AJAX

Asynchronous Javascript And XML

Sommaire

Pourquoi utiliser Ajax?

Ajax permet de modifier partiellement la page affichée par le navigateur pour la mettre à jour sans avoir à recharger la page entière.

Par exemple le contenu d'un champ de formulaire peut être changé, sans avoir à recharger la page avec le titre, les images, le menu, etc.

Ajax permet ainsi d'effectuer des traitements sur le poste client (avec JavaScript) à partir d'informations prises sur le serveur. Cela répartit la charge de traitement. Auparavant, toutes les modifications de pages étaient faites sur le serveur ce qui nécessitait des échanges maintenant inutiles.

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.

Dynamic HTML est aussi un ensemble de techniques, qui comprend: HTML, CSS, JavaScript.

Cela permet de modifier le contenu d'une page selon les commandes de l'utilisateur, à partir de données préalablement fournies ou avec un texte tapé par l'utilisateur.

Ajax est DHTML plus l'objet XHR pour communiquer avec le serveur.

Comment cela fonctionne?

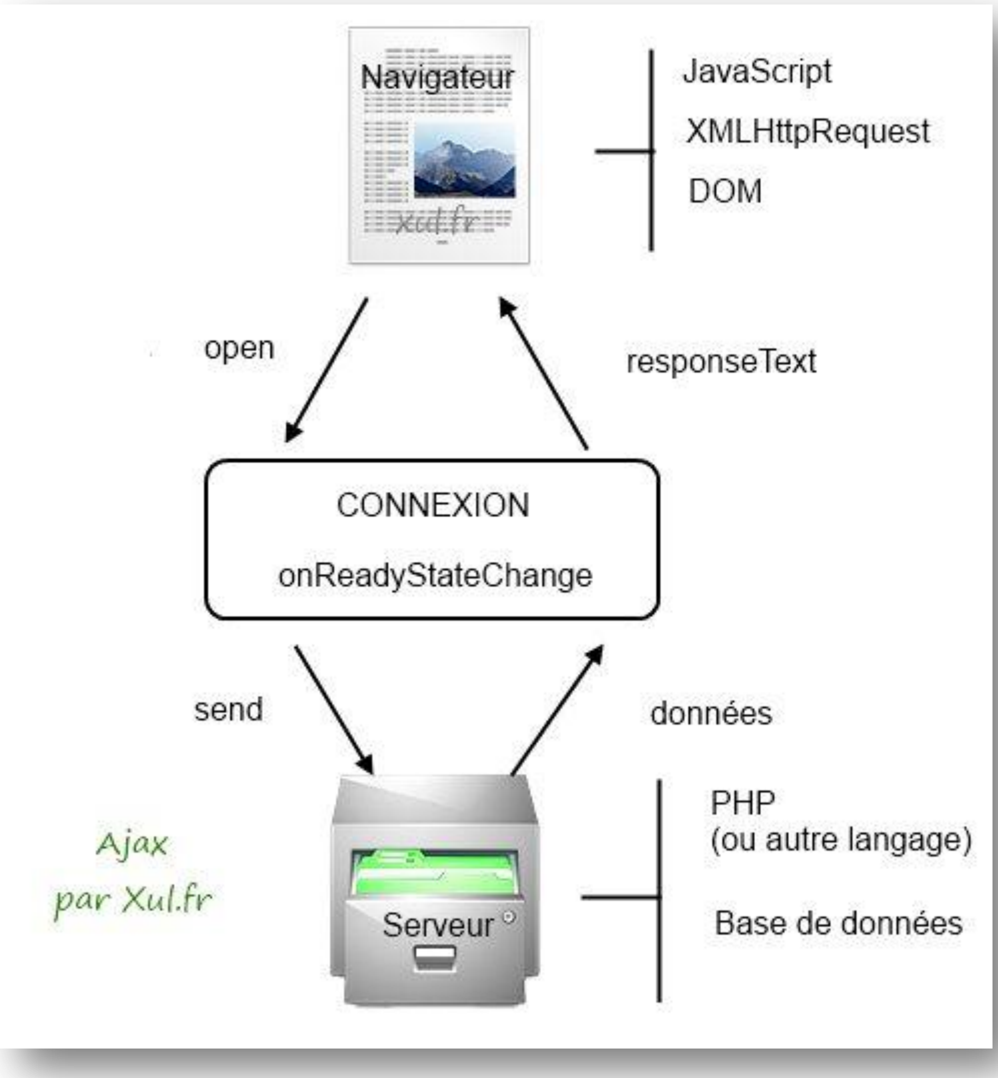

Schéma du fonctionnement d'Ajax

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

Les évènements sont les actions de l'utilisateur, qui provoquent l'appel des fonctions associées aux éléments de la page.

L'interaction avec l'utilisateur se fait à partir des formulaires ou boutons html.

Ces 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 dans les champs de l'objet XMLHttpRequest:

- **responseXml** pour un fichier XML ou
- **responseText** pour un fichier de texte bru.

Noter qu'il faut créer un nouvel objet XMLHttpRequest, pour chaque fichier que vous voulez charger.

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.
```

L'objet XMLHttpRequest

Elle 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 ou 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("chaine")	null pour une commande GET.

Construire une requête, pas à pas

Première étape: créer une instance

