

L'API DOM et son implémentation en JavaScript

Ingénierie Documentaire
<http://doc.crzt.fr>



Stéphane Crozat

14 septembre 2016

Table des matières



I - Introduction au DOM	3
1. Principes du DOM	3
2. L'interface DOM	4
II - Implémentation JavaScript du DOM	7
1. Introduction à JavaScript	7
2. Fonctions et objets DOM en JavaScript	8
3. Spécificités HTML	9
III - Fonctions et objets JavaScript liés à XML	10
1. Utiliser XPath	10
2. Exécuter une XSLT	10
3. XMLHttpRequest : Communication avec le serveur	11
4. XMLSerializer : Sérialiser un DOM	12
IV - Exercices	13
1. Travaux pratiques DOM et JavaScript	14
1.1. Exercice : "Hello World !" en JavaScript	14
1.2. Exercice : Manipulation du DOM en JavaScript	14
1.3. Exercice : Un éditeur XML en JavaScript	16
Solutions des exercices	19
Bibliographie	24

Introduction au DOM



1. Principes du DOM

Définition

Le DOM (*Document Object Model*) est un standard W3C qui décrit une API orientée objet permettant de représenter un document XML ou HTML en mémoire afin de la manipuler avec un langage de programmation.

Le standard DOM est indépendant d'un langage de programmation en particulier, et implémenté dans de nombreux langages (JavaScript, Java, PHP, C, ...).

« *The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.* »

<http://www.w3.org/DOM/>

Fonctions du DOM

L'API DOM permet de :

- Parcourir un arbre XML ou HTML
- Rechercher des nœuds particuliers, en consulter le contenu
- Créer, ajouter, supprimer, déplacer des nœuds dans l'arbre

Exemple

L'API DOM implémentée en JavaScript permet de manipuler un document HTML ou XHTML, afin d'en modifier dynamiquement (en fonction des actions utilisateurs), la structure et le contenu :

- Faire apparaître ou disparaître des éléments `div` en fonction d'une action utilisateur
- Créer ou initialiser des contrôles de formulaire
- ...

```
1 function fDisplay(pDivId) {  
2   var vDiv = document.getElementById(pDivId);  
3   vDiv.style.display="block";  
4 }
```

Rappel

L'arbre du document XML (cf. p.)

2. L'interface DOM

Fondamental

-
- Document Object Model (DOM) Technical Reports
<http://www.w3.org/DOM/DOMTR>
 - Document Object Model Core
<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html>

Objets principaux

- Node : Élément central du DOM, il représente tout nœud de l'arbre DOM, les autres interfaces en héritent.
- NodeList : Ensemble ordonné de Nodes.
- Document : Racine d'un fichier XML bien formé (ou d'un fichier HTML valide), son fils est donc l'élément racine du document XML.
- Element : Représentation d'un élément au sens XML
- Attr : Représentation d'un attribut au sens XML
- Text : Contenu textuel d'un nœud texte ou d'un attribut au sens XML

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1590626201>

Attributs principaux de Node

- `string nodeName`
- `integer nodeType`
- `Node parentNode`
- `NodeList childNodes`
- `Node firstChild`

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1950641247>

Attributs principaux de NodeList

- `Node item(integer index)`
- `integer length`

Fonctions principales de Node

- `Node insertBefore(Node newChild, Node refChild)`
- `Node replaceChild(Node newChild, Node oldChild)`
- `Node removeChild(Node oldChild)`
- `Node appendChild(Node newChild)`
- `boolean hasChildNodes()`
- `Node cloneNode(boolean deep)`

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1950641247>

Fonctions principales de Document

- Element getElementById(string elementId)
- NodeList getElementsByTagName(string tagName)
- Element createElement(string tagName)
- Text createTextNode(string data)
- Attr createAttribute(string name)

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#i-Document>

Fonctions principales de Element

- NodeList getElementsByTagName(string name)
- Attr getAttributeNode(string name)
- Attr setAttributeNode(Attr newAttr)
- Attr removeAttributeNode(Attr oldAttr)
- boolean hasAttribute(string name)

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-745549614>

Text (attributs et fonctions)

Attributs :

- string data
- integer length

Fonctions :

