

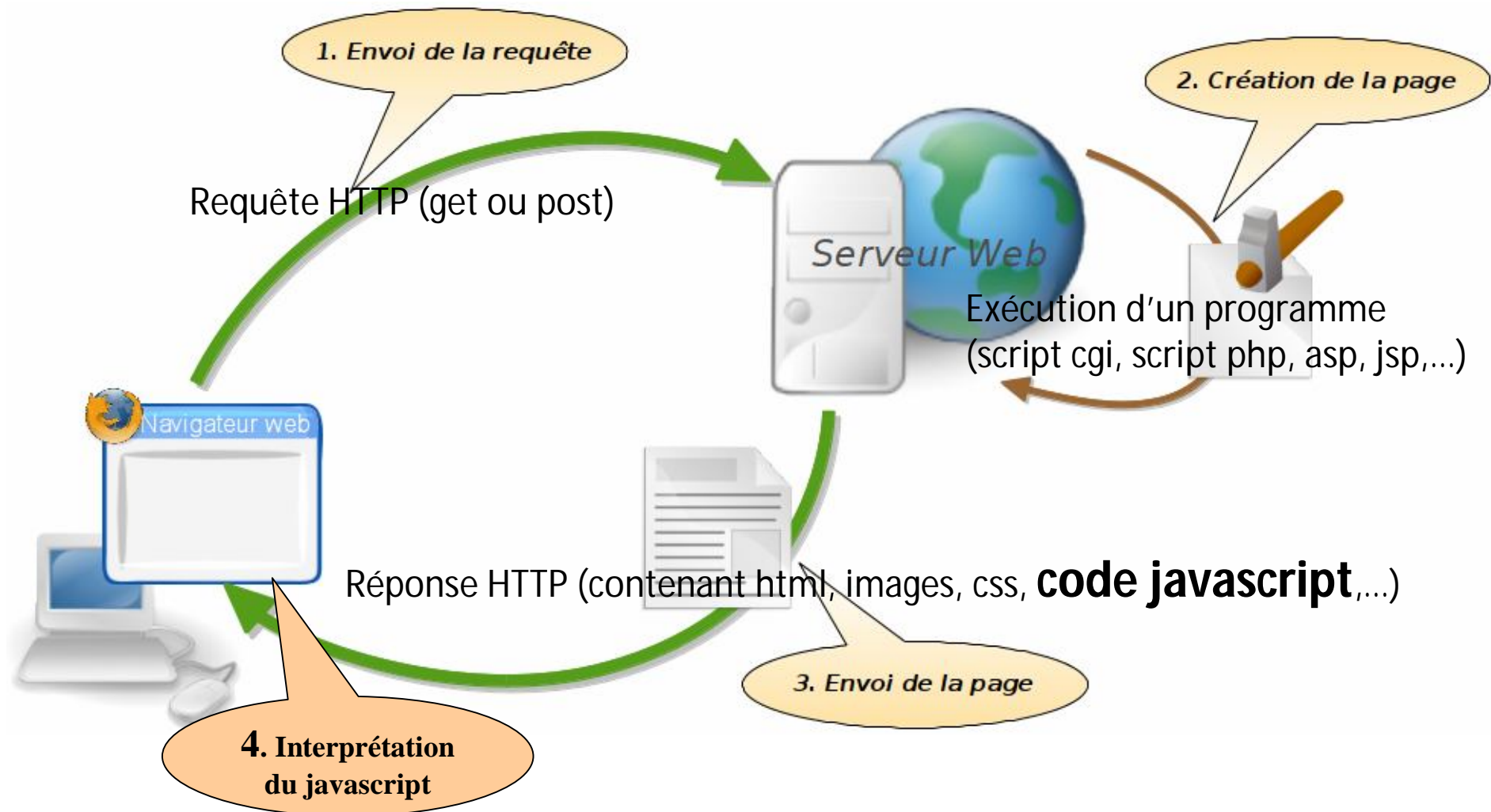
# Cours 2 – M4103C

## Programmation Web orientée client AJAX & JQuery

2016-2017

# Environnement d'exécution d'une application WEB

---



# Un exemple d'application Web orientée client

---

## Démo

# Documentation

---

## **AJAX :**

<http://api.jquery.com/category/ajax/>

<http://www.w3schools.com/ajax/>

<http://fr.openclassrooms.com/informatique/cours/ajax-et-l-echange-de-donnees-en-javascript>

