Module 15 : AJAX Manipulation du DOM

1. **Introduction**

* Tel que vu précédemment, nous avons la possibilité de récupérer avec l’objet XMLHttpRequest plusieurs formats de données (text, XML, HTML...).
* Avec **DOMImplementation** nous avons la possibilité de récupérer directement des données au format XML
* L'interface **DOMImplementation** est supportée par la plupart des navigateurs.
* Internet Explorer supporte le **DOMImplementation**, mais par le biais d'un contrôle ActiveX (comme les versions antérieures au 7 d'IE avec XMLHttpRequest).
* Internet Explorer gère le **DOMImplementation** que partiellement. Il ne gère pas **createDocument** dont nous allons nous servir. Donc pour savoir comment réagir en fonction du navigateur, il faut tester la prise en charge de l'objet **implementation**, ainsi que la prise en charge de la méthode **createDocument,** tel que ci-dessous :

if (document.implementation && document.implementation.createDocument) { }

1. **Exemples d’utilisation :**

L’exemple ci-dessous créer une fonction générique qui ira récupérer une source XML à l'adresse sUrl, et qui renverra le contenu dans une fonction de callback fCallback.

function getDOMImplementation(sUrl, fCallback) {

var dom;

if (window.ActiveXObject) {

dom = new ActiveXObject("Microsoft.XMLDOM");

dom.onreadystatechange = function() {

if(dom.readyState == 4) {

fCallback(dom);

}

};}

else if (document.implementation && document.implementation.createDocument) {

dom = document.implementation.createDocument("", "", null);

dom.onload = function() {

fCallback(dom);

}

}

else {

alert("Votre navigateur ne gère pas l'importation de fichiers XML");

return;

}

dom.load(sUrl);}

* L'objet **implementation** requiert la méthode **createDocument** qui permettra de créer un document XML.
* Les deux premiers arguments de la méthode **createDocument** sont le **namespace** et le nom qualifié du document XML.
* Le troisième argument est le **doctype**. Laissez les deux premiers vides, et le troisième null.
  1. **Affichage le contenu d’un XML statique**

L’exemple suivant charge le fichier XML ci-dessous et en lit une entrée

<?xml version="1.0" encoding="iso-8859-1"?>

<topic>

<author>Thunderseb</author>

<country>Belgium</country>

<lang>French</lang>

</topic>

Contenu de la page HTML

<!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">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<link href="mep.css" rel="stylesheet" type="text/css" media="all" />

<title>Techniques AJAX - DOMImplementation - XML Statique</title>

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

<script type="text/javascript">

<!--

function getData(oData) {

var author = oData.getElementsByTagName('author')[0].firstChild.data;

var country = oData.getElementsByTagName('country')[0].firstChild.data;

var lang = oData.getElementsByTagName('lang')[0].firstChild.data;

alert(author + ' - ' + country + ' - ' + lang);

}

//-->

</script>

</head>

<body>

<h1>Techniques AJAX - DOMImplementation - XML Statique </h1>

<p>

<input type="button" value="Récupérer les données" onclick="getDOMImplementation('DOMImplementation\_1.xml', getData);" />

</p></body></html>

Au clique sur le bouton "**Récupérer les données**", le programme affiche le contenu des balises **author**, **country** et **lang**:



* 1. **Affichage du contenu d’un XML dynamique**
* L’exemple ci-dessous utilise une page PHP pour générer un fichier XML.
* Les variables GET sont transmises à la page PHP via l’url.
* La page HTML reste la même que dans l'exemple précèdent, à l'exception de l'appel de la fonction. Tel que ci-dessous :

<input type="button" value="Récupérer les données" onclick="getDOMImplementation(**'DOMImplementation\_2.php**?Pseudo=Thunderseb', getData);" />

La page PHP permettant de générer le XML dynamiquement est tel que ci-dessous :

<?php

header("Content-Type: text/xml");

echo "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>";

echo "<topic>";

if (isset($\_GET['Pseudo'])) {

$pseudo = $\_GET['Pseudo'];

if ($pseudo == 'Thunderseb') {

echo '<author>Thunderseb</author>';

echo '<country>Belgium</country>';

echo '<lang>French</lang>';

} else if ($pseudo == 'Laurence') {

echo '<author>Laurence</author>';

echo '<country>Belgium</country>';

echo '<lang>French</lang>';

}

}

echo "</topic>";

?>

* 1. **Gestion des données**
* L’exemple ci-dessous réalise une petite base de données (un fichier XML) laquelle est lue par un script JavaScript.
* L’idée du script est d’afficher via une liste déroulante le contenu de la BD (bandes dessinées), les différents albums d'une série (tous les albums de la BD *Pierre Tombal* par exemple).