- void appendData(string arg)
- void insertData(integer offset, string arg)
- void deleteData(integer offset, integer count)
- void replaceData(integer offset, integer count, string arg)

<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-1312295772>

Attr (attributs)

- string name
- string value


<http://www.w3.org/TR/2004/REC-DOM-Level-3-Core-20040407/core.html#ID-637646024>



Remarque : Namespace

Il existe toutes les fonctions nécessaires pour gérer les *namespaces*, comme par exemple :

- NodeList getElementsByTagNameNS(string namespaceURI, string localName)
- Element createElementNS(string namespaceURI, string qualifiedName)
- Attr createAttributeNS(string namespaceURI, string qualifiedName)

 *Complément*

(Brillant07 *, pp238-245)

Implémentation JavaScript du DOM

II

1. Introduction à JavaScript

Définition : JavaScript

JavaScript est un langage de programmation instance du standard ECMAScript, principalement utilisé comme langage client dans les navigateurs Web, afin de rendre dynamique l'affichage et la manipulation de pages HTML.

Rappel : Intégration JavaScript HTML

```
1 //file.js
2 function func(pArg1, pArg2) {
3 ...
4 }
```

```
1 <html>
2 <head>
3 ...
4 <script src="file.js" type="text/javascript"></script>
5 ...
6 </head>
7 <body>
8 ...
9 <div ... onClick="func(vArg1, vArg2)">
10 ...
11 </body>
12 </html>
```

Rappel : Introduction au JavaScript

<http://fr.selfhtml.org/javascript/intro.htm>

Rappel : Syntaxe JavaScript

<http://fr.selfhtml.org/javascript/langage>

Complément : Fonctions JavaScript

<http://fr.selfhtml.org/javascript/objets/independant.htm>

Complément : Objets JavaScript

<http://fr.selfhtml.org/javascript/objets/>

Rappel : Console JavaScript

Pour afficher la console JavaScript sous Firefox : CTRL+MAJ+J

2. Fonctions et objets DOM en JavaScript

Exemple

```
1 vDiv = document.getElementById("id1");
```

Exemple

```
1 vDivs = document.getElementsByTagName("p");  
2 for (var i=0; i<vDivs.length; i++) {  
3   vDivs.item(i)...;  
4 }
```

Exemple

```
1 alert(vDiv.firstChild.data) ;
```

Exemple

```
1 vDiv.removeChild(vDiv.firstChild);
```

Exemple

```
1 vDiv.appendChild(vDiv2);
```

Syntaxe

- Document
<http://fr.selfhtml.org/javascript/objets/document.htm>
- Node
<http://fr.selfhtml.org/javascript/objets/node.htm>
- getElementById
http://fr.selfhtml.org/javascript/objets/document.htm#get_element_by_id
- getElementsByTagName
http://fr.selfhtml.org/javascript/objets/document.htm#get_elements_by_tag_name
- firstChild
http://fr.selfhtml.org/javascript/objets/node.htm#first_child
- data
<http://fr.selfhtml.org/javascript/objets/node.htm#data>

3. Spécificités HTML

Attributs et méthodes spécifiques aux éléments HTML

- input, select : value, name, ...
- a : href
- ...


<http://fr.selfhtml.org/javascript/objets/elementshtml.htm>

querySelectorAll : Manipulation HTML via les classes CSS

querySelectorAll est une méthode de l'objet document permettant de renvoyer une liste de nœuds correspondant à l'union des sélecteurs CSS passés en paramètre.

```
NodeList document.querySelectorAll(<list_of_selectors>)
```

<https://developer.mozilla.org/En/DOM/Document.querySelector>

 *Exemple : querySelectorAll*

```
1 ...
2 <body>
3 <div class="class1">...</div>
4 <div class="class2">...</div>
5 <div class="class1 class2">...</div>
6 <div class="class2 class3">...</div>
7 ...
```

document.querySelectorAll("div.class1, div.class2") renvoie tous les div de l'extrait HTML ci-avant.

 *Conseil : Firebug*

Installer l'extension Firebug sous Firefox pour pouvoir visualiser dynamiquement le DOM.

<https://addons.mozilla.org/fr/firefox/addon/firebug/>

Fonctions et objets JavaScript liés à XML



1. Utiliser XPath

L'API XPath est moins pratique que l'API DOM en JavaScript, et en conséquence moins utilisée. Il est néanmoins possible d'exécuter un XPath en fonction d'un nœud courant, les deux étant passés en paramètres à la méthode `evaluate()` de `document`.