## **jQuery :**

<http://api.jquery.com/>

<http://www.w3schools.com/jquery/>

<http://learn.jquery.com/>

<http://fr.openclassrooms.com/informatique/cours/simplifiez-vos-developpements-javascript-avec-jquery/>

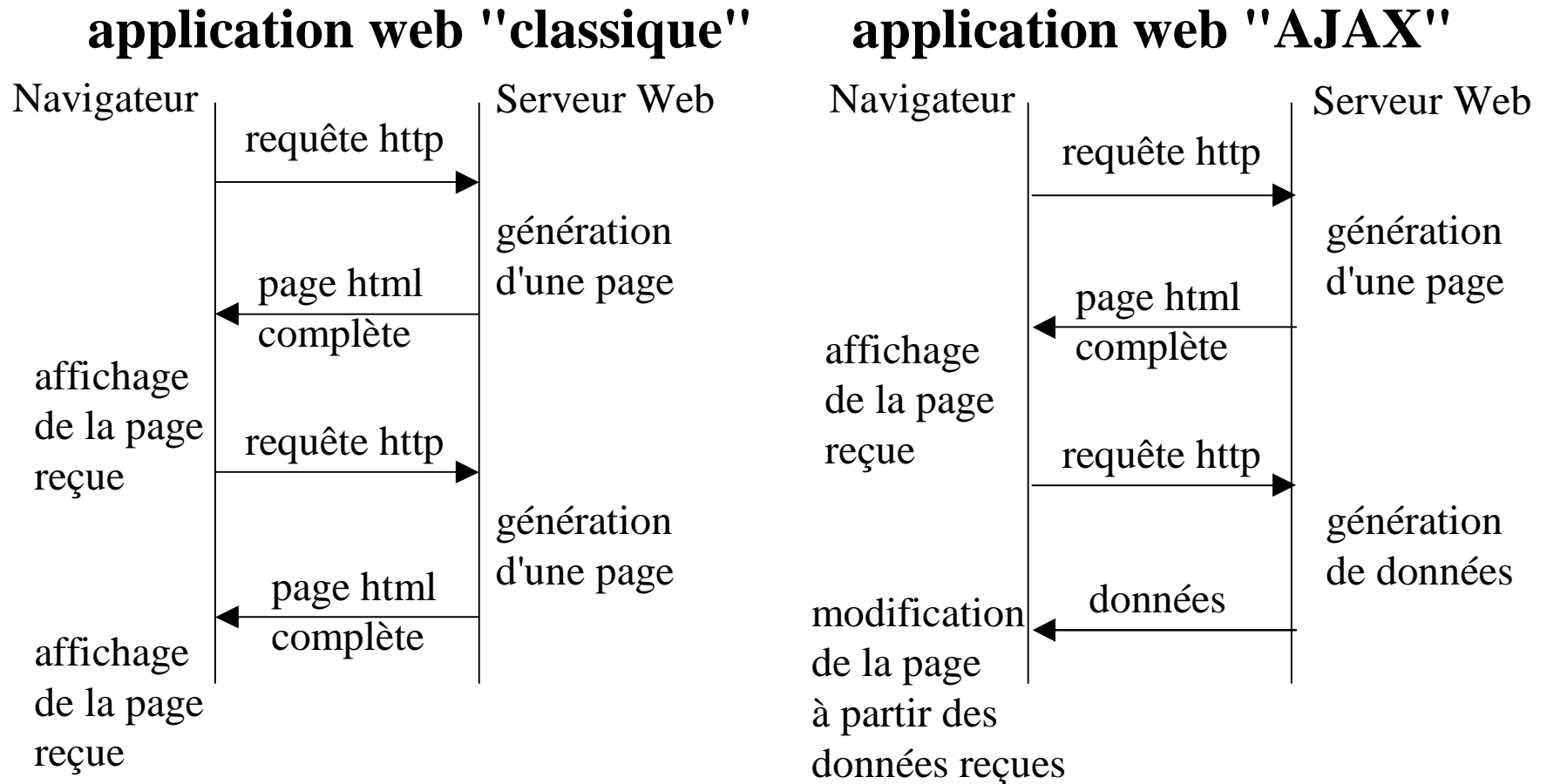
<http://fr.openclassrooms.com/informatique/cours/jquery-ecrivez-moins-pour-faire-plus>

