Table des matières

Bibliographie :

- http://fr.wikipedia.org/wiki/Asynchronous_JavaScript_and_XML
- http://www.xul.fr/xml-ajax.html
- http://ajax.developpez.com/
- http://www.xul.fr/XMLHttpRequest.html

Liste des traductions françaises des recommandations du W3C :

- http://www.w3.org/Consortium/Translation/French

Ressources :

- jQuery : http://docs.jquery.com/
- script.aculo.us : http://script.aculo.us/

Séquence 1 –L'objet XMLHttpRequest

Objectif : afficher l'heure locale (du client) et l'heure distante (du serveur) en utilisant l'objet `XMLHttpRequest`.

1 . A partir d'un éditeur de texte, créer la page web **heure.html** qui permettra l'affichage de l'heure locale (du client) et de l'heure distante (du serveur) en utilisant la technologie Ajax.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
   <head>
      <title>Exemple n°1 : Ajax (XMLHttpRequest)</title>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <meta http-equiv="Pragma" content="no-cache">
      <link rel="stylesheet" media="screen" type="text/css" title="design" href="heure.css">
      <script type="text/javascript" src="heure.js"> </script>
   </head>
   <body>
      <h1>Exemple n°1 : Ajax (XMLHttpRequest)</h1>
      <div id="htm">
        <span style="cursor: pointer; text-decoration: underline"
onclick="requete()">Demande heure</span>
      </div>
      <div id="client"><!-- heure locale --></div>
      <div id="serveur"><!-- heure distante --></div>
</body>
</html>
```

2 . Créer le script **heure.js** qui définit la fonction `requete()`.

```
function requete()
{
   // Affichage de l'heure locale (le client)
   var d = new Date(); var h = d.getHours(); var m = d.getMinutes(); var s = d.getSeconds();
   document.getElementById('client').innerHTML = ("Heure (client) : " + h +':'+ m +':'+ s);

   // Affichage de l'heure distante (le serveur )
   if(window.XMLHttpRequest) // Firefox
      xhr_object = new XMLHttpRequest();
   else if(window.ActiveXObject) // IE
      xhr_object = new ActiveXObject("Microsoft.XMLHTTP");
   else
   { document.getElementById('client').innerHTML = "Erreur : no XMLHttpRequest !"; return; }

   /* 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).
    */
   xhr_object.open("POST", "heure.php", false);

   xhr_object.send(null);
```

```
   /* Les états de readyState sont les suivants :
      0: non initialisé.
      1: connexion établie.
      2: requête reçue.
      3: réponse en cours.
      4: terminé (le seul vraiment utile) */
   if(xhr_object.readyState == 4)
        document.getElementById('serveur').innerHTML = xhr_object.responseText; // Affichage
   else document.getElementById('serveur').innerHTML = "Erreur : réponse serveur !";
}
```

3 . A partir d'un éditeur de texte, créer le script **heure.php** qui fournit l'heure du serveur.

