AJAX
Asynchronous JAvascript and Xml

AJAX apporte encore plus de dynamisme à une page Web : Il permet d'intégrer à la page des informations provenant d'un serveur, et cela sans recharger la page.

L'exemple d'utilisation le plus connu est celui du moteur de recherche Google qui affiche des propositions pendant votre saisie de recherche.

Pour cela, le navigateur doit déclencher, par exemple sur l'évènement « onkeyup », une fonction JavaScript qui exécute une requête asynchrone vers Google, lit le résultat, et l'insère à l'écran de façon dynamique.

La clé de ce mode de fonctionnement est l'objet de classe **XMLHttpRequest** qui s'instancie comme suit :

```
var xhr = new XMLHttpRequest();
```

L'objet ainsi obtenu (`xhr`) est ensuite utilisé comme suit :

```
<script>
var xhr ;

function declencheur() // lancée par un évènement. Ex : onclick, onmouseover, onkeyup…
{
	xhr = new XMLHttpRequest();

	xhr.onreadystatechange = travail; //Nom de la fonction sans parenthèses
	xhr.open("GET","toto.txt",true);
	xhr.send();

}

function travail()
{
	// Seuls ces valeurs indiquent un résultat exploitable
	if (xhr.readyState==4 && xhr.status==200)
	{
		var txt= xhr.responseText;
		document.querySelector("#info").innerHTML=txt;
	}
}
</script>
```

2017 : méthode *onload*

```
xhr.onload=travail;
xhr.open("GET",
        "toto.txt", true);
xhr.send();

function travail() {

 var txt=xhr.responseText;
…
}
```

1. `xhr.open("GET","toto.txt",true);`
   Cette méthode ouvre la connexion asynchrone avec le serveur. GET a un comportement similaire à l'attribut *method* de la balise <form>.
   toto.txt est le nom d'un fichier présent sur le serveur. Le serveur sait renvoyer des fichiers avec l'extension html, txt, et xml.
   toto.txt peut être remplacé par le nom d'une page de script (PHP/ASP/JSP) sur le même serveur que celui qui héberge la page Ajax (Limitation de sécurité). Cette page de script devra générer une page de texte brut, JSON ou XML.
   Note importante : Si un script PHP génère du XML, il faut modifier l'entête http avant l'envoi du XML :
   `header('Content-Type: text/xml');`

2. `xhr.send();`
   Envoi de la requête AJAX. Entre les parenthèses, on peut ajouter des informations pour le *header* de la trame, par exemple des données POST (utiliser un objet JavaScript *FormData*).

3. `xhr.`**`onreadystatechange`** `= function() { …….. }`
   On associe à l'objet le code d'une fonction qui sera exécuté à chaque fois que la requête asynchrone change d'état. Tous les états ne nous intéressent pas. Par exemple readyState=1 indique que la connexion est établie avec le serveur, et status=400 indique que la page demandée n'est pas disponible.
   Ce n'est que lorsque **`readyState`**`=4 et` **`status`**`=200` qu'il y a une information valide à lire.

4. `txt= xhr.`**`responseText`**`;`
   La variable txt contient le texte envoyé par le s.erveur. Ce texte peut ensuite être affiché.
   Le contenu JSON est considéré comme du texte.
   Si la réponse est au format XML, on utilise l'objet `xhr.`**`responseXML`** associé aux méthodes JavaScript de parcours des fichiers à balises.

Exemple 1 : Récupération des valeurs fournies par un WebService en réponse XML :

```
x = xhr.responseXML.documentElement.getElementsByTagName("Temperature");


for (i=0; i<x.length; i++)
{
  txt = txt + "Temp: "+ x[i].childNodes[0].nodeValue + "</br>";
}

document.querySelector("#info").innerHTML=txt;
```

Exemple 2 : Récupération des valeurs fournies par un Webservice en réponse JSON :

```
var tabAjax = []; // Le tableau résultat
var j = xhr.responseText;

tabAjax = JSON.parse(j); // Convertir le Json en array JavaScript

for (i = 0; i < tabAjax.length; i++)
{
  txt = txt + "Temp: "+ tabAjax[i][0] + "</br>";
}

document.querySelector("#info").innerHTML=txt;
```

Appel de script sur un autre domaine (*Cross-Domain*) :

Une limitation de sécurité interdit à la méthode *open()* de l'objet XMLHttpRequest (version 1) d'appeler un script sur un site différent. Une explication complète :

http://fr.openclassrooms.com/informatique/cours/ajax-et-l-echange-de-donnees-en-javascript/introduction-33

Ce qu'il faut retenir : Que l'on utilise l'objet *XDomainRequest* de Microsoft ou l'objet *XMLHttpRequest* (version 2) du W3C, le serveur contacté doit ajouter dans l'en-tête http de sa réponse une directive de ce type :

```
header("Access-Control-Allow-Origin: *"); // Autorise TOUTES le requêtes
```

ou une directive plus restrictive pour n'autoriser que certains clients à utiliser le script.

Inconvénient du *XDomainRequest* : il ne fonctionne que sous IE et il n'a pas de méthode *ResponseXML* ...

On préfèrera donc encore un fois le W3C et son objet *XMLHttpRequest*, sachant que seules les dernières versions des navigateurs gèrent le *cross-domain*.

Site de référence recommandé : http://www.w3schools.com