## **jQueryUI :**

<http://api.jqueryui.com/>

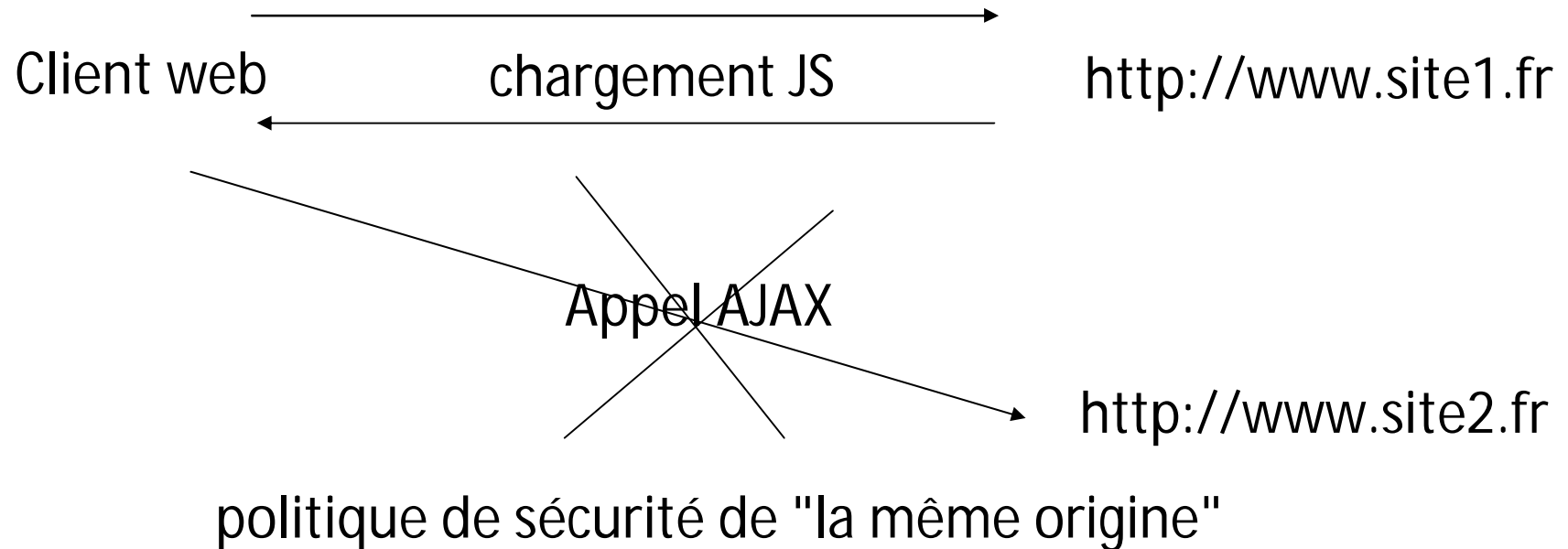
# AJAX : Le principe

---



## Attention au croisement de domaines

---



Solution : site2 doit autoriser des accès à la ressource provenant de pages du site1.

```
<?PHP
```

```
header('Access-Control-Allow-Origin: http://www.site1.fr');...
```

---

# AJAX : Le Sigle

---

**A**synchronisme de la  
communication (pas d'attente)  
(en fait pas toujours)

format **X**ml des données  
échangées (en fait pas toujours)

**Asynchronous Javascript And Xml**

## Vue du client

i1

i2

appel serveur

i3

i4

traitement retour serveur

par fonction dite de

**callback**

la demande auprès du serveur  
et l'utilisation des données reçues  
est faite en **J**avascript (toujours)

