

JavaScript

CM1-2 : Ajax

Mickaël Martin Nevot

V3.0.0



Cette œuvre de [Mickaël Martin Nevot](#) est mise à disposition selon les termes de la [licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage à l'Identique 3.0 non transposé](#).

JavaScript

- I. Présentation du cours
- II. JavaScript
- III. Ajax
- IV. JS appliqué aux nouvelles technologies Web

Web 2.0 et RIA

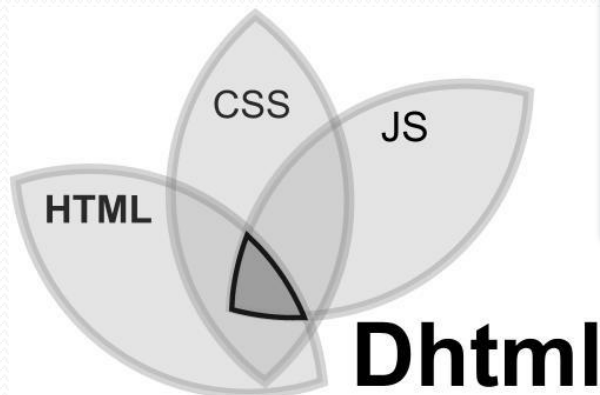
- Web 2.0, évolution du Web :
 - Plus de **simplicité**
 - Plus d'**interactivité**
 - **Participatif et social**
- RIA : *rich Internet application* (**application Internet riche**) :
 - Application Web qui offre des caractéristiques similaires aux logiciels d'ordinateurs



DHTML vs Ajax

DHTML

- DHTML : *dynamic HTML* (HTML dynamique) :
 - Ensemble de techniques permettant à une page Web de se modifier elle-même en cours de consultation



Ajax

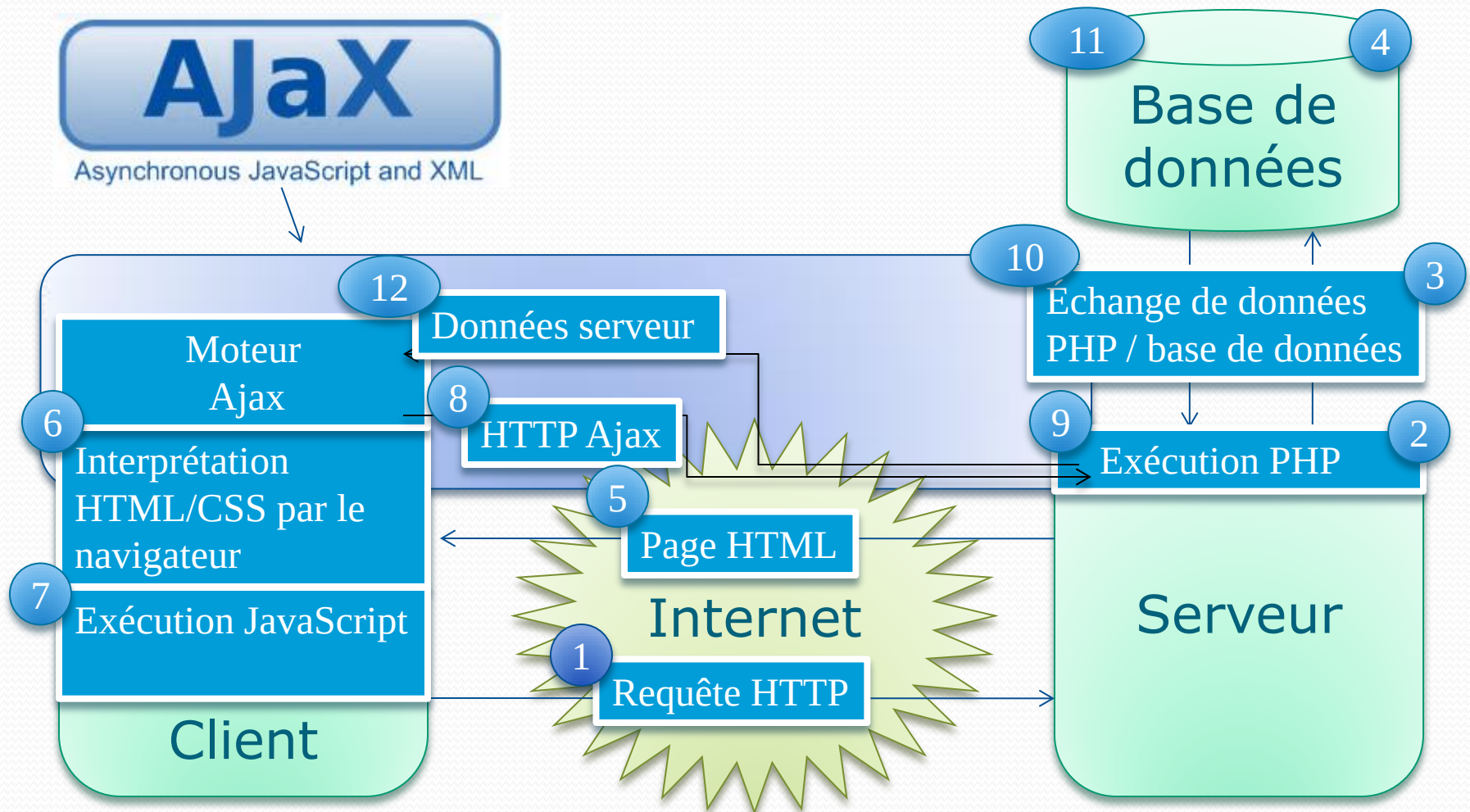
- Ajax : *asynchronous JavaScript and XML* :
 - Combinaison de technologies permettant de réaliser des sites Web interactifs et des RIA