```
<?php
   $Lheure = @date("H:i:s"); // @ désactive les warnings
   echo "Heure (serveur) : $Lheure\n"; // envoie la réponse
?>
```

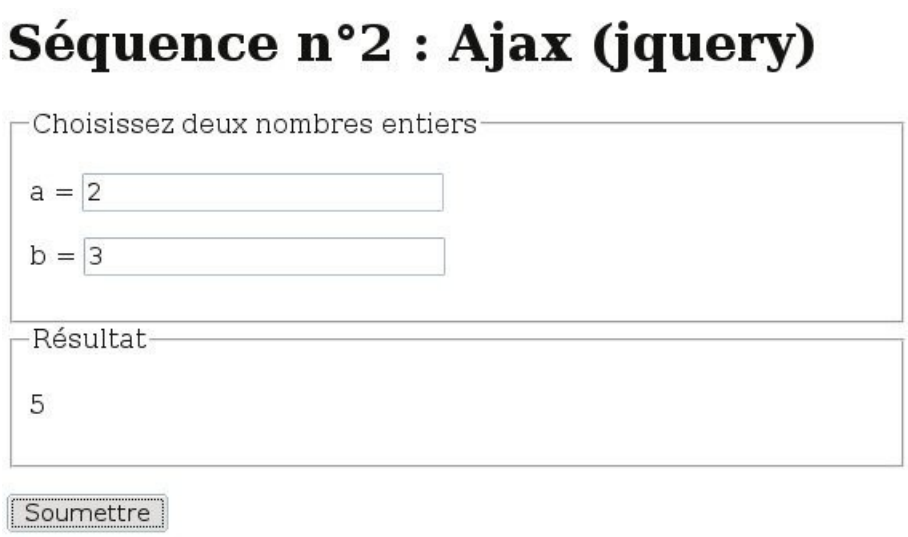
4 . Tester (attention l'accès au document se fait à partir d'une URL vers le serveur du type : **http://...**).

5 . Modifier le(s) fichier(s) afin d'assurer en plus l'affichage de la date au format jj/mm/AAAA.

Séquence 2 –L'API jquery

Objectif : saisir deux nombres et demander au serveur d'assurer le calcul d'addition et de renvoyer le résultat en utilisant l'API `jquery` (**`http://docs.jquery.com/`**).

Remarque : **jQuery est une bibliothèque JavaScript libre** qui porte sur l'interaction entre JavaScript (comprenant **AJAX**) et HTML, et a pour but de simplifier des commandes communes de JavaScript.



1 . Créer la page web **add.html** qui permettra la saisie des 2 nombres et l'affichage du résultat en utilisant la technologie Ajax.

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="UTF-8">
    <title>Exemple n°2 : Ajax (jquery)</title>
    <link rel="stylesheet" media="screen" href="add.css">
    <script src="jquery-1.6.4.min"></script><!-- librairie JQuery : http://docs.jquery.com/
-->
    <script src="add.js"></script> <!-- La source qui contient le code d'envoi en Ajax -->
  </head>
  <body>
    <h1>Exemple n°2 : Ajax (jquery)</h1>
    <form method="post" action="add.php"> <!-- Formulaire envoyé par la méthode POST -->
      <fieldset>
        <legend>Choisissez deux nombres entiers</legend>
        <p><label>a = <input name="a" type="number" required></label></p>
        <p><label>b = <input name="b" type="number" required></label></p>
      </fieldset>
      <fieldset>
        <legend>Résultat</legend>
        <p id="result"><!-- Le résultat sera placé ici --></p>
      </fieldset>
      <p><button>Soumettre</button></p> <!-- Bouton de soumission -->
    </form>
  </body>
</html>
```

2 . Créer le script **add.js** qui utilise l'API **`jquery`**.

```
$(document).ready(OnReady); // Abonne le callback à exécuter lorsque tout le DOM est chargé

function OnReady()
{
    $("form").submit(OnSubmit); // Abonne un callback à l'évènement "submit" du formulaire
}

function OnSubmit(data)
{
    $.ajax(
      {
        type: $(this).attr("method"), // Récupère la méthode d'envoi du formulaire, ici
                                                "POST"
        url: $(this).attr("action"), // Récupère l'url du script qui reçoit la requête, ici
                                                "add.php"
        data: $(this).serialize(), // Fabrique la "query string" contenant les deux nombres
        success: OnSuccess         // Callback qui récupère la réponse du serveur
      });
    return false; // Annule l'envoi classique du formulaire
}

function OnSuccess(result)
{
    $("#result").html(result); // Insère le résultat dans la balise d'id "result"
}
```

3 . Créer le script **add.php** qui assure le calcul et renvoi le résultat.

```
<?php
   print($_POST["a"] + $_POST["b"]); // Envoi au client le résultat du calcul de a + b
?>
```

4 . Tester.

5 . Modifier le(s) fichier(s) afin d'assurer les 4 opérations arithmétiques (addition, soustraction, multiplication et division).

Séquence 3 - XML

Objectif : traiter des données au format XML en provenance du serveur et en utilisant l'objet `XMLHttpRequest`.

Remarque : voir le cours sur XML.

1 . Créer le fichier **dvd.xml** qui fournit la liste d'items suivants :

```
<?xml version="1.0" encoding="utf-8"?>
<dvd>
    <item>
        <title>Princesse Mononoké</title>
        <director>Hayao Miyazaki</director>
        <price>14,99</price>
    </item>
    <item>
        <title>Mon voisin Totoro</title>
        <director>Hayao Miyazaki</director>
        <price>19,99</price>
    </item>
    <item>
        <title>Blood, The Last Vampire</title>
        <director>Hiroyuki Kitakubo</director>
        <price>11,99</price>
    </item>
</dvd>
```

2 . Créer la page web **dvd.html** qui permettra l'affichage d'une liste de DVD en utilisant la technologie Ajax.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
   <head>
      <title>Exemple n°3 : Ajax (XML)</title>
      <meta http-equiv="content-type" content="text/html; charset=UTF-8">
      <meta http-equiv="Pragma" content="no-cache">
      <link rel="stylesheet" media="screen" type="text/css" title="design" href="dvd.css">
      <script type="text/javascript" src="dvd.js"> </script>
   </head>
   <body>
      <h1>Exemple n°3 : Ajax (XML)</h1>
      <div id="header">
        <span style="cursor: pointer; text-decoration: underline"
onclick="requete()">Requête liste DVD</span>
      </div>
      <div id="content"><!-- Le résultat sera placé ici --></div>
   </body>
</html>
```

3 . Créer le script **dvd.js** qui permet notamment de traiter les données XML reçues du serveur.