# Format des données reçues par le client

---

## •Texte Simple

Gordon Barney Eli

simple mais que faire  
quand les données sont structurées

## •HTML

```
<ul>
```

```
  <li>Gordon</li>
```

```
  <li>Barney</li>
```

```
  <li>Eli</li>
```

```
</ul>
```

pratique : `.innerHTML =`  
mais verbeux

## •XML

```
<friends>
```

```
  <f name="Gordon"/>
```

```
  <f name="Barney"/>
```

```
  <f name="Eli"/>
```

```
</friends>
```

choix de mise en forme  
laissée au client

## •JSON

```
[{"name": "Gordon"}, {"name": "Barney"}, {"name": "Eli"}]
```

léger et complet, tableaux, objets



# Exemple JSON (JavaScript Object Notation)

---

## JSON

```
{ "menu": { "id": "file",  
            "value": "File",  
            "popup": {  
                "menuitem": [  
                    { "value": "New", "onclick": "CreateNewDoc()" },  
                    { "value": "Open", "onclick": "OpenDoc()" },  
                    { "value": "Close", "onclick": "CloseDoc()" } ]  
            }  
}
```

## Equivalent XML

```
<menu id="file" value="File">  
  <popup>  
    <menuitem value="New" onclick="CreateNewDoc()" />  
    <menuitem value="Open" onclick="OpenDoc()" />  
    <menuitem value="Close" onclick="CloseDoc()" />  
  </popup>  
</menu>
```

# L'objet XMLHttpRequest()

---

```
function ajax_get_request(callback,url,async)
{
    var xhr = new XMLHttpRequest(); //création de l'objet

    xhr.onreadystatechange = function()
    {
        if ((xhr.readyState==4) && (xhr.status == 200))
        {
            callback(xhr.responseText);
        }
    };

    xhr.open("GET",url, async); // initialisation de l'objet
    xhr.send();                // envoi de la requête
}
```

méthode http

asynchrone ou pas

script appelé (avec les paramètres)

## xhr.readyState : 0,1,2,3 ou 4

- 0: L'objet XHR a été créé, mais pas encore initialisé
  - 1: L'objet XHR a été créé, mais pas encore envoyé
  - 2: La méthode send vient d'être appelée
  - 3: Le serveur traite les informations et a commencé à renvoyer des données
  - 4: Le serveur a fini son travail, et toutes les données sont réceptionnées
-

# Un exemple d'appel avec callback

---

```
<body><p>
<button onclick="ajax_get_request(readData,
                                'http:../trait1.php?par=2',
                                true)">
Afficher les données reçues</button>
<div id="output"></div>
</p>
...
<script>
// callback
function readData(sData)
{
                                // traitement des données reçues
                                document.getElementById("output")

    .innerHTML=sData;
}
</script>
</body>
</html>
```

---

# Ajax POST avec callback

---

```
function ajax_post_request(callback,url,async,data)
{
    var xhr = new XMLHttpRequest(); //création de l'objet
    xhr.onreadystatechange = function()
    {
        if ((xhr.readyState==4) && (xhr.status == 200))
        {
            callback(xhr.responseText);
        }
    };

    xhr.open("POST",url,async); // initialisation de l'objet
    xhr.setRequestHeader("Content-Type",
        "application/x-www-form-urlencoded");
    //format des données envoyées dans le corps de la requête HTTP
    xhr.send("data="+data);//envoi de la requête avec données
}
```

# Ajax POST sans callback

---

```
function ajax_post_request_base(url,async,data)
{
    var xhr = new XMLHttpRequest(); //création de l'objet

    xhr.open("POST",url,async); // initialisation de l'objet

    xhr.setRequestHeader(
        "Content-Type", "application/x-www-form-urlencoded"
    ); //format des données envoyées

    xhr.send("data="+data); // envoi de la requête avec données
}
```