Nous avons ci-dessous le contenu du fichier XML :

<?xml version="1.0" encoding="utf-8"?>

<bds>

<!-- PIERRE TOMBAL -->

<serie name="Pierre Tombal">

<bd num="4" title="des os pilants" />

<bd num="5" title="ô suaires" />

<bd num="12" title="la pelle aux morts" />

<bd num="21" title="k.os" />

<bd num="22" title="ne jouez pas avec la Mort !" />

<bd num="24" title="on s'éclate mortels !" />

</serie>

<!-- LES PROFS -->

<serie name="Les Profs">

<bd num="2" title="loto et colles" />

<bd num="3" title="tohu-bahut" />

<bd num="4" title="rentrée des artistes" />

<bd num="5" title="chute des cours" />

<bd num="9" title="rythme scolaire" />

</serie>

<!-- et ça continue, encore et encore, c'est que le début, d'accord d'accord... -->

</bds>

* L'idée est de récupérer le contenu XML, via **DOMImplementation** et de lister toutes les séries, et les afficher dans une liste déroulante.
* A partir de cette liste, l'utilisateur choisit une série et clique sur un bouton pour afficher les albums de cette série.
* Ci-dessous le contenu de la page HTML

<fieldset>

<legend>Sélection de la série</legend>

<form id="bdForm" method="get" action="">

<div>

<label for="serieName">Choisir une série : </label>

<!-- On écrira le SELECT dans le SPAN ci-dessous -->

<span id="outputListDiv"><em>Liste des séries non chargée...</em></span>

</div>

<p><input type="button" value="Afficher les BD de cette série" onclick="displayBD();" /></p>

</form>

</fieldset>

<fieldset id="outputBDFieldset">

<legend>BD en ma possession</legend>

<!-- On écrira la TABLE avec la liste des albums dans le DIV ci-dessous -->

<div id="outputBDDiv"></div>

</fieldset>

Ci-dessous le début du script :

var DOMXML = null;

window.onload = function() {

DOMImplementation('data.xml', getData);

}

* **DOMXML** est une variable globale dans laquelle on mettra la structure XML, de manière à l'exploiter par la suite.
* La fonction d’initialisation est appelée lors du chargement (onload) et se charge de lire le fichier XML et d'appeler la fonction de callback **getData**.

Ci-dessous le contenu de la fonction **getData** :

function getData(oData) {

DOMXML = oData;

var series = oData.getElementsByTagName("serie");

var sorted = [];

// Classement par ordre alphabétique, via tableau à 2 dimensions

for (var i=0, c=series.length; i<c; i++)

{

sorted.push([series[i].getAttribute("name") , i]);

}

sorted.sort();

// Génération du SELECT

var list = "<select name=\"serieName\" id=\"serieName\">\n";

for (var i=0, c=sorted.length; i<c; i++) {

list += "<option value=\"" + sorted[i][1] + "\">" + sorted[i][0] + "</option>\n";

}

list += "</select>\n";

document.getElementById("outputListDiv").innerHTML = list;



}

* On commence par récupérer tous les éléments **<serie>**, et pour chaque élément, on récupère le **nom de la série** ainsi que l'indice de parcourt **(i)** que l'on met dans un tableau. Le tableau est ajouté dans le tableau **sorted**.
* On applique un **sort** sur le tableau **sorted** pour trier les sous-tableaux par ordre alphabétique.
* Une fois que le tri est fait, on reboucle pour créer le **<select>** que l'on ajoute dans la page.
* Avec ça, on récupère les séries et on affiche le **<select>** .
* Maintenant, quand l'utilisateur clique sur un bouton, on affiche la liste des albums contenus dans la série sélectionnée.

<input type="button" value="Afficher les BD de cette série" onclick="displayBD();" />

* Ci-dessous le contenu de la fonction **displayBD**

function displayBD() {

// récupération de l'id de la série, écrit dans la VALUE de l'OPTION

with (document.getElementById("serieName")) var idSerie = parseInt(options[selectedIndex].value);

var serie = DOMXML.getElementsByTagName("serie")[idSerie];

var bds = serie.getElementsByTagName("bd");

// Création de la TABLE avec les résultats

var table = "<table class=\"sortable\" width=\"100%\">\n";

table += "<tr><th width=\"20\">N°</th><th>Titre de l'album</th></td>\n";

for (var i=0, c=bds.length; i<c; i++) {

table += "<tr><td>" + bds[i].getAttribute("num") + "</td><td>" + bds[i].getAttribute("title") + "</td></tr>\n";

}

table += "</table>\n"; document.getElementById("outputBDDiv").innerHTML = table;

}

* Les données XML sont dans la variable globale **DOMXML**
* On se sert de la variable **DOMXML** pour récupérer les albums (variable **bds**) de la série donnée.
* On affiche le contenu dans un tableau

En Résumé :

En comparant le DOMImplementation par rapport XMLHttpRequest nous sortons avec les remarques ci-dessous :

#### **Points forts du DOMImplementation**

* Plus facile à mettre en place que XMLHttpRequest
* Adapté pour le XML

#### **Points faibles du DOMImplementation**

* Non supporté par tous les navigateurs
* Seulement des requêtes GET

**Mise en Pratique :**

Créer une interface graphique de votre choix permettant d’afficher à partir **d’un titre de livre** l’auteur et le nombre de pages et ce en se basant le document XML ci-dessous :

<?xml version="1.0"?>

<livres>

<livre>

<titre>Guerre et Paix</titre>

<auteur>Léon TolstoÏ</auteur>

<nombrepages>1572</nombrepages>

</livre>

<livre>

<titre>La Dame aux camélias</titre>

<auteur>Alexandre Dumas</auteur>

<publication>

<editeur>Lebègue</editeur>

<dateparution>1848</dateparution>

</publication>

</livre>

</livres>

* Créer une première version avec XMLHttpRequest
* Créer une première version avec DOMImplementation

**Liste des références :**

Sébastien de la Marck et Johann Pardanaud. Créez des pages web interactives avec

JavaScript. Repéré au : https://openclassrooms.com/courses/simplifiez-vosdeveloppements-javascript-avec-jquery/plus-loin-dans-la-selection-d-elements