Exemples

1. Exemple 1 : Affichage d'un contenu de mail codé en XML

L'objectif dans ce premier exemple va être d'afficher en HTML un flux de données XML récupéré sur un site Web.

Fichier email.xml

Le document en question (`email.xml`) a une structure très simple :

```
<email>
    <de>Olivier VIGOUROUX</de>
    <a>Christian VIGOUROUX</a>
    <entete>Important</entete>
    <message>Partie de badminton samedi à 10 heures</message>
</email>
```

Vous l'aurez compris, il peut s'agir d'un message de type mail (codé ici en XML) entre un émetteur (Olivier VIGOUROUX) et un destinataire (Christian VIGOUROUX).

Section HTML <body>

Dans le script HTML/JavaScript (nommé `DOM_01.htm`), le code de gestion du flux XML est intégralement intégré dans la section HTML `<body>`.

Le script débute par le positionnement dans la page Web du message XML. En réalité le message n'apparaît pas à ce niveau, il s'agit uniquement de mettre en place une division HTML (incluant des balises `span` identifiées).

En général la balise `<div>` est employée pour contenir et imbriquer d'autres éléments HTML (`input`, `span`…) alors que la balise `<span>` s'emploie plutôt pour encadrer et identifier des mots ou des groupes de mots.

Le code prévu pour l'affichage du contenu du mail est le suivant :

```
<!-- Affichage d'un message de type mail codé en XML -->
<h3>Affichage d'un message de type mail codé en XML</h3>
<div>
    <b>Emetteur : </b> <span id="de"></span><br />
    <b>Destinataire : </b> <span id="a"></span><br />
    <b>Sujet : </b> <span id="sujet"></span><br />
    <b>Message : </b> <span id="message"></span>
</div>
```

Chaque élément `span` est identifié, ce qui permettra ensuite le renseignement du message par les données effectives lues dans le fichier XML `email.xml`.

Le script JavaScript placé ensuite dans cette même section HTML `<body>` est ici listé intégralement (Il sera ensuite commenté en détail) :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
<!-- Début script JavaScript -->
<script type="text/Javascript">

    /*
    Création d'un objet XMLHttpRequest pour échanger des données
    avec le serveur au format texte, XML ou JSON

    NB : Les fichiers XML sont automatiquement parsés par l'objet
    et accessibles par les méthodes du DOM
    */
    if (window.XMLHttpRequest)
    {
        // Code spécifique pour les navigateurs
        // IE7+, Firefox, Chrome, Opera, Safari
        var xmlhttp = new XMLHttpRequest();
    }
    else
    {
        // Code spécifique pour les navigateurs IE6, IE5
        var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }

    /* Ouverture du fichier XML email.xml en mode synchrone (false) */
    xmlhttp.open("GET", "email.xml", false);

    /* Emission de la requête HTTP vers le serveur */
    xmlhttp.send();

    /* Création du document XML en mémoire */
    var xmlDoc = xmlhttp.responseXML;

    /* Affichage du champ Emetteur */
    document.getElementById("de").innerHTML =
    xmlDoc.getElementsByTagName("de")[0].childNodes[0].nodeValue;

    /* Affichage du champ Destinataire */
    document.getElementById("a").innerHTML =
    xmlDoc.getElementsByTagName("a")[0].childNodes[0].nodeValue;

    /* Affichage du champ Sujet */
    document.getElementById("sujet").innerHTML =
    xmlDoc.getElementsByTagName("entete")[0].childNodes[0].nodeValue;

    /* Affichage du champ Message */
    document.getElementById("message").innerHTML =
    xmlDoc.getElementsByTagName("message")[0].childNodes[0].nodeValue;