# Récupération d'un fichier XML ou JSON

---

```
function ajax_get_request_xml(callback,url,async)
{

    var xhr = new XMLHttpRequest();
    xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 &&
            (xhr.status == 200 || xhr.status == 0))
            {callback(xhr.responseXML);}
    };
    xhr.open("GET", "fichier.xml", true);
    xhr.send();
}
```

Pour JSON : du texte (JSON.parse, JSON.stringify)  
var objet = eval('(' + textejson + ')');  
ou mieux var objet = JSON.parse(textejson);

---

# fichier XML récupéré

---

```
<?xml version="1.0" encoding="utf-8"?>
<!-- XMLHttpRequest_getXML.xml -->
<root>
  <soft name="Adobe Dreamweaver" />
  <soft name="Microsoft Expression Web" />
  <soft name="Notepad++" />
  <soft name="gedit" />
  <soft name="Emacs" />
</root>
```

# Traitement du fichier XML

---

```
function readData(oData) {  
    var nodes = oData.getElementsByTagName("soft");  
    var ol = document.createElement("ol"), li, cn;  
  
    for (var i=0, c=nodes.length; i<c; i++) {  
        li = document.createElement("li");  
        cn = document.createTextNode(nodes[i].getAttribute("name"));  
  
        li.appendChild(cn);  
        ol.appendChild(li);  
    }  
  
    document.getElementById("output").appendChild(ol);  
}
```



# Limites de la programmation en JavaScript

---

Attention des différences entre navigateurs qui tendent à s'estomper

Par exemple dans le cas des appels AJAX sous IE 7 (c'est vieux quand-même) :

```
if (window.XMLHttpRequest || window.ActiveXObject) {  
    if (window.ActiveXObject) {  
        try {  
            xhr = new ActiveXObject("Msxml2.XMLHTTP");  
        } catch(e) {  
            xhr = new ActiveXObject("Microsoft.XMLHTTP");  
        }  
    } else {  
        xhr = new XMLHttpRequest();  
    }  
} else ...
```

Et tout ça est très verbeux...

---

## Solution : Utiliser une bibliothèque JavaScript

---



# JQuery - "installation"

---

`<script type="text/javascript" src="jquery-3.1.1.js"></script>`  
Fichier à télécharger (environ 250 ko - 10 000 lignes de codes)

Ou bien par CDN (Content Delivery Network)

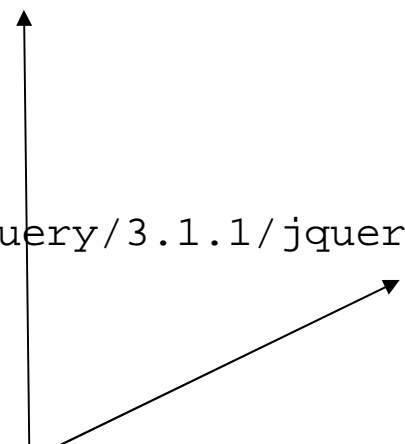
`<script src="//code.jquery.com/jquery-3.1.1.js"></script>`

Ou bien utilisation de la version compressée

`<script src="//code.jquery.com/jquery-3.1.1.min.js"></script>`

Ou bien

`<script type="text/javascript"`  
`src="http://ajax.googleapis.com/ajax/libs/jquery/3.1.1/jquery.min.js">`  
`</script>`