Différences entre les deux :

- Appel serveur en Ajax
- Jamais de rechargement de page Web en Ajax



Ajax



Ajax

- **HTML**
- **CSS**
- **JavaScript**
- **DOM**
- **XMLHttpRequest**
- **XML/JSON**
- ***Frameworks***



Ajax est un concept permettant des appels asynchrones (parfois synchrones) à un serveur depuis un client

Lors des appels Ajax, le serveur retourne du XML, du JSON ou du texte (pas de HTML) qui sera « récupéré » et traité par JavaScript

DOM

- *Document object model* (modèle objet de document) :
 - **Arborescence** des éléments d'une page « balisée » (HTML)
 - **API** qui permet aux scripts d'accéder ou de mettre à jour le contenu, la structure ou le style d'une page « balisée »
 - **Arbre** (DOM) : ← Cette organisation forme un arbre
 - **Nœud** (*node*) : élément constituant de l'arbre (balise, attribut, etc.)
 - Parent, enfant (fils) et frère : identique à une structure familiale
 - Racine : unique nœud n'ayant pas de parent
 - Feuille : nœud n'ayant pas de fils (attribut par exemple)



Objets du DOM



- Objet `window` :
 - Fenêtre du **navigateur**
 - Variable globale = attribut de `window`
 - En général, non spécifié ← `alert() = window.alert()`
- Objet `document` :
 - **Page Web** (fils de `window`)
 - Racine de l'arbre DOM

