Récupérer et traiter du XML

1. Par les nœuds

Soit le fichier cesar2014.xml :

```
<?xml version="1.0" encoding="UTF-8"?>
<cinema>
<evenement>Cérémonie des Césars</evenement>
<sujet>Palmarès</sujet>
<date>2014</date>
<cesar>
<categorie>
<prix>César du meilleur acteur</prix>
<nom>Guillaume Gallienne</nom>
<film>Les garçons et Guillaume, à table !</film>
</categorie>
<categorie>
<prix>César du meilleur film français</prix>
<nom>Guillaume Gallienne</nom>
<film>Les garçons et Guillaume, à table !</film>
</categorie>
<categorie>
<prix>César du meilleur film étranger</prix>
<nom>Felix Van Groeningen</nom>
<film>Alabama Monroe</film>
</categorie>
</cesar>
</cinema>
```

Sa structure est :

```
<cinema>
            <evenement></evenement
            <sujet></sujet
            <date></date
            <cesar
                        <categorie
                                    <prix></prix
                                    <nom></nom
                                    <film></film
                        </categorie
            </cesar
</cinema>
```

On souhaite, au clic sur un bouton, extraire les données incluses dans la troisième balise `<categorie>`. Soit les informations :

César du meilleur film étranger

Felix Van Groeningen

Alabama Monroe

Le fichier HTML de départ comporte simplement un bouton de formulaire et une balise `<div id="affichage">` pour accueillir la réponse.

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<style>
#affichage { margin-top: 12px;}
</style>
</head>
<body>
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```

Abordons la partie JavaScript.

Pour se prémunir de tous problèmes dans l'interprétation des espaces vides du document XML, nous élaborons une fonction (`enleverespaces()`) qui supprimera ceux-ci. Le script sera ainsi parfaitement compatible avec les différents navigateurs.

Pour commencer, passons en revue, par une boucle `for`, les différents nœuds enfant par la propriété `childNodes`.

```
function enleverespaces(xmldocument) {
var index;
for (index = 0; index < xmldocument.childNodes.length; index++) {
var noeudc = xml.childNodes(index);
...
...
}
}
```

À cette étape de la boucle, le nœud courant est stocké dans la variable `noeud`. S'il s'agit d'un nœud élément (`noeud.nodeType == 1`), cet élément peut, à son tour, avoir des nœuds enfant pour lesquels il faudra enlever les espaces vides. Dans ce cas, on soumet à nouveau ce nœud courant à la fonction `enleverespaces()` en le fournissant comme argument de la fonction, soit `enleverespaces(noeud)`.

```
function enleverespaces(xmldocument) {
var index;
for (index = 0; index < xmldocument.childNodes.length; index++) {
var noeud = xmldocument.childNodes[index];
```

```
if (noeud.nodeType == 1) {
enleverespaces(noeud);
}
...
...
}
}
```

Par ailleurs, si le nœud courant est un nœud texte (`noeud.nodeType == 3`), il s'agit peut-être d'un espace vide. Nous allons vérifier le texte présent dans le nœud (propriété `nodeValue`) au moyen d'une expression régulière qui décèle les espaces vides, soit par (`/^\s+$/.test(noeud.nodeValue)`). Nous allons donc tester si le nœud courant comporte des espaces vides et est un nœud texte (`if ((/^\s+$/.test(noeud.nodeValue)) && (noeud.nodeType == 3))`). Dans l'affirmative, il suffit alors d'appliquer la propriété `removeChild` pour enlever ce nœud. On n'oubliera pas de décrémenter l'index d'une position.

```
function enleverespaces(xmldocument) {
var index;
for (index = 0; index < xmldocument.childNodes.length; index++) {
var noeudc = xmldocument.childNodes[index];
if (noeudc.nodeType == 1) {
enleverespaces(noeudc);
}
if ((/^\s+$/.test(noeudc.nodeValue)) && (noeudc.nodeType == 3)) {
xmldocument.removeChild(xmldocument.childNodes[index–]);
}
}
}
```

Le script commence par l'initialisation de l'objet XMLHttpRequest ou du contrôle ActiveX correspondant.

```
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
```

Si l'objet `xhr` existe, le script lance la requête HTTP en veillant à utiliser cette fois la propriété `responseXML` pour récupérer le fichier césar2014.xml. Celui-ci est alors transmis à la fonction `afficher()`.

```
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
enleverespaces(xmldocument);
afficher(xmldocument);
}
}
```

```
xhr.open("GET", "cesar2014.xml", true);
xhr.send(null);
}
}
```