Exemple

```
1 var nodesSnapshot = document.evaluate(<pXPath>, <pContextNode>, null, XPathResult.  
  ORDERED_NODE_SNAPSHOT_TYPE, null );  
2 for ( var i=0; i < nodesSnapshot.snapshotLength; i++ )  
3 {  
4 ...nodesSnapshot.snapshotItem(i)...  
5 }
```

Complément

https://developer.mozilla.org/fr/Introduction_%C3%A0_l%27utilisation_de_XPath_avec_JavaScript

2. Exécuter une XSLT

Syntaxe

Objet : `XSLTProcessor`

Méthodes :

- `importStylesheet(vXsl)`
- `DocumentFragment transformToFragment(vXml, document)`
- `Document transformToDocument(vXml)`

Syntaxe

```
1 // Chargement d'une XSLT depuis le serveur  
2 var vXhr = new XMLHttpRequest();  
3 vXhr.open("GET", "fichier.xslt", false);  
4 vXhr.send(null);  
5 var vXsl = vXhr.responseXML;
```


```

6
7 // Initialisation d'un processeur XSLT
8 var vProc = new XSLTProcessor();
9 vProc.importStylesheet(vXsl);
10
11 // Sélection d'un nœud XML source de la transformation dans le Document
12 var vXml = document.getElementById("source");
13
14 // Exécution de la XSLT dans un DocumentFragment vResult
15 var vResult = vProc.transformToFragment(vXml,document);
16
17 // Ajout de vResult dans le Document
18 document.getElementById("source").appendChild(vResult);

```

3. xmlHttpRequest : Communication avec le serveur

L'objet XMLHttpRequest permet d'envoyer des commandes POST et GET à un serveur HTTP, en particulier (mais pas uniquement) pour transmettre des documents XML entre le client et le serveur.

 *Exemple : Exécuter un fichier PHP avec des paramètres*

```


1 var vXhr = new XMLHttpRequest();
2 vXhr.open("POST", "fichier.php", false);
3 vXhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
4 var vArgs="key1=" + encodeURIComponent(vValue1) + "&key2=" + encodeURIComponent(vValue2);
5 vXhr.send(vArgs);

```

```

1 <?php
2 $vParam1=$_POST['key1'];
3 $pParam2=$_POST['key2'];
4 ...
5 ?>

```

 *Exemple : Charger un document XML sur le client*

```

1 var vXhr = new XMLHttpRequest();
2 vXhr.open("GET", "fichier.xml", false);
3 vXhr.send(null);
4 vXml =vXhr.responseXML;

```

 *Exemple : Enregistrer un document XML sur le serveur (via PHP)*

```

1 var vXml = document.getElementById... //un node DOM
2 var vTxt = (new XMLSerializer()).serializeToString(vXml);
3 var vArgs = "xml=" + encodeURIComponent(vTxt);
4 var vXhr = new XMLHttpRequest();
5 vXhr.open("POST", "saveXml.php", false);
6 vXhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
7 vXhr.send(vArgs);

```

```

1 <?php
2 $xml= $_POST["xml"];
3 $dom = new DomDocument();

```

```
4 $dom->loadXML($xml);  
5 $dom->save('src.xml');  
6 ?>
```

Syntaxe

```
1 eval(vXhr.responseText);
```

Permet au client d'ex cuter une instruction JavaScript qui aura  t  g n r e par le serveur PHP (par exemple un compte-rendu d'ex cution).

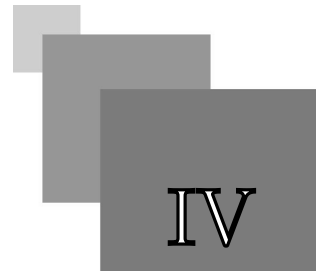
Compl ment

<http://www.xul.fr/Objet-XMLHttpRequest.html>

4. XMLSerializer : S rialiser un DOM

```
1 var xmlString = new XMLSerializer().serializeToString(Node);
```

Exercices



1. Travaux pratiques DOM et JavaScript

1.1. Exercice : "Hello World !" en JavaScript

Question 1

[solution n°1 p.19]

Créer un fichier HTML contenant un bouton. Ajouter la fonction JavaScript qui permet d'afficher "Hello World !" lorsque l'on clique sur le bouton.