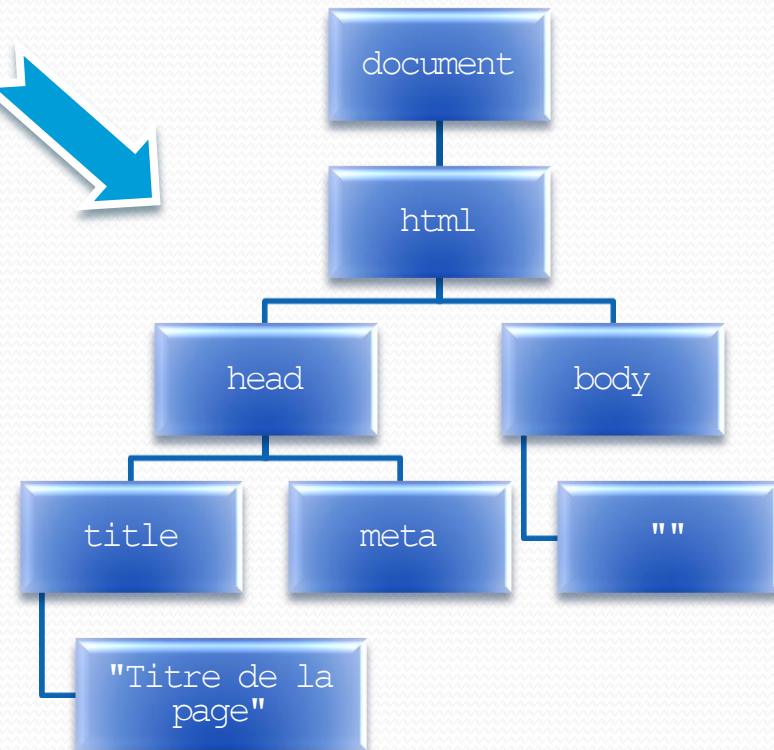


Arbre DOM

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
  <head>
    <title>Titre de la page</title>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
  </head>
  <body>

    <!-- Ici votre site Web. -->

  </body>
</html>
```



Propriétés du DOM

Propriété	Action
Informations	
<code>id</code>	Identifiant d'un nœud
<code>className</code>	Classe d'un nœud
<code>nodeName</code>	Nom du nœud
<code>nodeValue</code> (ou <code>data</code>)	Valeur/contenu du nœud
<code>nodeType</code>	Type du nœud (voir ci-après)
<code>offsetX</code>	<code>X</code> = Dimension (<code>Height</code> , <code>Width</code>)/position (<code>Left</code> , <code>Top</code>)
Accéder à un nœud	
<code>getElementById(id)</code>	Récupération d'un nœud par son identifiant
<code>getElementsByTagName(tag)</code>	Récupération des nœuds d'un type de balise
<code>getElementsByTagName(name)</code>	Récupération des nœuds d'un même nom
<code>getAttribute(attribute)</code>	Récupération de la valeur d'un attribut

Propriétés du DOM

Propriété	Action
Se déplacer dans l'arbre	
<code>hasChildNodes</code>	Existence de nœud enfant
<code>length</code>	Nombre de nœuds dans une liste
<code>childNodes</code>	Liste de nœuds enfants
<code>firstChild</code>	Premier nœud enfant
<code>lastChild</code>	Dernier nœud enfant
<code>nextSibling</code>	Prochain nœud d'un type (nœud de même niveau)
<code>parentNode</code>	Nœud parent
<code>previousSibling</code>	Nœud précédent d'un type (nœud de même niveau)

Propriétés du DOM

Méthode	Action
Gestion des nœuds	
<code>createElement(elem)</code>	Création d'un nouveau nœud (<code><div></div></code> , etc.)
<code>createTextNode(text)</code>	Création d'un nœud texte
<code>appendChild(node)</code>	Insertion d'un nœud après le dernier enfant
<code>insertBefore(n1, n2)</code>	Insertion d'un nœud avant un autre nœud
<code>replaceChild(n1, n2)</code>	Remplacement d'un nœud par un autre
<code>removeChild(node)</code>	Suppression d'un nœud
<code>cloneChild(opt)</code>	Clonage d'un nœud
<code>setAttribute(att, val)</code>	Création ou modification d'un attribut
<code>innerHTML</code>	Lecture ou écriture du contenu d'un nœud
<code>style</code>	Modification du style (CSS) d'un nœud

Un style DOM $\approx$ style CSS (sauf – : par exemple, `text-align` $\rightarrow$ `textAlign`)

Types de nœud

Numéro	Type de nœud
1	Nœud élément
2	Nœud attribut
3	Nœud texte
4	Nœud pour passage CDATA (relatif au XML)
5	Nœud pour référence d'entité
6	Nœud pour entité
7	Nœud pour instruction de traitement
8	Nœud pour commentaire
9	Nœud document
10	Nœud type de document
11	Nœud de fragment de document
12	Nœud pour notation

Évènements du DOM

Évènement	Action détectée
Souris	
<code>click</code>	Clic de la souris
<code>dblclick</code>	Double-clic de la souris
<code>mouseover</code>	Survol de la souris
<code>mouseout</code>	Fin de survol de la souris
<code>mousedown</code>	Pression (sans relâchement) du bouton gauche de la souris
<code>mouseup</code>	Relâchement du bouton gauche de la souris
<code>mousemove</code>	Déplacement de la souris
Clavier	
<code>keydown</code>	Pression (sans relâchement) d'une touche du clavier
<code>keyup</code>	Relâchement d'une touche du clavier
<code>keypress</code>	Pression et relâchement d'une touche du clavier

Évènements du DOM

Évènement	Action détectée
Page/ fenêtre	
<code>error</code>	Erreur durant le chargement
<code>load</code>	Fin du chargement de la page
<code>unload</code>	Déchargement de la page (changement de page, etc.)
<code>resize</code>	Redimensionnement de la fenêtre
Formulaire	
<code>focus</code>	Prise de focus (« ciblage »)
<code>blur</code>	Perte de focus
<code>change</code>	Changement d'un élément de saisie d'un formulaire
<code>select</code>	Sélection d'un champ textuel d'un formulaire
<code>submit</code>	Action du bouton de soumission d'un formulaire
<code>reset</code>	Action du bouton de réinitialisation d'un formulaire

Objet Event

Propriété de Event	Description
<code>altkey</code>	Renvoie vrai si la touche <code>Alt</code> a été actionnée
<code>ctrlKey</code>	Renvoie vrai si la touche <code>Ctrl</code> a été actionnée
<code>shiftKey</code>	Renvoie vrai si la touche <code>Shift</code> a été actionnée
<code>keyCode</code>	Renvoie le code de touche actionnée
<code>clientX</code>	Abscisse de la souris
<code>clientY</code>	Ordonnée de la souris
<code>screenX</code>	Abscisses de l'évènement
<code>screenY</code>	Ordonnée de l'évènement

Lorsqu'un évènement se produit, un objet `Event` est créé et il est accessible depuis la fonction de traitement de l'évènement

Gestionnaires d'évènements

- Utilisation recommandée des gestionnaires d'évènements :