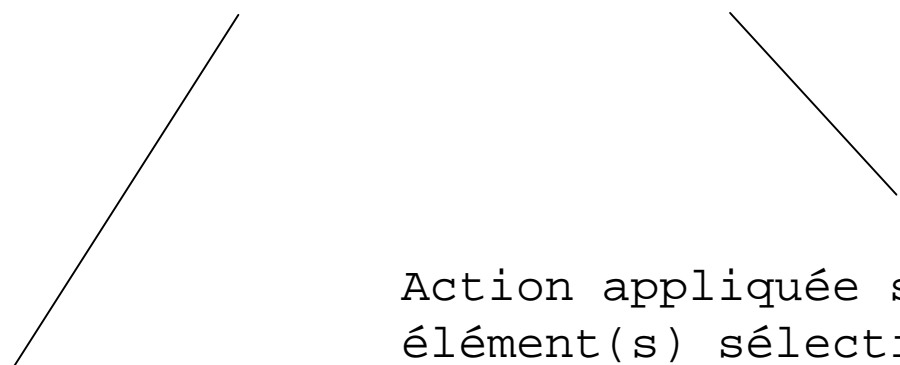
version compressée  
dite de production

---

# JQuery - Syntaxe vue de l'extérieur

---

`$(sélecteur).action`



Action appliquée sur le(s)  
élément(s) sélectionnés

Sélection d'élément(s) de la page

Exemple : `$('#mondiv').html('ce que je veux');`

# JQuery - Syntaxe vue de l'intérieur

---

`$(sélecteur).action`

Raccourci pour la  
fonction `jQuery()`  
Retourne un objet contenant  
une collection d'éléments  
(! Structure de données  
propre à jQuery)

Méthode de l'objet retourné  
(! Méthode propre à l'objet  
Retourné par la fonction  
`jQuery()` )

Paramètre de la fonction `jQuery`  
Chaîne décrivant les élément(s) à  
Sélectionner dans la page

~~`$('#mondiv').innerHTML=....`~~

---

# JQuery : Sélecteurs

Expression	Retour
<code>*</code>	Toutes les balises.
<code>elem</code>	Les balises elem.
<code>#id</code>	Balise ayant l'id "id".
<code>.class</code>	Balises ayant la classe "class".
<code>elem[attr]</code>	Balises elem dont l'attribut "attr" est spécifié.
<code>elem[attr="val"]</code>	Balises elem dont l'attribut "attr" est à la valeur val.
<code>elem bal</code>	Balises bal contenues dans une balise elem.
<code>elem &gt; bal</code>	Balises bal directement descendantes de balises elem.
<code>elem + bal</code>	Balises bal immédiatement précédées d'une balise elem.
<code>elem ~ bal</code>	Balises bal précédées d'une balise elem.

# JQuery : Sélecteurs

```
1 1 : <p id="premier_texte">
2 2 :   <span class="texte">
3       Salut tout le monde
4   </span>
5 3 :   
6   </p>
7 4 : <p>
8 5 :   
9 6 :   <span class="texte">
10      ma Seconde Photo de Vacances !
11   </span>
12 </p>
13 7 : 
```

Expression	Numéros des éléments sélectionnés
#premier_texte .texte	2.
p > span	2 et 6.
span + img	3.
span > img	Rien, car les balises img ne sont pas contenues dans les balises span.

# JQuery : Sélecteurs

```
1 1 : <p id="premier_texte">
2 2 :   <span class="texte">
3       Salut tout le monde
4   </span>
5 3 :   
6   </p>
7 4 : <p>
8 5 :   
9 6 :   <span class="texte">
10      ma Seconde Photo de Vacances !
11   </span>
12 </p>
13 7 : 
```

<code>p</code>	1 et 4.
<code>img[src\$=.jpg]</code>	3 et 7 (pas la 5 car l'attribut src se finit par .gif).
<code>img[src*=hoto]</code>	3, 5 et 7 (car ils contiennent tous hoto (photo_ est en commun) dans leur attribut src).
<code>img:visible</code>	3 et 5.
<code>p ~ img</code>	7.



# JQuery : Sélecteurs

Expression	Retour
<code>:hidden</code>	Éléments invisibles, cachés.
<code>:visible</code>	Éléments visibles.
<code>:parent</code>	Éléments qui ont des éléments enfants.
<code>:header</code>	Balises de titres : h1, h2, h3, h4, h5 et h6.
<code>:not(s)</code>	Éléments qui ne sont pas sélectionnés par le sélecteur s.
<code>:has(s)</code>	Éléments qui contiennent des éléments sélectionnés par le sélecteur s.
<code>:contains(t)</code>	Éléments qui contiennent du texte t.
<code>:empty</code>	Éléments dont le contenu est vide.
<code>:eq(n)</code> et <code>:nth(n)</code>	Le n-ième élément, en partant de zéro.