Question 2

[solution n°2 p.19]

Ajouter un champ de saisie texte au fichier HTML. Ajouter la fonction JavaScript qui permet d'afficher le contenu de ce champ de saisie lorsque l'on clique sur le bouton.

1.2. Exercice : Manipulation du DOM en JavaScript

Soit le fichier HTML :

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4 <title>Mon titre</title>
5 <script type="text/javascript" src="lib.js"></script>
6 </head>
7 <body>
8 <h1 onClick="showHxContent(1)">Partie 1</h1>
9 <div id="title1" style="display:none">
10 <p>Mon premier paragraphe</p>
11 <p>Mon second paragraphe</p>
12 </div>
13 <h1 onClick="showHxContent(2)">Partie 2</h1>
14 <div id="title2" style="display:none">
15 <p>Mon premier paragraphe</p>
16 <p>Mon second paragraphe</p>
17 </div>
18 <p>
19 <input type="submit" value="Masquer" onClick="hideAllDivs()" />
20 </p>
21 <p>
22 Titre n°: <input type="text" id="title" />
23 <input type="submit" value="Afficher" onClick="alertTitle()" />
24 </p>
25 </body>
26 </html>
```

Question 1

[solution n°3 p.20]

Écrire la fonction JavaScript showHxContent permettant d'afficher le contenu du div correspondant au titre sur lequel on clique.

Indice :

```
1 function showHxContent (pNumber) {
2   vDiv = document.getElementById(...);
3   vDiv.style.display = "...";
4 }
```

Question 2*[solution n°4 p.20]*

Écrire la fonction JavaScript `hideAllDivs` permettant de masquer le contenu de tous les `div` du document.

Indice :

```
1 function hideAllDivs () {
2   vDivs = document.getElementsByTagName(...);
3   for (var i=0; i<vDivs.length; i++) {
4     vDivs.item(i).style.display = 'none';
5   }
6 }
```

Question 3*[solution n°5 p.20]*

Écrire la fonction `alertTitle` permettant d'afficher (avec la fonction JavaScript `alert`) le contenu du *i*ème titre, défini par le champ de saisie `title`.

Indice :

```
1 function alertTitle () {
2   vHx = document.getElementsByTagName('h1');
3   vIndice = document.getElementById('title').value;
4   vIndice = vIndice - 1;
5   alert(vHx.item(vIndice).innerHTML);
6 }
```

Question 4*[solution n°6 p.20]*

Écrire la fonction `deleteTitle` permettant d'effacer le contenu du titre défini par le champ `title`. (suppression du nœud fils de type texte)

Indice :

```
1 function deleteTitle () {
2   ...
3   vHx.item(vIndice).removeChild(vHx.item(vIndice).firstChild);
4 }
```

Question 5*[solution n°7 p.20]*

Écrire la fonction `defineTitle` permettant d'affecter le contenu du titre défini par le champ `title` avec la valeur "Nouveau titre". On testera que le titre a ou non déjà une valeur (nœud texte) avant l'ajout, afin de la supprimer si besoin.

Indice :

```
1 function defineTitle () {
2   ...
3   vText = document.createTextNode("Nouveau titre");
4   if (vHx.item(vIndice).firstChild) {
5     vHx.item(vIndice).removeChild(vHx.item(vIndice).firstChild);
6   }
7   vHx.item(vIndice).appendChild(vText);
8 }
```

1.3. Exercice : Un éditeur XML en JavaScript

L'objectif de ce TP est de réaliser un éditeur XML avec HTML et JavaScript.

Soit le fichier XHTML "editor.html" ci-après.