```
// Par exemple : <span id="id">Cliquez-moi !</span>.
var element = document.getElementById("id");

function addEvent(element, event, func) {
    if (element.addEventListener) {
        // Si notre élément possède la méthode addEventListener().
        // Mozilla, Chrome, Safari, Opera,...
        element.addEventListener(event, func, false);
    } else {
        // Si notre élément ne possède pas la méthode addEventListener().
        // Internet Explorer.
        element.attachEvent("on" + event, func);
    }
}

function myFunction(e) {
    alert("Clic : (" + e.clientX + ", " + e.clientY + ")");
}

addEvent(element, "click", myFunction);
```

Gestionnaires d'évènements

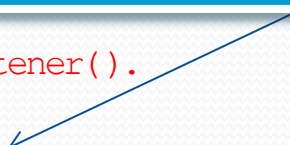
- Utilisation recommandée des gestionnaires d'évènements :

```
// Par exemple : <span id="id">Cliquez-moi !</span>.
```

```
var element = document.getElementById("id");
```

```
function addEvent(element, event, func) {  
    if (element.addEventListener) {  
        // Si notre élément possède la méthode addEventListener().  
        // Mozilla, Chrome, Safari, Opera,...  
        element.addEventListener(event, func, false);  
    } else {  
        // Si notre élément ne possède pas la méthode addEventListener().  
        // Internet Explorer.  
        element.attachEvent("on" + event, func);  
    }  
}  
  
function myFunction(e) {  
    alert("Clic : (" + e.clientX + ", " + e.clientY + ")");  
}  
  