# JQuery : Sélecteurs

---

<code>:gt (n)</code> (greater than, signifiant plus grand que)	Éléments dont le numéro (on dit l'« index ») est plus grand que n.
<code>:lt (n)</code> (less than, signifiant plus petit que)	Éléments dont l'index est plus petit que n.
<code>:first</code>	Le premier élément (équivalent à <code>:eq (0)</code> ).
<code>:last</code>	Le dernier élément.
<code>:even</code> (pair)	Éléments dont l'index est pair.
<code>:odd</code> (impair)	Éléments dont l'index est impair.

# JQuery : Sélecteurs

```
1 1 : <p id="premier_texte">
2 2 :   <span class="texte">
3       Salut tout le monde
4       </span>
5 3 :   
6       </p>
7 4 : <p>
8 5 :   
9 6 :   <span class="texte">
10      ma Seconde Photo de Vacances !
11      </span>
12      </p>
13 7 : 
```

`p:first + img`

Rien, car aucune balise img ne suit directement la première balise p.

`:hidden`

7, car dans son attribut style, display est à none.

`img:hidden:not(.superimage)`

Rien, car la seule image non-visible a la classe superimage

`p:contains('Salut'):has(span)`

1, car contient "Salut" (dans le span) et une balise span.

`:not(html):not(body):even:not(img)`

2, 4 et 6 (les `:not(html)` et `:not(body)` évitent de récupérer ces balises, `:even` sélectionne les numéros pairs et le `:not(img)` ne change rien).

# JQuery : Sélecteurs

---

Expression	Retour
<code>[attr]</code>	Éléments qui ont l'attribut <code>attr</code> .
<code>[attr="val"]</code>	Éléments dont la valeur de l'attribut <code>attr</code> est égale à <code>val</code> .
<code>[attr*="val"]</code>	Éléments dont la valeur de l'attribut <code>attr</code> contient <code>val</code> .
<code>[attr!="val"]</code>	Éléments dont la valeur de l'attribut <code>attr</code> ne contient pas <code>val</code> .
<code>[attr^="val"]</code>	Éléments dont la valeur de l'attribut <code>attr</code> commence par <code>val</code> .
<code>[attr\$="val"]</code>	Éléments dont la valeur de l'attribut <code>attr</code> finit par <code>val</code> .

---

## JQuery en action - accès/modification de contenu

---

```
// Affiche le contenu "J'aime les frites."  
<div id="titre">J'aime les frites.</div>  
alert($('#titre').html());  
  
// Remplace le contenu ("J'aime les frites.") par "Je mange une pomme".  
$('#titre').html('Je mange une pomme');  
  
// Remplace le contenu par le titre de la page  
$('#titre').html($('#title').html());  
  
// Ajoute du contenu après chaque balise textarea.  
$('#textarea').after('<p>Veuillez ...</p>');  
  
// Ajoute "Voici le titre :" avant la balise ayant comme id "titre".  
$('#titre').before('Voici le titre :');  
  
// Ajoute "! Wahou !" après la balise ayant comme id "titre".  
$('#titre').after('! Wahou !');
```

---

# jQuery en action - modification de contenu

---

```
// Multiplie le nombre de boutons par 2.
$('button').clone().appendTo($('body'));

// Revient à faire :
$('body').append($('button').clone());

// Supprime les liens de la page ayant la classe "sites".
$('a').remove('.sites');

// Supprime les balises <strong> dans des <button>.
$('button strong').remove();

$('button').empty(); // Vide les boutons.

$('body').empty(); // Vide la page web.

// Revient à faire :
$('body').html('');
```

---

# JQuery en action - modification de contenu

---

```
// Remplace les liens <a>...</a> par <em>censuré</em>.
$('a').replaceWith('<em>censuré</em>');

$('h1').replaceWith($('h1').html());
$('#titre').replaceWith('<h1>'+$('#titre').html()+'</h1>');
$('.recherche').replaceWith('<a href="http://google.com">G</a>');

$('span').prepend('balise span » ');
transformera <span>Hello World !</span>
en <span>balise span » Hello World !</span>.

$('span').append(' « balise span');
transformera <span>Hello World !</span> en
<span>Hello World ! « balise span</span>.
```