</script>
```

Le script débute par l'instanciation d'un objet de la classe `XMLHttpRequest`. Cet objet nommé `xmlhttp` dans notre cas sert à communiquer avec le serveur distant sur lequel un fichier XML de nom `email.xml` est positionné. Vous noterez que pour les versions anciennes du navigateur Microsoft Internet Explorer (versions 6 et antérieures), il faut avoir recours à un objet ActiveX en lieu et place d'un objet `XMLHttpRequest`.

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Ensuite l'ouverture du fichier distant se fait via la méthode `open` de l'objet `xmlhttp`. La méthode `open` requiert trois indications :

- La méthode d'accès (`GET` ou `POST`) : la valeur `GET` est en général utilisée et suffisante, la valeur `POST` est réservée dans le cas d'envoi de paramètres volumineux au serveur et quant les paramètres sont postés par un formulaire.
- Le nom du fichier à lire (`email.xml` dans notre cas).
- Le comportement synchrone ou asynchrone de la requête soumise au serveur, le mode synchrone (`false`) choisi dans notre cas sert à indiquer que toute activité du navigateur est suspendue pendant l'exécution de la requête (cette valeur n'est en réalité pas la valeur conseillée).

La requête est ensuite soumise au serveur par la méthode `send`, ici utilisée sans paramètres.

La réponse renvoyée par le serveur, c'est-à-dire le fichier XML, est ensuite affectée à une variable nommée `xmlDoc` :

```
/* Création du document XML en mémoire */
var xmlDoc = xmlhttp.responseXML;
```

C'est cette variable qui va ensuite être exploitée via des méthodes DOM pour en extraire le contenu utile.

Par exemple pour la récupération du champ émetteur du mail, le code est :

```
/* Affichage du champ Emetteur */
document.getElementById("de").innerHTML =
xmlDoc.getElementsByTagName("de")[0].childNodes[0].nodeValue;
```

Par la méthode DOM `getElementsByTagName`, le nom de l'émetteur (`Olivier VIGOUROUX` dans notre cas) est extrait du flux XML et affecté à l'élément HTML nommé `de`.

La séquence `[0].childNodes[0].nodeValue` indique qu'il est fait référence au premier tag de nom `de` du document, et à la valeur de son premier enfant.

À l'affichage, nous aurons :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

2. Exemple 2 : Liste des marques des voitures (fichier voitures.xml)

Dans ce deuxième exemple, l'objectif est de lister l'ensemble des marques des voitures de sport mémorisées dans un fichier XML. Bien sûr ce traitement présente des similitudes avec le traitement vu dans le premier exemple.

L'objectif dans ce premier exemple va être d'afficher en HTML un flux de données XML récupéré sur un site Web.

Fichier voitures.xml

Le document `voitures.xml` va être exploité par le script du deuxième exemple et aussi par ceux des exemples qui vont suivre dans ce chapitre. Consultez-le attentivement :

```
<voitures>
    <voiture origine="ITALIE" groupe="FIAT">
        <marque>FERRARI</marque>
        <modele>512 BB</modele>
        <moteur>V12 FERRARI</moteur>
        <cylindree>5</cylindree>
        <puissance>380</puissance>
        <vitesse_pointe>280</vitesse_pointe>
        <site_web>
            http://fr.wikipedia.org/wiki/Ferrari_Berlinetta_Boxer
        </site_web>
    </voiture>
    <voiture origine="ITALIE" groupe="VOLKSWAGEN">
        <marque>LAMBORGHINI</marque>
        <modele>MIURA</modele>
        <moteur>V12 LAMBORGHINI</moteur>
        <cylindree>3.9</cylindree>
        <puissance>385</puissance>
        <vitesse_pointe>280</vitesse_pointe>
        <site_web>
            https://fr.wikipedia.org/wiki/Lamborghini_Miura
        </site_web>
        <site_web>
            http://en.wikipedia.org/wiki/Lamborghini_Miura
        </site_web>
    </voiture>
    <voiture origine="ALLEMAGNE" groupe="VOLKSWAGEN">
        <marque>PORSCHE</marque>
        <modele>964 TURBO</modele>
        <moteur>FLAT 6 PORSCHE</moteur>
        <cylindree>3.6</cylindree>
        <puissance>360</puissance>
        <vitesse_pointe>295</vitesse_pointe>
        <site_web>
            https://en.wikipedia.org/wiki/Porsche_964
        </site_web>
    </voiture>
</voitures>
```


© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Section HTML <body>

Dans le script HTML/JavaScript (nommé `DOM_02.htm`), vous retrouverez l'instanciation d'un objet pour l'accès distant (`XMLHttpRequest` ou ActiveX `Microsoft.XMLHTTP` selon le navigateur), l'ouverture du fichier et la soumission de la requête et enfin la création du document DOM en mémoire.

Dans ce script, il n'est pas prévu de zones d'affichage `span` dans une division HTML, les informations vont être simplement affichées dans la page Web via la méthode `document.write`.

La séquence de code débute par la création d'un tableau mémoire, nommé `marque`, dans lequel seront stockés les nœuds `marque` du fichier `voitures.xml` :