addEvent(element, "click", myFunction);
```

Capture (sens de propagation)



Certains événements appliqués à un élément parent peuvent se propager d'eux-mêmes aux éléments enfants, c'est le cas des événements mouseover, mouseout, mousemove, click

HTTP : rappels

- HTTP (*hypertext transfer protocol*) :

- Protocole de communication (client-serveur)

- Méthodes (commandes) :  Utilisent très souvent une URL

- **GET** : construction d'URL dynamique

 `http://localhost/login.php?login=login&pwd=test`

- **POST** : passage « caché » des valeurs

 `http://localhost/login.php`

URL : de l'anglais *uniform resource locator*, littéralement « localisateur uniforme de ressource », est une chaîne de caractères utilisée pour adresser les ressources du Web, aussi appelée **adresse Web**

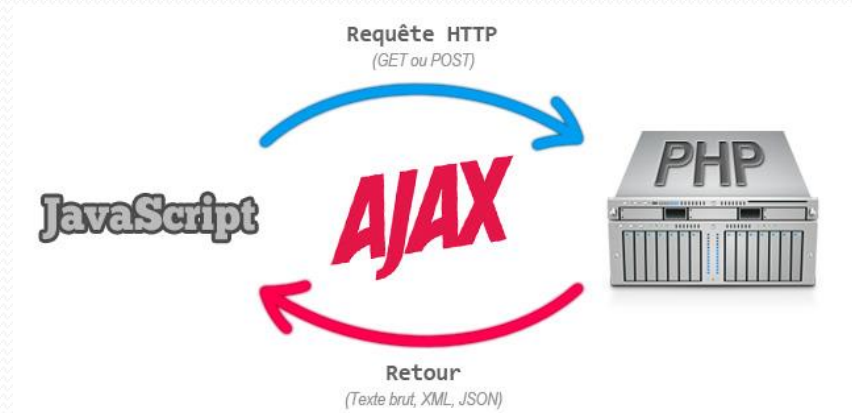
Synchrone/asynchrone

- **Synchrone :**

- La fonction qui envoie une requête au serveur est la même que celle qui en recevra la réponse
- Exécution suspendue en attendant la réponse du serveur

- **Asynchrone :** ← Attention : penser à avertir l'utilisateur !

- La fonction qui envoie une requête au serveur n'est pas la même que celle qui en recevra la réponse
- Non bloquant
- Fonction de *callback*



XMLHttpRequest



- Création du moteur Ajax

```
function createXHR() {  
    // Mozilla, Safari, Opera,...  
    try { return new XMLHttpRequest(); } catch(e) {}  
    // Internet Explorer.  
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0"); } catch (e) {}  
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0"); } catch (e) {}  
    try { return new ActiveXObject("Msxml2.XMLHTTP"); } catch (e) {}  
    try { return new ActiveXObject("Microsoft.XMLHTTP"); } catch (e) {}  
    alert("XMLHttpRequest non supporté");  
    // Non supporté.  
    return null;  
}
```



XMLHttpRequest

- Utilisation synchrone :

```
...  
var anticache = new Date().getTime();  
// Oblige le navigateur à mettre à jour.  
// Le paramètre URL « anticache » peut être appelé autrement.  
var param = "p1=foo&p2=bar" + "&anticache=" + anticache;  
// Création de l'objet XMLHttpRequest.  
var XHRObj = createXHR();  
// Ouvre une connexion synchrone avec le serveur en GET.  
XHRObj.open("get", "server.php?" + param, false);  
// Envoi de la « requête » au serveur en GET (toujours null).  
XHRObj.send(null);  
// Récupération de la réponse du serveur.  
var res = XHRObj.responseText; // Ou XHRObj.responseXML...
```

Attention : Penser à protéger les variables (caractères spéciaux, espaces, etc.) : `encodeURIComponent()`

XMLHttpRequest

- Utilisation asynchrone : Le plus souvent