```

1 <html>
2 <!--editor.html-->
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
5 <title>Éditeur de document</title>
6 <script type="text/javascript" src="lib.js"></script>
7 </head>
8 <body id="body">
9 <h1>Édition</h1>
10 <form>
11 <p>Ajouter un paragraphe</p>
12 <input type="text" size="50" id="text" />
13 <input type="button" onclick="addPara(document.getElementById('text').value);" value="ajouter" />
14 <br />
15 <input type="button" onclick="alertXml(document);" value="Afficher DOM" />
16 <br />
17 <input type="button" onclick="saveXml();" value="Sérialiser XML" />
18 </form>
19 <hr>
20 <div id="visu">
21 <h1>Visualisation</h1>
22 <script type="text/javascript">
23 //ouvre le fichier XML sur le serveur
24 var xml = loadXml("src.xml");
25 //ouvre le fichier XSLT sur le serveur
26 var xsl = loadXml("xml2html.xsl");
27 //exécute le XSLT sur le XML
28 res=execXslt(xml,xsl);
29 //afficher le résultat
30 document.getElementById("visu").appendChild(res);
31 </script>
32 </div>
33 </body>
34 </html>

```

Question 1

[solution n°8 p.21]

Expliquer le fonctionnement de "editor.html".

Question 2

Créer un fichier XML avec un élément racine document contenant des éléments paragraphe. Le placer sur un serveur Web et le rendre accessible en lecture/écriture.

Indice :

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <document>
3 <paragraphe>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pellentesque sit amet libero ac
   mauris egestas venenatis nec vitae sapien. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices
   posuere cubilia Curae; Ut commodo, neque at auctortincidunt, turpis quam molestie augue, sit amet mattis nibh
   neque eget felis.</paragraphe>
4 <paragraphe>Donec a lectus sed augue pellentesque accumsan eu ac justo. Etiam est urna, sagittis ac
   cursus nec, placerat quis velit.</paragraphe>
5 </document>

```

Question 3

[solution n°9 p.21]

Écrire les XSLT `xml2html.xsl` et `html2xml.xsl`, puis les déposer sur le serveur à côté du fichier XML.

Question 4

[solution n°10 p.22]

Écrire les fonctions `loadXml` et `alertXml` (la seconde permettant de tester la première).

Indice :

- Pour `loadXml` utiliser `XMLHttpRequest` : *xmlHttpRequest : Communication avec le serveur* (cf. p.11)
- Pour `alertXml` utiliser `XMLSerializer` : *XMLSerializer : Sérialiser un DOM* (cf. p.12)

Question 5

[solution n°11 p.22]

Écrire la fonction `execXslt`.

Indice :

Exécuter une XSLT (cf. p.10)

Question 6

[solution n°12 p.22]

Écrire la fonction `addPara`.

Question 7

[solution n°13 p.22]

Écrire la fonction `saveXml` permettant d'exécuter le fichier PHP ci-après.

```

1 <?php
2 //saveXml.php
3 $xml= $_POST["xml"];
4 $dom = new DomDocument();
5 $dom->loadXML($xml);
6 $dom->save('src.xml');
7 ?>

```


Solutions des exercices

> Solution n°1

Exercice p. 14

hw.html

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4 <title>Hello World !</title>
5 <script type="text/javascript" src="hw.js"></script>
6 </head>
7 <body>
8 <input type="button" onClick="hw()" value="Clic me !"/>
9 </body>
10 </html>
```

hw.js

```
1 function hw () {
2 alert("Hello World")
3 }
```

> Solution n°2

Exercice p. 14

hw.html

```
1 <html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
4 <title>Hello World !</title>
5 <script type="text/javascript" src="hw.js"></script>
6 </head>
7 <body>
8 <input type="text" id="text"/>
9 <input type="button" onClick="hw2()" value="Clic me !"/>
10 </body>
11 </html>
```

hw.js

```

1 function hw2() {
2 alert (document.getElementById('text').value)
3 }

```

> **Solution n°3**

Exercice p. 14

```

1 function showHxContent (pNumber) {
2 vDiv = document.getElementById("title" + pNumber);
3 vDiv.style.display = "block";
4 }

```

> **Solution n°4**

Exercice p. 15

```

1 function hideAllDivs () {
2 vDivs = document.getElementsByTagName("div");
3 for (var i=0; i<vDivs.length; i++) {
4 vDivs.item(i).style.display="none";
5 }
6 }

```

> **Solution n°5**

Exercice p. 15

```

1 function alertTitle () {
2 vHx = document.getElementsByTagName("h1");
3 vIndice = document.getElementById("title").value;
4 vIndice = vIndice - 1;
5 alert(vHx.item(vIndice).firstChild.data);
6 }

```

> **Solution n°6**

Exercice p. 15