```
/* Récupération de l'ensemble des noeuds marque (tableau mémoire) */
var marque = xmlDoc.getElementsByTagName("marque");
```

Ensuite, passons aux affichages :

```
/* Affichage de la marque de la 1ère voiture */
document.write("Marque voiture n°1 : " +
marque[0].childNodes[0].nodeValue + "<br />");

/* Affichage de la marque de la 2ème voiture */
document.write("Marque voiture n°2 : " +
marque[1].childNodes[0].nodeValue + "<br />");

/* Affichage de la marque de la 3ème voiture */
document.write("Marque voiture n°3 : " +
marque[2].childNodes[0].nodeValue);
```

Évidemment cette solution d'affichage n'est pas tout à fait satisfaisante car elle dépend du nombre de voitures (trois) du fichier XML `voitures.xml`. Nous verrons dans un exemple ultérieur comment résoudre cette difficulté.

Dans notre exemple `marque[0].childNodes[0].nodeValue` désigne la première occurrence de balise `marque`, puis la valeur du premier enfant de cette balise.

À l'exécution nous aurons l'affichage suivant :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

3. Exemple 3 : Liste des marques des voitures avec une boucle

Nous avions remarqué une limite gênante dans l'exemple précédent, voyons comme traiter cette difficulté via une boucle.

Le code est somme toute assez simple :

```
/*
Boucle d'affichage des marques des voitures
*/
/* Récupération de l'ensemble des noeuds marque (tableau mémoire) */
var marque = xmlDoc.getElementsByTagName("marque");
alert("Nombre de voitures : " + marque.length);
for (i=0; i<marque.length; i++)
{
    /* Affichage de la marque de la voiture en cours */
    /* NB : La numérotation des voitures commence à zéro */
    document.write("Marque de la voiture n°" + (i+1) + " : "
    + marque[i].childNodes[0].nodeValue + "<br />");
}
```

Après la récupération des nœuds `marque` dans un tableau, il est possible de balayer ce tableau par une boucle `for`. Les éléments du tableau sont indicés classiquement de zéro jusqu'à `marque.length - 1` (`length` étant le nombre d'éléments du tableau).

À chaque tour de boucle, la marque est affichée par l'intermédiaire de `marque[i].childNodes[0].nodeValue`, `i` étant l'indice de la boucle `for`.

L'affichage obtenu est le suivant :

Il est donc strictement identique à celui de l'exemple précédent.

4. Exemple 4 : Liste des nœuds rattachés à la racine

L'objectif de cet exemple est plutôt d'ordre technique, connaître la liste des nœuds rattachés à la racine et le nom de cette racine :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Le code, une fois le document XML créé en mémoire, est le suivant :

```
/* Affichage de la liste des noeuds rattachés à la racine */
document.write("Liste des noeuds rattachés à la racine :<br />")
listeNoeudsRacine = xmlDoc.documentElement.childNodes;
for (i=0; i<listeNoeudsRacine.length; i++)
{
    /* Affichage uniquement de noeuds de type 1 */
    if (listeNoeudsRacine[i].nodeType == 1)
    {
        document.write(" - " + listeNoeudsRacine[i].nodeName + "<br />");
    }
}

/* Affichage du nom du noeud racine */
document.write("<br />");
document.write("Le noeud racine est "
+ xmlDoc.documentElement.nodeName + "<br />");

/* Affichage du type du noeud racine */
document.write("Le type du noeud racine est "
+ xmlDoc.documentElement.nodeType);
```

Un tableau mémoire, nommé `listeNoeudsRacine`, est créé par :

```
listeNoeudsRacine = xmlDoc.documentElement.childNodes;
```

Ce tableau est ensuite balayé par une boucle `for`.

Une particularité tout de même, pour afficher uniquement les nœuds directement rattachés à la racine, il faut prévoir un test sur le type des nœuds comme suit :