```
function callback() {  
    // 0 non init., 1 en chargement, 2 chargé, 3 en traitement, 4 terminé.  
    if (XHRObj.readyState == 4) {  
        // 404 page non trouvée, 403 accès refusé, 200 requête réussie, etc.  
        if (XHRObj.status == 200) {  
            alert("Réponse : " + XHRObj.responseText);  
        }  
    }  
}  
...  
var param = "p1=foo&p2=bar"; // Pas besoin d'anti-cache.  
// Création de l'objet XMLHttpRequest.  
var XHRObj = createXHR();  
// Ouvre une connexion asynchrone avec le serveur en POST.  
XHRObj.open("post", "server.php", true);  
// Déclaration de la fonction de rappel.  
XHRObj.onreadystatechange = callback;  
// Internet media type: application/x-www-form-urlencoded, text/xml, etc.  
XHRObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  
// Envoi de la « requête » au serveur en POST (toujours un paramètre).  
XHRObj.send(param);
```

XMLHttpRequest

- Utilisation asynchrone : ← Le plus souvent

```
function callback() {  
    // 0 non init., 1 en chargement, 2 chargé, 3 en traitement, 4 terminé.  
    if (XHRObj.readyState == 4) {  
        // 404 page non trouvée, 403 accès refusé, 200 requête réussie, etc.  
        if (XHRObj.status == 200) {  
            alert("Réponse : " + XHRObj.responseText);  
        }  
    }  
}
```

```
...  
var param = "p1=foo&p2=bar"; // Pas besoin d'anti-cache.  
// Création de l'objet XMLHttpRequest.  
var XHRObj = createXHR();  
// Ouvre une connexion asynchrone avec le serveur en POST.  
XHRObj.open("post", "server.php", true);  
// Déclaration de la fonction de rappel.  
XHRObj.onreadystatechange = callback;  
// Internet media type: application/x-www-form-urlencoded, text/xml, etc.  
XHRObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");  
// Envoi de la « requête » au serveur en POST (toujours un paramètre).  
XHRObj.send(param);
```

XMLHttpRequest

- Côté serveur (PHP) :

```
// Type de la réponse : HTML, encodage : UTF-8.
header('Content-Type: text/html ; charset=utf-8');
// Anti-cache pour HTTP/1.1.
header('Cache-Control: no-cache, private');
// Anti-cache pour HTTP/1.0.
header('Pragma: no-cache');

if (isset($_REQUEST['p1']) {
    $p1 = $_REQUEST['p1'];
}
if (isset($_REQUEST['p2']) {
    $p2 = $_REQUEST['p2'];
}

// Traitement PHP qui affecte le résultat à renvoyer à la variable $res.
// Formats possibles de $res : XML, JSON ou textuel (chaîne de caractères).

// Renvoi au client.
echo $res;
```



JSON



- *JavaScript object notation*
- Format de données textuelles
- Dérivé de la notation des objets ECMAScript

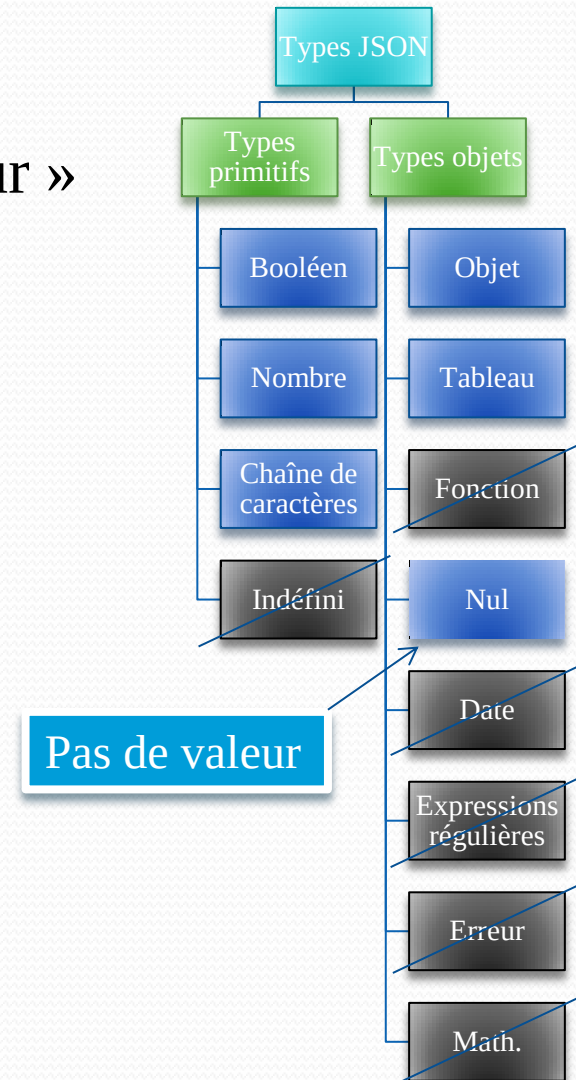
```
{
  "menu":
  {
    "id":"file",
    "value":"File",
    "popup":
    {
      "menuitem":
      [
        { "value":"New", "onclick":"CreateNewDoc()" },
        { "value":"Open", "onclick":"OpenDoc()" }
      ]
    }
  }
}
```

Format très lisible et compact

Structure d'un document JSON

- Deux éléments structurels :
 - Ensembles de paires : « nom:valeur »
 - Listes ordonnées de valeurs