Il ne reste plus qu'à se préoccuper de l'affichage.

```
function afficher(xmldocument) {
var cinema, cesar, categorie;
var prix, nom, film, textediv;
cinema = xmldocument.documentElement;
cesar = cinema.lastChild;
categorie = cesar.lastChild;
prix = categorie.firstChild;
nom = prix.nextSibling;
film = categorie.lastChild;
textediv = (prix.firstChild.nodeValue + "<br>" + nom.firstChild.nodeValue +
"<br>" +  film.firstChild.nodeValue);
var target = document.getElementById("affichage");
target.innerHTML = textediv;
}
```

Après avoir défini une série de variables, les données de la dernière balise `<catégorie>` du fichier XML sont reprises par les méthodes `firstChild`, `lastChild` et `nextSibling`. La variable `textediv` affichera les différents éléments dans la division prévue à cet effet (`getElementById("affichage")`).

Le fichier HTML complet se présente comme suit :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<script>
function enleverespaces(xmldocument) {
var index;
for (index = 0; index < xmldocument.childNodes.length; index++) {
var noeud = xmldocument.childNodes[index];
if (noeud.nodeType == 1) {
enleverespaces(noeud);
}
if ((/^\s+$/.test(noeud.nodeValue)) && (noeud.nodeType == 3)) {
xmldocument.removeChild(xmldocument.childNodes[index--]);
}
}
}
function afficher(xmldocument) {
var cinema, cesar, categorie;
var prix, nom, film, textediv;
cinema = xmldocument.documentElement;
cesar = cinema.lastChild;
categorie = cesar.lastChild;
prix = categorie.firstChild;
```

```
nom = prix.nextSibling;
film = categorie.lastChild;
textediv = (prix.firstChild.nodeValue + "<br>" +
nom.firstChild.nodeValue + "<br>" +  film.firstChild.nodeValue);
var target = document.getElementById("affichage");
target.innerHTML = textediv;
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n’est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
enleverespaces(xmldocument);
afficher(xmldocument);
}
}
xhr.open("GET", "cesar2014.xml", true);
xhr.send(null);
}
}
</script>
<style>
#affichage { margin-top: 12px;}
</style>
</head>
<body>
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```

2. Par la méthode getElementsByTagName

Le procédé précédent, qui permet d'accéder aux objets par les propriétés des nœuds, est certes élégant, mais il faut bien admettre que les sauts successifs sur les éléments d'un document XML produisent un code complexe. Il se révèle en outre peu pratique dans le cas d'une mise à jour de la page car le moindre élément ajouté ou retiré nécessiterait la réécriture complète du code.

L'utilisation de la méthode `getElementsByTagName` se révèle plus simple et plus fonctionnelle.

Considérons toujours notre document *cesar2014.xml* : il s'agit toujours d'accéder aux informations des balises `<prix>`, `<nom>` et `<film>` de la troisième balise `<categorie>`.

```
<?xml version="1.0" encoding="UTF-8"?>
<cinema>
<evenement>Cérémonie des Césars</evenement>
<sujet>Palmarès</sujet>
<date>2014</date>
<cesar>
<categorie>
<prix>César du meilleur acteur</prix>
<nom>Guillaume Gallienne</nom>
<film>Les garçons et Guillaume, à table !</film>
</categorie>
<categorie>
<prix>César du meilleur film français</prix>
<nom>Guillaume Gallienne</nom>
<film>Les garçons et Guillaume, à table !</film>
</categorie>
<categorie>
<prix>César du meilleur film étranger</prix>
<nom>Felix Van Groeningen</nom>
<film>Alabama Monroe</film>
</categorie>
</cesar>
</cinema>
```

Construisons le script.

```
<script>
function afficher(xmldocument) {
noeudsprix = xmldocument.getElementsByTagName("prix");
noeudsnom = xmldocument.getElementsByTagName("nom");
noeudsfilm = xmldocument.getElementsByTagName("film");
...
...
}
```