```

1 function deleteTitle () {
2 vHx = document.getElementsByTagName("h1");
3 vIndice = document.getElementById("title").value;
4 vIndice = vIndice - 1;
5 vHx.item(vIndice).removeChild(vHx.item(vIndice).firstChild);
6 }

```

> **Solution n°7**

Exercice p. 15

```

1 function defineTitle () {
2 vHx = document.getElementsByTagName("h1");

```

```

3 vIndex = document.getElementById("titre").value;
4 vIndex = vIndex - 1;
5 vText = document.createTextNode("Nouveau titre");
6 if (vHx.item(vIndex).hasChildNodes()) {
7   vHx.item(vIndex).removeChild(vHx.item(vIndex).firstChild);
8 }
9 vHx.item(vIndex).appendChild(vText);
10 }

```

> **Solution n°8**

Exercice p. 16

1. Après avoir affiché les éléments statiques, le script charge un fichier XML situé sur le serveur (`src.xml`).
2. Il lui applique une XSLT située sur le serveur pour le transformer en HTML (`xml2html.xml`).
3. Il affiche le résultat dans un `div` de la page (`<div id="visu">`).
4. Le formulaire propose de :
 1. modifier le DOM du HTML en JavaScript (`function addPara()`);
 2. afficher le code HTML correspondant au DOM courant (`function alertXml(document)`);
 3. sérialiser le DOM sur le serveur, à la place du fichier XML source (`function saveXml()`):
 1. en appliquant une XSLT inverse de la première (`html2xml.xml`),
 2. puis en postant le flux XML obtenu à une page PHP (`saveXml.php`),
 3. qui écrase le fichier `src.xml` avec la nouvelle sérialisation du DOM.

> **Solution n°9**

Exercice p. 17

html2xml.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="xml"/>
4 <xsl:template match="div">
5   <document>
6     <xsl:apply-templates select="p"/>
7   </document>
8 </xsl:template>
9 <xsl:template match="p">
10  <paragraphe><xsl:value-of select="."/></paragraphe>
11 </xsl:template>
12 </xsl:stylesheet>

```

xml2html.xml

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
3 <xsl:output method="html"/>

```

```

4 <xsl:template match="document">
5   <div id="srcXml">
6     <xsl:apply-templates select="paragraphe"/>
7   </div>
8 </xsl:template>
9 <xsl:template match="paragraphe">
10  <p><xsl:value-of select="."/></p>
11 </xsl:template>
12 </xsl:stylesheet>

```

> **Solution n°10**

Exercice p. 17

```

1 function loadXml (pPath){
2   var vXhr = new XMLHttpRequest();
3   vXhr.open("GET", pPath, false);
4   vXhr.send(null);
5   return rXml = vXhr.responseXML;
6 }
7 function alertXml (pDom) {
8   var xmlString = new XMLSerializer().serializeToString(pDom);
9   alert(xmlString);
10 }

```

> **Solution n°11**

Exercice p. 17

```

1 function execXslt (pXml, pXsl) {
2   var proc = new XSLTProcessor();
3   proc.importStylesheet(pXsl);
4   return proc.transformToFragment(pXml,document);
5 }

```

> **Solution n°12**

Exercice p. 17

```

1 function addPara (pText) {
2   vNewP = document.createElement("p");
3   vNewP.appendChild(document.createTextNode(pText));
4   document.getElementById("srcXml").appendChild(vNewP);
5 }

```

> **Solution n°13**

Exercice p. 17

```

1 function saveXml () {
2   var xsl = loadXml("html2xml.xml");
3   var xml = document.getElementById("srcXml");
4   var proc = new XSLTProcessor();
5   proc.importStylesheet(xsl);
6   var res = proc.transformToDocument(xml);
7   var resTxt = (new XMLSerializer()).serializeToString(res);

```

```
8 var data = "xml=" + encodeURIComponent(resTxt);
9 var vXHR = new XMLHttpRequest();
10 vXHR.open("POST", "saveXml.php", false);
11 vXHR.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
12 vXHR.send(data);
13 }
```

Bibliographie



Alexandre Brilliant, *XML : Cours et exercices*, Eyrolles, 2007 [ISBN 978-2212126914]