```
if (listeNoeudsRacine[i].nodeType == 1)
```

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

L'affichage est obtenu par l'intermédiaire de :

```
document.write(" - " + listeNoeudsRacine[i].nodeName + "<br />");
```

Après cette boucle `for`, à titre indicatif le nom et le type du nœud racine (`voitures`) sont également annoncés à l'écran.

L'indication `nodeType` peut dans un arbre DOM prendre les valeurs suivantes (source developer.mozilla.org) :

- 1 : Nœud élément,
- 2 : Nœud attribut,
- 3 : Nœud texte,
- 4 : Nœud de section CDATA,
- 5 : Nœud de référence à une entité,
- 6 : Nœud d'entité,
- 7 : Nœud d'instruction de traitement,
- 8 : Nœud de commentaire,
- 9 : Nœud de document,
- 10 : Nœud de type de document,
- 11 : Nœud de fragment de document,
- 12 : Nœud de notation.

5. Exemple 5 : Liste des champs (nœuds) de chaque voiture

Dans cet exemple, nous allons nous attacher à lister les champs (nœuds) de chaque `voiture`.

Dans notre cas nous obtiendrons cet affichage :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Nous avons aussi profité de ce parcours des champs pour afficher les deux liens Web décrivant la deuxième voiture de notre fichier `voitures.xml`.

Passons à l'étude du code JavaScript, placé dans la section HTML `<body>`.

Après la création de l'objet `XMLHttpRequest`, l'accès au fichier XML `voitures.xml` et la création du document DOM en mémoire, analysons la séquence de code spécifique à cet exemple.

La première séquence concerne la liste des champs (nœuds) de chaque `voiture` :

```
/*
Affichage de liste des champs (noeuds) de chaque voiture
NB : La liste est basée sur celle de la 1ère voiture
*/
/* Affichage du titre de la liste */
document.write("Liste des champs (noeuds) de chaque voiture :<br />")
/* Constitution de la liste des noeuds enfants
de voiture (basée sur la 1ère voiture) */
listeChampsVoiture =
xmlDoc.getElementsByTagName("voiture")[0].childNodes;
/* Initialisation de la valeur du premier enfant (marque) */
champVoiture =
xmlDoc.getElementsByTagName("voiture")[0].firstChild;
/* Parcours des enfants */
for (i=0; i<listeChampsVoiture.length; i++)
{
    /* Test du type du noeud pour n'afficher que les noeuds éléments */
    /*
    nodeType : 1  -> nœud élément
    nodeType : 2  -> nœud attribut
    nodeType : 3  -> nœud texte
    nodeType : 4  -> nœud de section CDATA
    nodeType : 5  -> nœud de référence à une entité
```

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
    nodeType : 6  -> nœud d'entité
    nodeType : 7  -> nœud d'instruction de traitement
    nodeType : 8  -> nœud de commentaire
    nodeType : 9  -> nœud de document
    nodeType : 10  -> nœud de type de document
    nodeType : 11  -> nœud de fragment de document
    nodeType : 12  -> nœud de notation
    */
    if (champVoiture.nodeType == 1)
    {
            document.write(" - " + champVoiture.nodeName + "<br />");
    }
    /* Passage au champ suivant */
    champVoiture = champVoiture.nextSibling;
}
```

Après avoir constitué la liste des nœuds dépendants (enfants) du niveau `voiture`, en se basant sur la première occurrence de `voiture`, comme suit :

```
listeChampsVoiture =
xmlDoc.getElementsByTagName("voiture")[0].childNodes;
```

un positionnement sur le premier enfant est effectué :

```
champVoiture =
xmlDoc.getElementsByTagName("voiture")[0].firstChild;
```

Il reste ensuite à parcourir les champs en utilisant la méthode `nextSibling` (passage au nœud suivant de même niveau). Vous noterez qu'il faut une fois de plus filtrer pour n'afficher qu'en cas de nœud de type `1` (nœud élément).