# JQuery en action - modification des attributs

---

```
$('div.header_gauche img').attr('title','nouveau titre');

// Tous les éléments de votre page perdront leurs classes.
$('*').removeAttr('class');

// Tous les liens de votre page ne s'ouvriront pas
// dans une nouvelle fenêtre .
$('a').removeAttr('target');

// Décoche toutes les checkbox et tous les boutons radio de la page.
$(':checkbox').removeAttr('checked');
```



# jQuery en action - style, navigation dans l'arbre

---

```
// Affiche 'HTML'.
alert($('body').parent().tagName);

// Affiche 'HEAD'.
alert($('title').parents()[0].tagName);

// Couleur de fond
$('body').css('background-color', '#0ff');

$( "ul.level-2" ).children().css( "background-color", "red" );

$( "li.third-item" ).next().css( "background-color", "red" );
```

# jQuery en action - boucle EACH

---

```
<ul>
<li>foo</li>
<li>bar</li>
</ul>
```

```
$( "li" ).each(function( index ) {
console.log( index + ": " + $( this ).text() );
});
```

# JQuery en action - évènements

---

```
$('#a').attr('onclick','...')
```

OU

```
$('#a').click(function(){  
    alert('Vous allez vers : '+$(this).attr('href'));  
});
```

OU

```
$('#a').on('click', function() {...})
```

EXEMPLE AVEC `$(this)` :

```
$('#p').click(function(){  
    $(this).hide();  
});
```

# JQuery en action - Ajax

---

```
<!DOCTYPE html>
<html>
  <head>
    ...
  <body>
    <button id="action">Lancer la requête AJAX</button><br />
    <script src="jquery.js"></script>
    <script>
      $(function() {
        $('#action').click(function() {
          $.ajax({
            type: 'GET',
            url: 'proverbes.php?l=7',
            timeout: 3000,
            success: function(data) {
              alert(data); },
            error: function() {
              alert('La requête n\'a pas abouti'); }});
        });
      });
    </script>
  </body>
</html>
```

---

# jQuery en action - Ajax

---

## **\$.get(URL, callback):**

```
$( "button" ).click(function() {  
    $.get("demo_test.asp", function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

## **\$.post(URL, data, callback):**

```
$( "button" ).click(function() {  
    $.post("demo_test_post.asp",  
    {  
        name: "Donald Duck",  
        city: "Duckburg"  
    },  
    function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

---

# JQuery en action - ui -<http://jqueryui.com>

---

```
<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>jQuery UI Datepicker - Default functionality</title>
<link rel="stylesheet" href="//code.jquery.com/ui/1.11.3/themes/
smoothness/jquery-ui.css">
<script src="//code.jquery.com/jquery-2.2.3.js"></script>
<script src="//code.jquery.com/ui/1.11.3/jquery-ui.js"></script>
<link rel="stylesheet" href="/resources/demos/style.css">
<script>
$(function() {$( "#datepicker" ).datepicker();});
</script>
</head>
<body>
<p>Date: <input type="text" id="datepicker"></p>
</body>
</html>
```

Voir les exemples donnés sur <http://jqueryui.com>

---

## jQuery en action -onload, ready,\$(function){...

---

Rappel : les actions effectuées sur des éléments HTML ne peuvent être accomplies que si la page web est chargée !

`$(document).ready(function){....})`  
`$( function(){....} )` équivalent

actions déclenchées une fois la page web chargée équivalent à onload en JS

# JQuery en action - Conflit de l'alias \$

---

```
<script src="other_lib.js"></script>
<script src="jquery.js"></script>
<script>
// le $ est celui de JQuery          solution1
$.noConflict();
// restore l'ancienne valeur de $ (celle de other_lib)
jQuery( document ).ready(function( $ ) {
// JQuery est automatiquement passé en argument
// à la fonction gérant l'évènement ready
});
// Le code de other_lib avec les $ ici.
</script>
```

```
jQuery.noConflict();
(function( $ ) {
$(function() {
// Le code JQuery avec les $ ici.
});
})(jQuery);
// Le code de other_lib avec les $ ici.
```

solution2