```
function handleHttpResponse() {
      if(xhr.readyState == 4 && xhr.status == 200)
      {
         /* xhr.responseXML permet d'obtenir le fichier XML
            xhr.responseText aurait retourné le fichier sous format texte */
         response     = xhr.responseXML.documentElement;

         /* Récupérer la liste des items donc la liste des DVD */
         var items    = response.getElementsByTagName("item");
         var html     = ''; /* Présentation HTML de la liste des DVD */
         /* Nombre de DVD */
         count        = items.length;
         for(i = 0; i < count; i++) /* POUR CHAQUE item */
         {
            html += '<div class="item">';
            html += '<h1 class="title">' + items[i].getElementsByTagName("title")[0
].firstChild.nodeValue + '</h1>';
            html += '<div class="author">R&eacute;alisateur : ';
            html += items[i].getElementsByTagName("director")[0].firstChild.nodeValue +
'</div>';
            html += '<div class="price">Prix : ';
            html += items[i].getElementsByTagName("price")[0].firstChild.nodeValue + '
€</div>';
            html += '</div>';
         }
         document.getElementById('content').innerHTML += html;
      }
}

function getXMLHttpRequest() {
   if(window.XMLHttpRequest) // Firefox
      xhr_object = new XMLHttpRequest();
   else if(window.ActiveXObject) // IE
      xhr_object = new ActiveXObject("Microsoft.XMLHTTP");
   else
   {
      alert("Erreur : no XMLHttpRequest !"); return;
   }

   return xhr_object;
}
```

```
function requete()
{
   xhr = getXMLHttpRequest();

   xhr.onreadystatechange = handleHttpResponse;

   var url = "dvd.xml";

   xhr.open("GET", url, true);

   /* A préciser pour les requêtes de type POST
      xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
   */

   /* Exemple pour POST : xhr.send("var1=value&var2=othervalue"); */
   xhr.send(null);
}
```

4 . Tester.

5 . Modifier la fonction `handleHttpResponses()` afin qu'elle n'affiche qu'un simple fichier texte **dvd.txt** :

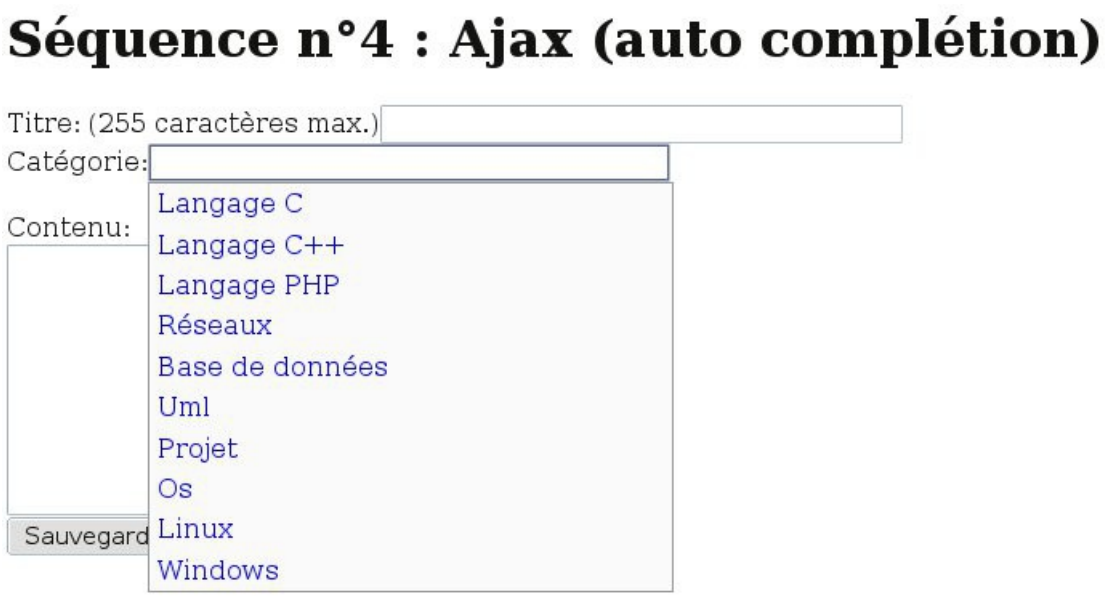
```
Princesse Mononoké de Hayao Miyazaki Prix : 14,99
Mon voisin Totoro de Hayao Miyazaki Prix : 19,99
Blood, The Last Vampire de Hiroyuki Kitakubo Prix : 11,99
```

6 . Qu'apporte le format XML par rapport à un simple fichier texte ?

Séquence 4 - L'auto-complétion

Objectifs : fournir la fonction d'auto-complétion lors d'une saisie dans un formulaire. Ce formulaire permet de publier un article lié à une catégorie connue du serveur. L'auto-complétion sera mise en oeuvre pour la **catégorie**. Dans cette séquence, on utilisera l'API **`scriptaculous`**. Lien : http://script.aculo.us/

Pour en savoir plus : Un article du site developpez.com décrit pas à pas l'utilisation d'Ajax pour réaliser l'auto-complétion. Lien : http://dcabasson.developpez.com/articles/javascript/ajax/ajax-autocompletion-pas-a-pas/



1 . Créer une page web **auto.html** permettant de réaliser l'affichage d'un formulaire de saisie d'un nouvel article.