Une deuxième séquence est prévue ensuite pour afficher les deux liens `web` décrivant la deuxième voiture :

```
/*
Affichage du nombre de sites Web décrivant
la 2ème voiture (numérotée 1)
*/
document.write("<br />");
document.write("Liste des sites Web décrivant le 2ème voiture :<br />")
listeChampsVoiture = xmlDoc.getElementsByTagName("voiture")[1].childNodes;
champVoiture = xmlDoc.getElementsByTagName("voiture")[1].firstChild;
for (i=0; i<listeChampsVoiture.length; i++)
{
    if (champVoiture.nodeType == 1 && champVoiture.nodeName == "site_web")
    {
        document.write(" - " + champVoiture.nodeName + " : "
        + champVoiture.childNodes[0].nodeValue + "<br />");
    }
    /* Passage au champ suivant */
    champVoiture = champVoiture.nextSibling;
}
```


© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Vous constatez que la démarche est assez similaire à celle du traitement précédent. La seule différence est trouvée au niveau du test :

```
if (champVoiture.nodeType == 1 && champVoiture.nodeName == "site_web")
```

6. Exemple 6 : Remplacement d'une valeur de nœud

Bien qu'il s'agisse d'une opération bien moins fréquente, regardons comment il est possible de modifier une valeur dans le document XML en mémoire après sa lecture depuis le fichier `voitures.xml`. Le code est, une fois de plus, positionné dans la section HTML `<body>` :

```
/* Remplacement d'une valeur de noeud (marque) sur la première voiture */
marqueVoiture = xmlDoc.getElementsByTagName("marque")[0].childNodes[0];
marqueVoiture.nodeValue="FERRARI ITALIA";

/* Contrôle du remplacement effectué */
marqueVoiture = xmlDoc.getElementsByTagName("marque")[0].childNodes[0];
texteMarqueVoiture = marqueVoiture.nodeValue;
document.write("La marque de la 1ère voiture est désormais : "
+ texteMarqueVoiture);
```

Rien de bien difficile en réalité, après avoir repéré la première occurrence de balise `voiture`, il suffit d'intervenir sur la valeur du nœud (`nodeValue`). Un contrôle est ensuite effectué pour vérifier l'impact de la modification.

À l'affichage, nous obtenons :

7. Exemple 7 : Accès aux attributs

Revenons sur le contenu du fichier XML `voitures.xml` :

```
<voitures>
    <voiture origine="ITALIE" groupe="FIAT">
        ...
    </voiture>
    <voiture origine="ITALIE" groupe="VOLKSWAGEN">
```

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
        ...
    </voiture>
    <voiture origine="ALLEMAGNE" groupe="VOLKSWAGEN">
        ...
    </voiture>
</voitures>
```

Voyons comment accéder aux attributs `origine` et `groupe` des différentes voitures :

```
/* Accès aux attributs de la 1ère voiture (numérotée 0) */
premiereVoiture = xmlDoc.getElementsByTagName("voiture")[0].attributes;

/* Affichage de la valeur de l'attribut origine
de la 1ère voiture (numérotée 0) */
document.write("Origine de la 1ère voiture : "
+ premiereVoiture.getNamedItem("origine").nodeValue + "<br />");

/* Accès aux attributs de la 3ème voiture (numérotée 2) */
troisiemeVoiture = xmlDoc.getElementsByTagName("voiture")[2].attributes;

/* Affichage de la valeur de l'attribut origine
de la 3ème voiture (numérotée 2) */
document.write("Origine de la 3ème voiture : "
+ troisiemeVoiture.getNamedItem("origine").nodeValue + "<br />");

/* Affichage de la valeur de l'attribut groupe
de la 3ème voiture (numérotée 2) */
document.write("Groupe de la 3ème voiture : "
+ troisiemeVoiture.getNamedItem("groupe").nodeValue);
```

Le code largement commenté ne requiert pas d'explications supplémentaires. Signalons juste que la propriété `attributes` renvoie une collection des attributs spécifiés sur l'élément donné.

L'affichage obtenu à l'exécution est :

8. Exemple 8 : Accès à un nœud parent

Il peut être intéressant lors d'une navigation dans un arbre XML de pouvoir accéder au nœud parent du nœud en

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

cours. Voyons comment réaliser cette opération au travers d'une courte séquence de code :