Les variables `noeudsprix`, `noeudsnom` et `noeudsfilm` contiennent dans une liste toutes les balises `<prix>`, `<nom>` et `<film>` du document XML. Il suffit alors d'extraire la valeur (`nodeValue`) du premier élément enfant (`firstChild`) du troisième nœud `<prix>` (`noeudsprix[2]`). Ces nœuds `<nom>` et `<film>` sont extraits de la même façon. Une fois ces informations recueillies, la propriété `innerHTML` permet l'affichage de celles-ci.

```
function afficher(xmldocument) {
noeudsprix = xmldocument.getElementsByTagName("prix");
noeudsnom = xmldocument.getElementsByTagName("nom");
noeudsfilm = xmldocument.getElementsByTagName("film");
var textediv = (noeudsprix[2].firstChild.nodeValue + "<br>" +
noeudsnom[2].firstChild.nodeValue + "<br>" +
noeudsfilm[2].firstChild.nodeValue);
var target = document.getElementById("affichage");
target.innerHTML = textediv;
}
```

Le code complet du document HTML avec la méthode `getElementsByTagName` devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le AJAX</title>
<meta charset="UTF-8">
<script>
function afficher(xmldocument) {
noeudsprix = xmldocument.getElementsByTagName("prix");
noeudsnom = xmldocument.getElementsByTagName("nom");
noeudsfilm = xmldocument.getElementsByTagName("film");
var textediv = (noeudsprix[2].firstChild.nodeValue + "<br>" +
noeudsnom[2].firstChild.nodeValue + "<br>" +
noeudsfilm[2].firstChild.nodeValue);
var target = document.getElementById("affichage");
target.innerHTML = textediv;
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
```

```
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
afficher(xmldocument);
}
}
xhr.open("GET", "cesar2014.xml", true);
xhr.send(null);
}
}
</script>
<style>
#affichage { margin-top: 12px;}
</style>
</head>
<body>
<h2>Cérémonie des Césars 2014</h2>
<form>
<input type="button" value="Afficher le César"
onclick="extraire()">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```

Le résultat est identique à celui de l'exploration du document par la propriété des nœuds, mais le code est nettement plus facile à écrire. Celui-ci sera toujours compatible avec les principaux navigateurs que sont Internet Explorer, Google Chrome et Firefox.

3. Traitement des attributs

Un document XML peut également comporter des attributs de balises. Ceux-ci contiennent souvent des informations d'un intérêt non négligeable. Ainsi, il importe de pouvoir également y accéder.

Nous utilisons la propriété `attributes` qui renvoie une liste de tous les attributs d'un élément spécifié. Cette liste des nœuds attributs est renvoyée sous forme d'un objet de type `NamedNodeMap`. Ce qui implique que les attributs sont, par la suite, accessibles par leur nom.

Soit un document XML (stock.xml) qui comporte des attributs :

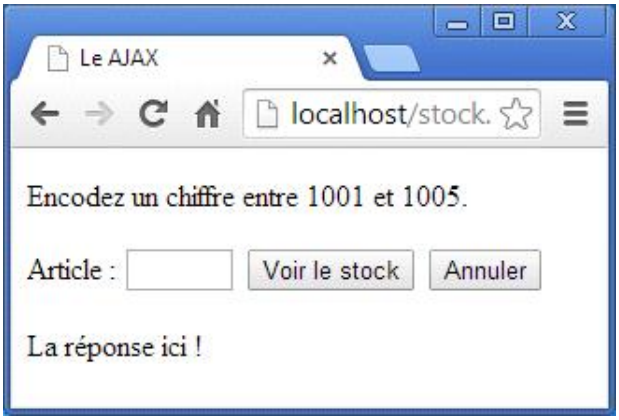
```
<?xml version="1.0" encoding="UTF-8"?>
<stock>
<album>
<article etat="en stock">1001</article>
```

```
<titre>Sarbacane</titre>
<artiste>Francis Cabrel</artiste>
</album>
<album>
<article etat="en stock">1002</article>
<titre>Nickel</titre>
<artiste>Alain Souchon</artiste>
</album>
<album>
<article etat="en rupture">1003</article>
<titre>Sheller en solitaire</titre>
<artiste>William Sheller</artiste>
</album>
<album>
<article etat="en rupture">1004</article>
<titre>Caché derrière</titre>
<artiste>Laurent Voulzy</artiste>
</album>
<album>
<article etat="en stock">1005</article>
<titre>Défoule sentimentale</titre>
<artiste>Alain Souchon</artiste>
</album>
</stock>
```

On souhaite savoir, après avoir consulté le fichier stock.xml, si l'article est en stock ou en rupture.