C'est juste une instance de classe classique mais deux options à essayer pour compatibilité avec les navigateurs.

```
if (window.XMLHttpRequest)    //  Objet standard
{
    xhr = new XMLHttpRequest();     //  Firefox, Safari, ...
}
else  if (window.ActiveXObject)      //  Internet Explorer
{
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
```

ou plus simplement, on peut utiliser les exceptions:

```
try
{
   xhr = new ActiveXObject("Microsoft.XMLHTTP"); // Essayer IE
}
catch(e)   // Echec, utiliser l'objet standard
{
  xhr = new XMLHttpRequest();
}
```

Seconde étape: attendre la réponse

Le traitement de la réponse et les traitements qui suivent sont inclus dans une fonction, et la valeur de retour de cette fonction sera assignée à l'attribut **onreadystatechange** de l'objet précédemment créé.

```
xhr.onreadystatechange = function()
{
 // instructions de traitement de la réponse
};
```

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

Troisième étape: faire la requête elle-même

Deux méthodes de XMLHttpRequest sont utilisées:

- **open**: commande GET ou POST, URL du document, true pour asynchrone.

- **send**: avec POST seulement, données à envoyer au serveur.

La requête ci-dessous lit un document sur le serveur.

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

Exemple de programme Ajax

Lire un texte

```
<html>
<head>
<script>
function submitForm()
{
   var xhr;
   try {  xhr = new ActiveXObject('Msxml2.XMLHTTP');   }
   catch (e)
   {
      try {   xhr = new ActiveXObject('Microsoft.XMLHTTP'); }
      catch (e2)
      {
        try {  xhr = new XMLHttpRequest();  }
```

```
            catch (e3) {  xhr = false;   }
          }
      }

      xhr.onreadystatechange  = function()
      {
         if(xhr.readyState  == 4)
         {
          if(xhr.status  == 200)
              document.ajax.dyn="Received:"  + xhr.responseText;
          else
              document.ajax.dyn="Error code " + xhr.status;
          }
      };