```
/* Boucle d'affichage de la marque des voitures */
marquesVoitures = xmlDoc.getElementsByTagName("marque");
alert("Nombre de voitures : " + marquesVoitures.length);
for (i=0; i<marquesVoitures.length; i++)
{
    /* Affichage de la marque de la voiture en cours
    et de son noeud parent */
    document.write("Marque de la voiture n° "
    + (i+1)
    + " : "
    + marquesVoitures[i].childNodes[0].nodeValue
    + " (a pour parent "
    + marquesVoitures[i].parentNode.nodeName
    + ")<br />");
}
```

Dans notre exemple, nous parcourons le tableau mémoire `marquesVoitures` contenant tous les nœuds `marque` par l'intermédiaire d'une boucle `for`.

Par l'instruction `marquesVoitures[i].parentNode.nodeName`, le nœud parent est annoncé.

C'est sans surprise que nous découvrons que chaque nœud `marque` a pour parent le nœud `voiture` :

9. Exemple 9 : Parcours arrière des nœuds

Étudions également au travers d'un petit script comment parcourir en marche arrière une liste de nœuds (du dernier au premier) :

```
/* Affichage des noeuds enfants (en marche arrière) du noeud voiture
pour la 3ème voiture (numérotée 2) */
listeChampsVoiture = xmlDoc.getElementsByTagName("voiture")[2].childNodes;
champVoiture = xmlDoc.getElementsByTagName("voiture")[2].lastChild;
for (i=0; i<listeChampsVoiture.length; i++)
{
    if (champVoiture.nodeType == 1)
```

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
    {
        document.write(" - "
        + champVoiture.nodeName
        + " : "
        + champVoiture.childNodes[0].nodeValue
        + "<br />");
    }
    /* Passage au champ précédent */
    champVoiture = champVoiture.previousSibling;
}
```

La liste des nœuds enfants de la troisième voiture (numérotée 2) est d'abord établie :

```
listeChampsVoiture =
xmlDoc.getElementsByTagName("voiture")[2].childNodes;
```

Ensuite le positionnement sur le dernier enfant se fait comme suit :

```
champVoiture = xmlDoc.getElementsByTagName("voiture")[2].lastChild;
```

Nous retrouvons ensuite une structure itérative avec cette fois-ci l'utilisation de la méthode `previousSibling` :

```
champVoiture = champVoiture.previousSibling;
```

Le résultat à l'exécution est le suivant :

10. Exemple 10 : Remplacement systématique d'une valeur d'attribut

Rappelez-vous les valeurs d'attributs pour nos trois voitures :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
<voitures>
    <voiture origine="ITALIE" groupe="FIAT">
        ...
    </voiture>
    <voiture origine="ITALIE" groupe="VOLKSWAGEN">
        ...
    </voiture>
    <voiture origine="ALLEMAGNE" groupe="VOLKSWAGEN">
        ...
    </voiture>
</voitures>
```

Il est possible de faire un remplacement systématique sur ces valeurs, comme ceci :

```
/* Remplacement de la valeur de l'attribut origine
pour toutes les voitures */
voiture = xmlDoc.getElementsByTagName("voiture");
for (i=0; i<voiture.length; i++)
{
    /* Remplacement */
    voiture[i].setAttribute("origine", "EUROPE");
    /* Contrôle du remplacement */
    document.write("Pays du constructeur de la voiture n°"
    + (i+1)
    + " : "
    + voiture[i].getAttribute("origine")
    + "<br />");
}
```

La méthode `setAttribue` appliquée à chaque élément du tableau `voiture` permet de modifier la valeur de cet attribut. Rappelons que `voiture` est un tableau regroupant tous les nœuds `voiture` du fichier `voitures.xml`.

Un contrôle, après modification, est aussi réalisé par la méthode `getAttribute`.

L'exécution du script donne l'affichage ci-après :

11. Exemple 11 : Conversion XML en HTML

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

Pour rendre l'affichage du flux XML plus aisé, il peut être pertinent de réaliser une conversion en code HTML.

Construisons une table HTML à deux colonnes à partir des nœuds `marque` et `modele` de notre flux XML, comme ceci :