```
<!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" lang="fr">
<head>
   <title>Exemple n°4 : Ajax (auto complétion)</title>
   <meta http-equiv="Content-type" content="text/html; charset=UTF-8" />
   <meta http-equiv="Content-Script-Type" content="text/javascript" />
   <meta http-equiv="Content-Style-Type" content="text/css" />
   <link rel="stylesheet" type="text/css" href="auto.css" media="screen" title="Normal" />
   <script src="./scriptaculous/prototype.js" type="text/javascript"></script>
   <script src="./scriptaculous/effects.js" type="text/javascript"></script>
   <script src="./scriptaculous/dragdrop.js" type="text/javascript"></script>
   <script src="./scriptaculous/controls.js" type="text/javascript"></script>
   <script src="./scriptaculous/scriptaculous.js" type="text/javascript"></script>
</head>
<body>
   <h1>Exemple n°4 : Ajax (auto complétion)</h1>
   <form action="#" method="POST" name="article" id="article">
      <input type="hidden" name="op" id="op" value="Annuler" style="display:none;" />
      <input type="hidden" name="type" id="type" value="article" style="display:none;" />
      <label for="titre">Titre: (255 caractères max.)</label><input type="text" name="titre"
id="titre" value="" maxlength="255" size="30" />
      <br style="clear: both;" />
      <label for="categorie">Catégorie:</label><input type="text" name="categorie"
id="categorie" value="" maxlength="255" size="30" />
      <br style="clear: both;" />
      <div id="cat_update"><!-- l'affichage de l'auto-complétion se fera ici --></div>
      <br style="clear: both;" />
```

```
      <label for="contenu">Contenu:</label><br style="clear: both;" />
      <textarea name="contenu" id="contenu" cols="35" rows="10"
wrap="virtual"></textarea><br style="clear: both;" />

      <script type="text/javascript">
         new Ajax.Autocompleter (
            'categorie',
            'cat_update',
            'auto.php',
            {method: 'post', paramName: 'categorie', minChars: 0}
         );
      </script>

      <input type="button" name="b1" id="b1" value="Sauvegarder" class="exo"
onClick="document.article.op.value='Sauvegarder';  document.article.submit();" />
      <input type="button" name="b2" id="b2" value="Annuler" class="exo"
onClick="document.article.op.value='Annuler';  document.article.submit();" />
   </form>
</body>
</html>
```

2 . Créer un script **auto.php** qui fournit les catégories en fonction de la saisie déjà réalisée :

```
<?php

// Une base de donnée (MySQL) pourrait être aussi utilisée ici ...
// Le libellé des catégories connues du serveur :
$libelles = array("langage C", "langage C++", "langage PHP", "réseaux", "base de données",
"uml", "projet", "os", "linux", "windows");

$categorie = $_POST['categorie']; // la saisie du client

$i = 0;

echo '<ul>';
foreach($libelles as $libelle)
{
   if (substr(strtolower($libelle),0,strlen($categorie)) ==
strtolower(stripslashes($categorie)))
   {
      echo '<li><a href="#" onclick="return false">'.ucfirst($libelle).'</a></li>';
      if (++$i > 11) die('<li>...</li></ul>'); // une limite de 10 !
   }
}
echo '</ul>';

?>
```

3 . Tester.

4 . Modifier le(s) fichier(s) pour gérer les catégories suivantes : "programmation", "base de données", "méthodologie" et "os". Puis ajouter un nouveau champ de saisie « **Thème** » et mettre en place l'auto-complétion pour ce champ avec les thèmes suivants pour chaque catégorie :

- "programmation" → "langage C", "langage C++", "langage PHP"
- "base de données" → "sqlite ", "MySQL"
- "méthodologie" → "uml", "projet"
- "os" → "linux", "windows"