converties en chaîne de caractères. La valeur par défaut est true. Pour empêcher la conversion, passez cette option à false.
- ✓ **contentType** (optionnel) : chaîne de caractères contenant le MIME des données lorsque des données sont envoyées au serveur. Par défaut, le MIME application/xwwwformurlencoded est retenu.
- ✓ **async** (optionnel) : une valeur booléenne qui indique si la requête doit s'effectuer de façon asynchrone ou synchrone. La valeur par défaut pour une requête AJAX est bien entendu true.

Ces options sont les plus couramment utilisées.

Nous avons cependant encore bien d'autres options :

- ✓ **cache** (optionnel) : une valeur booléenne qui, lorsqu'elle est mise sur false, empêche la page chargée d'être mise dans le cache du navigateur.
- ✓ **password** (optionnel) : dans le cas où l'accès HTTP de la requête nécessite un mot de passe.
- ✓ **username** (optionnel) : dans le cas où l'accès HTTP de la requête nécessite un nom d'utilisateur (username).
- ✓ **scriptCharset** (optionnel) : force les requêtes du type script à être interprétées avec un charset particulier.
- ✓ **xhr** (optionnel) permet de créer l'ActiveXObject (Internet Explorer) ou le XMLHttpRequest (pour les autres).

# Ajax - JQuery

## □ Ajax() - Simple

```
$.ajax( {url: "test.html", });
```

□ Le seul paramètre nécessaire est l'url à interroger

### Remarque

\$.ajax au contraire du load() ne traite pas automatiquement le retour.

```
$("#bouton1").click(function() {  
    $("#div").load("./ajax/option1.html");  
});
```

```
$("#bouton2").click(function() {  
    $.ajax({  
        url: "./ajax/option2.html",  
        success:  
            function(data, textStatus, XMLHttpRequest)  
            {  
                $("#div").append(XMLHttpRequest.responseText);  
            }  
    });  
});
```

Exemple [loadvsajax.html](#)

# Ajax - JQuery

## □ Ajax() – Envoyer des données via ajax

### **serialize()**

Transforme les données des champs de formulaire en une chaîne de caractères.

```
$("#form").serialize();
```

#### **Remarque**

Pour le bon fonctionnement de la méthode `serialize()`, les champs de formulaire doivent posséder un attribut `name`.

Exemple [serialize.html](#)

# Ajax - JQuery

## □ Un icone de chargement

Les requêtes AJAX peuvent parfois entraîner une relative lenteur au chargement du fichier externe en fonction de la taille du fichier, de l'encombrement du serveur et/ou de la qualité de la connexion.

Nous allons donc afficher un icone de chargement.

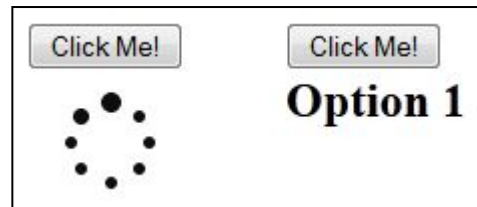
**Note** : vous pouvez trouver des icone de chargement sur <http://www.ajaxload.info/>



Deux méthodes s'offrent à nous :

### 1. Partage d'une même zone d'affichage

Exemple [wait.html](http://wait.html)



# Ajax - JQuery

## 2. Utiliser les événements Ajax ([evenementAjax.html](#))

Il existe plusieurs événements associé à la requête ajax pouvant être intercepté.

**A. `ajaxSend(fonction)`**

Assigne une fonction qui sera exécutée avant l'envoi de la requête AJAX.

**B. `ajaxStart(fonction)`**

Assigne une fonction à exécuter lorsqu'une requête AJAX débute.

**C. `ajaxStop(fonction)`**

Assigne une fonction qui sera exécutée à chaque fois qu'une requête se termine

**D. `ajaxSuccess(fonction)`**

Assigne une fonction qui sera exécutée à chaque fois qu'une requête se termine avec succès

**E. `ajaxComplete(fonction)`**

Assigne une fonction qui sera exécutée lorsque le processus total de la requête AJAX sera terminé.

**F. `ajaxError(fonction)`**

Assigne une fonction qui sera exécutée si la requête AJAX échoue.

# Ajax - Query

Il nous suffira donc d'afficher notre icone d'attente grâce à *ajaxSend(fonction)* et de le cacher grâce à *ajaxComplete(fonction)*

Exemple [wait2.html](#)

Grâce à ce principe, il nous est également possible d'interroger les différentes api (Flickr, google api, yahoo api, facebook api...)



# Exercice

## Exercice 10 : Flickr

Exécuter et étudier le code suivant :

```
<html><head>
  <script src="jquery.min.js"></script>
</head>
<body>
  <div id="images" style="height: 300px"></div>
  <script>$.getJSON("http://api.flickr.com/services/feeds/photos_public.gne?tags=besancon&tagmode=any&format=json&jsoncallback=?",
    function(data){
      $.each(data.items, function(i,item){
        $("<img/>").attr("src", item.media.m).appendTo("#images");
      });
    });</script>
</body></html>
```

Ensuite tenter de reproduire l'application suivante ☐

# Exercice

Flickr

Mot clé

Rechercher

**Remarque :**

Utiliser `ajax()` pour le chargement des photos( non `getJSON()`)