```
/*
Création du tableau HTML
*/
/* Balise HTML de début de table */
document.write("<table border='1'>");
/* Mise en place de la ligne des intitulé de colonnes */
document.write("<tr>");
document.write("<td>");
document.write("Marque");
document.write("</td>");
document.write("<td>");
document.write("Modèle");
document.write("</td>");
document.write("</tr>");
/* Parcours de la liste des voitures */
var voiture = xmlDoc.getElementsByTagName("voiture");
for (i=0; i<voiture.length; i++)
{
    /* Mise en place d'un nouvelle ligne dans le table HTML */
    document.write("<tr>");
    /* Ouverture de la 1ère colonne (marque de la voiture) */
    document.write("<td>");
    /* Ecriture de la marque de la voiture */
    document.write(voiture[i]
    .getElementsByTagName("marque")[0]
    .childNodes[0]
    .nodeValue);
    /* Fermeture de la 1ère colonne */
    document.write("</td>");
    /* Ouverture de la 2ème colonne (modèle de la voiture) */
    document.write("<td>");
    /* Ecriture du modèle de la voiture */
    document.write(voiture[i]
    .getElementsByTagName("modele")[0]
    .childNodes[0]
    .nodeValue);
    /* Fermeture de la 2ème colonne */
    document.write("</td>");
    /* Fermeture de la ligne dans le table HTML */
    document.write("</tr>");
}
/* Balise HTML de fin de table */
document.write("</table>");
```

Le script, un peu long, ne présente aucune difficulté. Un tableau `voiture` est construit à partir de la liste des nœuds `voiture` :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
var voiture = xmlDoc.getElementsByTagName("voiture");
```

puis est parcouru par une boucle `for`.

La récupération des deux valeurs de nœud (champ) pour chaque `voiture` est obtenue comme suit :

```
document.write(voiture[i]
.getElementsByTagName("marque")[0]
.childNodes[0]
.nodeValue);
```

```
document.write(voiture[i]
.getElementsByTagName("modele")[0]
.childNodes[0]
.nodeValue);
```

Le reste du code est constitué par de la mise en forme HTML de table.

Le tableau HTML aura cette allure :

12. Exemple 12 : Suppression d'un nœud dans un flux XML

Il nous reste à voir comment supprimer un nœud d'un document DOM en mémoire.

La méthode `removeChild` est facile à mettre en œuvre.

Nous allons par contre devoir développer un exemple un peu plus conséquent pour réaliser une suppression de nœud.

Section HTML <body>

Dans cette partie il sera fait appel consécutivement à trois fonctions JavaScript que nous étudierons plus loin :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
/* Chargement des voitures depuis le fichier XML dans un arbre */
chargerVoitures();

/* Affichage de la liste des voitures */
afficherVoitures();

/* Effacement de la première voiture */
var confirmationEffacementVoiture = confirm("Effacement de la première
voiture ?");
if (confirmationEffacementVoiture == true)
{
    effacerVoiture();
}
```

La séquence est somme toute logique, d'abord la lecture du fichier XML `voitures.xml` pour constituer notre arbre, ensuite l'affichage avant la suppression d'un nœud pilotée par un `confirm` (il s'agit de la première utilisation de la méthode `confirm` dans ce livre). Une fois la suppression effectuée, le réaffichage de l'arbre est prévu, nous le verrons plus loin.

Section HTML <head>

La première fonction, `chargerVoitures`, ne présente aucun élément nouveau :

```
/* Fonction chargerVoitures */
function chargerVoitures()
{
    /*
    Création d'un objet XMLHttpRequest pour échanger des données
    avec le serveur au format texte, XML ou JSON

    NB : Les fichiers XML sont automatiquement parsés
    par l'objet et accessibles par les méthodes du DOM
    */
    if (window.XMLHttpRequest)
    {
        // Code spécifique pour les navigateurs
        // IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp = new XMLHttpRequest();
    }
    else
    {
        // Code spécifique pour les navigateurs IE6, IE5
        xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    }

    /* Ouverture du fichier XML voitures.xml en mode synchrone (false)*/
    xmlhttp.open("GET", "voitures.xml", false);

    /* Emission de la requête HTTP vers le serveur  */
    xmlhttp.send();
```


© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare

```
    /* Création du document XML en mémoire */
    xmlDoc = xmlhttp.responseXML;