```
"price":  
{  
  "type":"number",  
  "minimum":0,  
  "required":true  
}  
"phoneNumber":  
[  
  {  
    "type":"home",  
    "number":"212 555-1234"  
  },  
  {  
    "type":"fax",  
    "number":"646 555-4567"  
  }  
]
```



XML vs JSON

- Formats de données structurées
- Objectif : faciliter l'échange de contenus (complexes)
- **Cross-domain interdit avec XMLHttpRequest pour un résultat au format XML... mais pas pour un au format ECMAScript (comme JSON !)**
- Performances :
 - Temps de traitement : XML $\approx$ JSON
 - Taille (stockage, transmission) : JSON
- Avantage de JSON :

JSON with Padding (JSONP)



```
// Transforme une expression JSON en variable JavaScript (non conseillé).  
var data = eval('(' + jsonData + ')');
```

JQuery

- Bibliothèque JavaScript très répandu : <http://jquery.com>
- Open source et compatible avec la plupart des navigateurs
- Un seul fichier ←

Disponible en version « développement » (lisible) ou « production » (compressé : min)

- Utilisation simplifiée de JavaScript/Ajax et du DOM
- Méthode recommandée d'utilisation de jQuery :

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
```



Appeler un script externe (chez Google) n'est pas plus lent qu'en local !

Fonction jQuery

- Une seule fonction : `$()` (version abrégée de `jQuery()`)
- Peut prendre un paramètre
- Retourne un objet
- Sans paramètre : `$.` (version abrégée de `jQuery.`)
- **Chaînage des méthodes** (l'objet appelant est retourné) :

```
$("anything").parent("p.quote").find("still anything").show();
```

Les méthodes s'appliquent généralement à tous les éléments sélectionnés



Sélecteurs jQuery

Expression	Retour
"#id"	La balise ayant pour id id
"h1"	Les balises
".class"	Les balises qui ont la classe class
"a, h1, h2"	Les balises <a>, ...
"*"	Toutes les balises
"a[href]"	Attribut
":parent"	Sélecteur spécifique
":visible"	Filtre

```
$("#id")...  
$(".class")...  
$("*")...  
$(":visible")...  
$("a[href]")...  
$("h1#titre")...
```

Les combinaisons
sont possibles

Sélecteurs spécifiques

Sélecteurs CSS « classiques »
(presque tous les sélecteurs CSS
sont utilisables en jQuery)

Méthodes jQuery

```
<div class="demo-cont">
  <div class="demo-box">Démo</div>
  <ul class="level-1">
    <li class="item-1">item 1</li>
    <li>item 2</li>
  </ul>
</div>
```

```
...
.selected { color:blue; }
```

La plupart des méthodes ont à la fois une utilisation d'accesseur (*getter*) et de mutateur (*setter*)

- Contenu textuel : `html()`, `text()`

```
$("div.demo-cont").html(); // <div class="demo-box">Démo... </ul>
$("div.demo-cont").text(); // Démo item 1 item 2
```

- CSS : `css()`, `addClass()`, `removeClass()`

```
$("div.demo-box").css("background-color");
$("li.item-1").addClass("selected");
```

Style CSS ou DOM !