Le fichier HTML de départ présente la forme suivante :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>Le AJAX</title>
<meta charset="UTF-8">
<style>
#affichage { margin-top: 18px;}
</style>
</head>
<body>
<p>Encodez un chiffre entre 1001 et 1005.</p>
<form>
Article : <input type="text" id="entree" size="4">
<input type="button" value="Voir le stock" onclick="extraire()">
<input type="reset" value="Annuler">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```



Passons à la partie script.

```
<script>
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
```

Au clic sur le bouton, la fonction `extraire()` est appelée. Elle crée un objet `XMLHttpRequest`, ou un objet ActiveX, pour rendre le script compatible.

```
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
afficher(xmldocument);
}
}
xhr.open("GET",<+>"stock.xml", true);
xhr.send(null);
}
}
```

Une requête asynchrone est effectuée sur le fichier stock.xml. Si l'opération s'est déroulée correctement, le script passe la main à la fonction `afficher()`, prenant en argument le fichier XML (`afficher(xmldocument)`).

```
function afficher(xmldocument) {
var entree = document.getElementById("entree").value;
var target = document.getElementById("affichage");
var article = xmldocument.getElementsByTagName("article");
```

Différents objets sont alors déterminés. La variable `entree` reprend le contenu de la ligne de texte. La variable `target` accède à la zone d'affichage définie par la balise `<div id="affichage">` et enfin la variable `article` liste les balises `<article>` du document XML.

```
for (i=0; i<article.length; i++){
if(article[i].firstChild.nodeValue==entree){
var attributs=article[i].attributes;
var etatstock=attributs.getNamedItem("etat").nodeValue;
```

Ensuite, au moyen d'une boucle `for`, les différents éléments de la liste `article` sont passés en revue. Si le numéro de référence (`article[i].firstChild.nodeValue`), contenu dans la balise `<article>`, est égal à la valeur saisie (variable `entree`), une variable `attributs` est créée comprenant tous les attributs de la balise `<article>` retenue. Enfin, il suffit d'aller chercher l'attribut `etat` (`attributs.getNamedItem("etat")`) et d'en prendre la valeur (`nodeValue`).

```
target.innerHTML="Cet article est " + etatstock;
}
}
}
</script>
```

Le script se termine par l'affichage du message.

Le script complet est le suivant :

```
<script>
function afficher(xmldocument) {
var entree = document.getElementById("entree").value;
var target = document.getElementById("affichage");
var article = xmldocument.getElementsByTagName("article");
for (i=0; i<article.length; i++){
if(article[i].firstChild.nodeValue==entree){
var attributs=article[i].attributes;
var etatstock=attributs.getNamedItem("etat").nodeValue;
target.innerHTML="Cet article est " + etatstock;
}
}
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n'est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
```

```
afficher(xmldocument);
}
}
xhr.open("GET", "stock.xml", true);
xhr.send(null);
}
}
</script>
```

Le fichier XHTML final devient :

```
<!DOCTYPE html>
<html lang="fr">
<head>
<title>AJAX</title>
<meta charset="UTF-8">
<script>
function afficher(xmldocument) {
var entree = document.getElementById("entree").value;
var target = document.getElementById("affichage");
var article = xmldocument.getElementsByTagName("article");
for (i=0; i<article.length; i++){
if(article[i].firstChild.nodeValue==entree){
var attributs=article[i].attributes;
var etatstock=attributs.getNamedItem("etat").nodeValue;
target.innerHTML="Cet article est " + etatstock;
}
}
}
var xhr = null;
function extraire(){
if(window.XMLHttpRequest) {
xhr = new XMLHttpRequest();
}
else if(window.ActiveXObject){
xhr = new ActiveXObject("Microsoft.XMLHTTP");
}
else{
alert("Votre navigateur n’est pas compatible avec AJAX...");
}
if(xhr) {
xhr.onreadystatechange = function(){
if(xhr.readyState == 4 && xhr.status == 200){
var xmldocument = xhr.responseXML;
afficher(xmldocument);
}
}
xhr.open("GET", "stock.xml", true);
xhr.send(null);
}
}
</script>
<style>
```

```
#affichage { margin-top: 18px;}
</style>
</head>
<body>
<p>Encodez un chiffre entre 1001 et 1005.</p>
<form>
Article : <input type="text" id="entree" size="4">
<input type="button" value="Voir le stock" onclick="extraire()">
<input type="reset" value="Annuler">
</form>
<div id="affichage">
La réponse ici !
</div>
</body>
</html>
```