     xhr.open( GET", "data.xml",  true);
     xhr.send(null);
  }
```

```
    </script>
    </head>

    <body>
       <FORM method="POST" name="ajax" action="">
          <INPUT type="BUTTON" value="Submit"  ONCLICK="submitForm()">
          <INPUT type="text" name="dyn"  value="">
       </FORM>
    </body>
    </html>
```

Syntaxe de formulaire utilisant Ajax

Commentaires sur le code:

new ActiveXObject(Microsoft.XMLHTTP)

Ce constructeur est pour Internet Explorer.

new XMLHttpRequest()

Ce constructeur est pour tout autre navigateur incluant Firefox.

http.onreadystatechange

On associe un traitement (une fonction anonyme en l'occurrence) à cet indicateur d'évènement.

http.readyState == 4

L'état 4 signifie que la réponse est envoyée par le serveur et disponible.

http.status == 200

Ce status signifie ok, sinon un code d'erreur quelconque est envoyé, 404 par exemple.

h**ttp.open("POST", "data.xml", true);**

- POST ou GET
- url du fichier.
- true pour asynchrone (false pour synchrone).

http.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");

Cette méthode s'utilise seulement avec POST.

http.send(document.getElementById("TYPEDTEXT").value);

Envoi des données au serveur. Les données sont prises dans le champ dont l'id est "TYPEDTEXT" et qui est rempli par l'utilisateur grâce à un formulaire.

Démonstration

Reçu:

Lire dans un fichier XML

Pour lire des données dans un fichier XML il suffit de remplacer la ligne:

```
document.ajax.dyn="Reçu: " + xhr.responseText;
```

par ce code:

```
//  Assigner le fichier XML à une variable
var doc = xhr.responseXML;
//  Lire le premier élément avec DOM
var element = doc.getElementsByTagName('root').item(0);
//  Copier le contenu dans le formulaire
document.ajax.dyn.value= element.firstChild.data;
```

Démonstration

Reçu:

Ecrire dans la page

Le texte récupéré est inscrit dans le corps de la page et non dans un champ de texte. Le code ci-dessous remplace l'objet de formulaire textfield et la seconde partie remplace l'assignement dans la fonction JavaScript.

```
<div id="zone">
 <span class="Style1">... un texte à remplacer...</span>
 </div>
```

```
document.getElementById("zone").innerHTML = "Received:" + xhr.responseText;
```

Démonstration

Attendre...

Envoyer un texte

Un texte est envoyé au serveur et sera écrit dans un fichier. L'appel de la méthode "open" change, le premier argument est POST, le second est le nom du fichier ou du script qui recevra les données et doit les traiter. Et la méthode "send" aussi a pour argument une valeur qui est une chaîne de paramètres.

```
xhr.open("POST", "ajax-post-text.php", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.send(data);
```

L'argument de send est au format des paramètres de la méthode POST. S'il y a plusieurs données, on les sépare par le symbole "et" commercial:

```
var data = "file=" + url +  "&content=" + content;
```

Le paramètre "file" est un fichier qui contiendra le contenu "content". Le nom du fichier doit être vérifié par le serveur pour éviter qu'un autre fichier ne puisse être modifié.

Une démonstration de POST est incluse dans l'archive.

Utiliser un fichier externe

Il est plus simple d'inclure un fichier JavaScript. Cette ligne sera incluse dans la section head de la page HTML:

```
<script src="ajax.js" type="text/javascript"></script>
```

Et la fonction est appelée avec cette instruction:

```
var xhr = createXHR();
```

Le script du fichier externe:

```
function createXHR()
{
  var request = false;
  try
  {
    request = new ActiveXObject('Msxml2.XMLHTTP');
  }
  catch (err2)
  {
  try {
      request = new ActiveXObject('Microsoft.XMLHTTP');
  }
  catch (err3) {
      try {  request = new XMLHttpRequest();}
      catch (err1) { request = false;}
   }
  }
 return request;
}
```

Comment faire un site Ajax?

Il faut une interface prédéfinie, comme celles qui sont indiquées dans les liens plus bas.
Puis votre programme JavaScript, intégré dans une page web, et accédant par leur nom ou leur identifieur aux éléments de la page, envoie des requêtes au serveur pour obtenir un fichier.
Celui-ci est exploité par les méthodes de DOM, et la page mise à jour sur le poste client, de façon fluide et instantanée.

Inconvénients d'Ajax

- Si JavaScript est désactivé, Ajax ne peut fonctionner. Il faut demander au lecteur de l'activer parmi les options du navigateur.
- 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.
- 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.
- Le bouton de retour en arrière peut se trouver désactivé (ce n'est pas le cas dans les exemples fournis ici). Cela peut être corrigé.