    /* Mise en place du titre du traitement */
    document.write("<h1>Editions ENI - JavaScript - DOM_12</h1>");
}
```

La fonction `afficherVoitures` reprend le code vu dans l'exemple précédent. Il n'est pas reproduit ici.

La troisième fonction, `effacerVoiture`, est plus intéressante :

```
/* Fonction effacerVoiture */
function effacerVoiture()
{
    /* Affichage du nombre de voiture(s) avant suppression */
    alert("Nombre de voiture(s) avant suppression : "
    + xmlDoc.getElementsByTagName('voiture').length);
    /* Effacement de la première voiture */
    voitureSupprimee = xmlDoc.getElementsByTagName("voiture")[0];
    xmlDoc.documentElement.removeChild(voitureSupprimee);
    /* Affichage du nombre de voiture(s) après suppression */
    alert("Nombre de voiture(s) après suppression : "
    + xmlDoc.getElementsByTagName('voiture').length);
    document.write("<br /><br />Liste des voitures après suppression");
    afficherVoitures();
    /* Valeur de retour */
    return true;
}
```

L'effacement de la première voiture est assuré par la séquence :

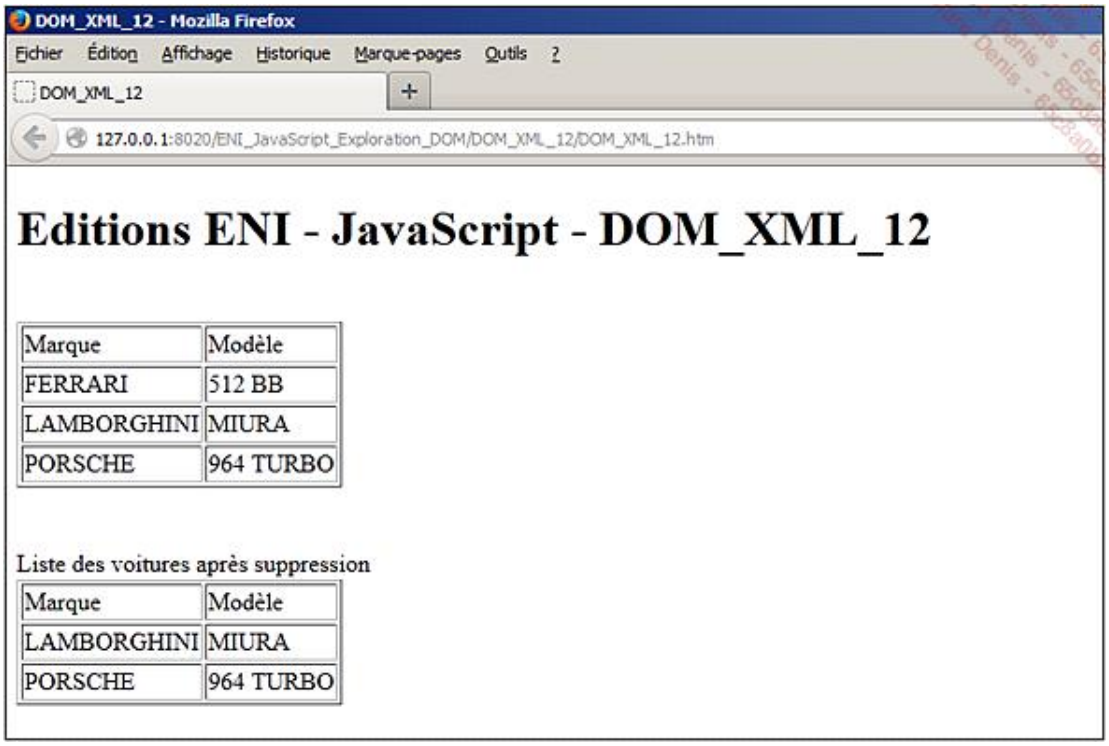
```
/* Effacement de la première voiture */
voitureSupprimee = xmlDoc.getElementsByTagName("voiture")[0];
xmlDoc.documentElement.removeChild(voitureSupprimee);
```

Cette fonction appelle aussi la fonction `afficherVoitures`, une fois la suppression de la première voiture réalisée, pour contrôle :

```
afficherVoitures();
```

Nous obtenons cet affichage à l'exécution :

© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare



© Editions ENI – Tous droits réservés – Copie personnelle de Denis Gaudare