- DOM : `parent()`, `first()`, `last()`, `next()`, `children()`

```
$("li.item-1").parent().css("background-color", "white");
$("ul.level-1").first().css("background-color", "black");
$("li.item-1").next().css("background-color", "red");
$("ul.level-1").children().css("background-color", "blue");
```

Méthodes jQuery

`<span>Hello World !</span>`

- Manipule : `show()`, `hide()`, `wrap()`, `append()`, `prepend()`

```
$("#span").show();
$("#span").hide();
$("#span").wrap("<em></em>"); // <em><span>Hello World !</span></em>.
$("#span").append("<em>Test</em>");
// <span>Hello World !<em>Test</em></span>.
$("#span").prepend("<em>Test</em>");
// <span><em>Test</em>Hello World !</span>.
```

- Méthode utilisateur (JavaScript, rappel) :

```
function myFunction(i) {
    return i + "ème : ";
}
```

```
$("#p").prepend(myFunction);
// ?
```

- Évènements : `.event` (évènement DOM)

Visual jQuery : <http://visualjquery.com>

Méthodes jQuery

- Itération sur tous les éléments d'un objet : `each( )`

```
$("#img").each(function(i){  
    this.alt = "test" + i + ".jpg"; // Ajoute un attribut alt aux images.  
});  
$.each([0,1,1,2,3,5,8,13,21], function(n){  
    $("#body").append(n + "ème nombre de Fibonacci : " + this + "<br />");  
});
```

- Ajax : `ajax( )`

```
$.ajax({  
    url:url,  
    data:data,  
    success:callback  
});
```

- JSON :

```
$.ajax({  
    url:url,  
    dataType:"json",  
    data:data,  
    success:callback  
});
```

L'évènement DOM load n'est levé que lorsque le DOM et les images de la page Web sont chargés (ce qui peut être long). Mieux vaut utiliser à la place la méthode jQuery `$(document).ready(fct)` qui appelle la fonction de traitement `fct` dès que le DOM est chargé

Moteur Ajax : sans jQuery

```
function createXHR() {
    try { return new XMLHttpRequest(); } catch(e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.6.0"); } catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP.3.0"); } catch (e) {}
    try { return new ActiveXObject("Msxml2.XMLHTTP"); } catch (e) {}
    try { return new ActiveXObject("Microsoft.XMLHTTP"); } catch (e) {}
    alert("XMLHttpRequest non supporté");
    return null;
}

function callback() {
    if (XHRObj.readyState == 4) {
        if (XHRObj.status == 200) {
            alert("Réponse : " + XHRObj.responseText);
        }
    }
}

var param = "p1=foo&p2=bar";
var XHRObj = createXHR();
XHRObj.open("post", "server.php", true);
XHRObj.onreadystatechange = callback;
XHRObj.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
XHRObj.send(param);
```

Moteur Ajax : avec jQuery

```
function callback(res) {  
    alert("Réponse : " + res);  
}
```

```
$.ajax({  
    type:"POST",  
    url:"server.php",  
    data:"p1=foo&p2=bar",  
    success:callback  
});
```



Utilisation courante d'Ajax

- **Actualisation d'information** en tâche de fond
- **Complétion automatique**
- Contrôle en temps réel des données d'un **formulaire**
- Navigation dynamique
- Lecture d'un flux **RSS**
- Sauvegarde de documents éditables
- Personnalisation d'interface Web (Netvibes)
- **Widget** interactif
- Chargement progressif de l'information (Google Maps)
- Moteur de recherche sans rechargement de la page Web

Avantages/inconvénients Ajax

- Avantages :
 - Économie de bande passante
 - Améliore l'**ergonomie**
 - Augmente la **réactivité**
 - Évite les blocages
- Inconvénients :
 - Dépendant de **JavaScript**
 - Pas stocké dans l'historique
 - **Pas d'indexation** des contenus



Crédits

Auteur

Mickaël Martin Nevot

mmartin.nevot@gmail.com



Carte de visite électronique

Relecteurs

Cours en ligne sur : www.mickael-martin-nevot.com

