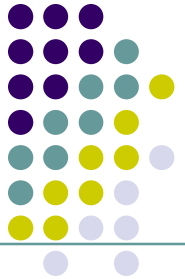
Université Gaston Berger de Saint-Louis

Année académique 2015-2016



XML et JavaScript

Références

1. XML Programming Success in a Day: Beginner's Guide to Fast, Easy, and Efficient Learning of XML Programming. Sam Key, 2015.

2. XML Programming: The Ultimate Guide to Fast, Easy, and Efficient Learning of XML Programming. Christopher Right, 2015

3. Beginning XML. Joe Fawcett and Danny Ayers, 2012.

4. XML, Cours et exercices. Modélisation, Schémas et DTD, design patterns, XSLT, DOM, Relax NG, XPath, SOAP, XQuery, XSL-FO, SVG, eXist. *Alexandre Brillant*, Édition : Eyrolles 2eédition, 2010

5. Schémas XML, Jean-Jacques Thomasson, Edition Eyrolles 2002

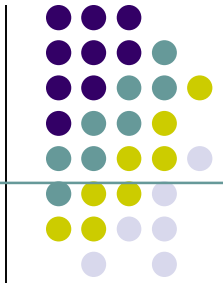
6. World Wide Web Consortium (W3C) : w3.org

7. World Wide Web School : w3schools.com/xml

8. Xml.developpez.com

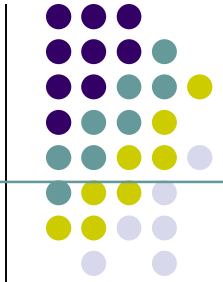
9. Youtube.com

Sommaire



- Introduction
- L'objet XMLHttpRequest
- Le parseur XML
- Le DOM XML
- Les nœuds du DOM XML
- Accès aux nœuds
- Les propriétés des nœuds
- Listing des nœuds
- Changer la valeur d'un nœud
- Suppression d'un nœud
- Remplacer un nœud
- Créer un nœud
- Ajouter un nœud
- Cloner un nœud

Introduction :

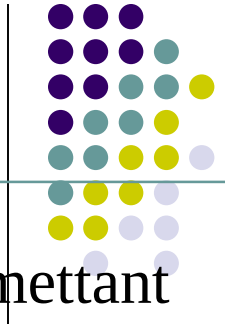


Au cours des cours XML et JavaScript, nous avons appris:

- Comment créer un document XML
- Comment ajouter des définitions dans un document XML en utilisant:
  - DTD - Définition du type de document
  - Schéma XML

Il reste maintenant à apprendre comment traiter et comment afficher un document XML.

Ce cours présente comment traiter et afficher le document XML en utilisant JavaScript et HTML.



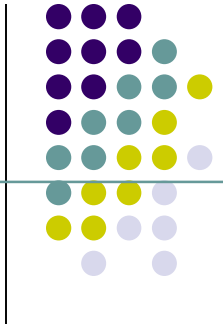
L’objet XMLHttpRequest

Les navigateurs ont un objet XMLHttpRequest intégré permettant d’accéder aux données à partir d'un serveur.

L’objet XMLHttpRequest est utilisé pour échanger des données avec un serveur.

Il permet de :

- mettre à jour une page web sans la recharger
- demander des données au serveur après le chargement de la page
- Recevoir des données du serveur après le chargement de la page
- Envoyer des données au serveur en arrière plan



L'objet XMLHttpRequest

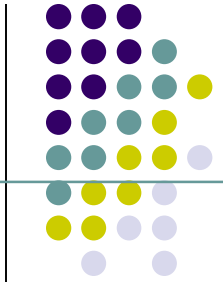
Envoi d'un objet XMLHttpRequest

Une syntaxe JavaScript courante pour l'utilisation de l'objet XMLHttpRequest ressemble à ceci:

Exemple

```
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      // Action typique à exécuter quand la document est prêt:
      document.getElementById("demo").innerHTML = xhttp.responseText;
    }
};

xhttp.open("GET", "filename", true);
xhttp.send();
```

L'objet XMLHttpRequest



Explication :

- La première ligne dans l'exemple ci-dessus crée un objet **XMLHttpRequest**

```
var xhttp = new XMLHttpRequest();
```

- La propriété **onreadystatechange** spécifie une fonction à exécuter chaque fois que l'état de l'objet XMLHttpRequest change:

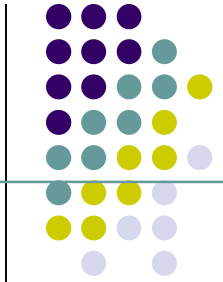
```
xhttp.onreadystatechange = function()
```

- Lorsque la propriété **readyState** est 4 et la propriété **status** est 200, la réponse est prête: `if (this.readyState == 4 && this.status == 200)`

- La **propriété responseText** renvoie la réponse du serveur sous forme de chaîne de caractères :

La chaîne peut être utilisée pour mettre à jour une page Web par :

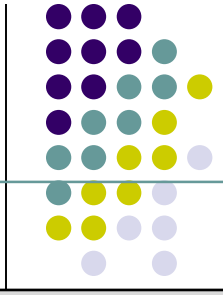
```
document.getElementById("demo").innerHTML = xhttp.responseText;
```



L'objet XMLHttpRequest

Méthodes d'objet XMLHttpRequest

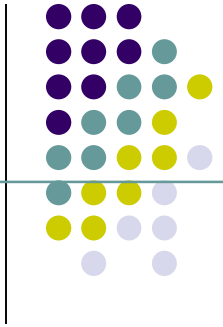
Méthode	Description
new XMLHttpRequest()	Crée un nouvel objet XMLHttpRequest
abort()	Annule la requête en cours
getAllResponseHeaders()	Renvoie l'information d'en-tête
getResponseHeader()	Renvoie des informations d'en-tête spécifiques
open(*method,url,async,user,psw*)	Spécifie la requête *method* : le type de requête GET ou POST *url* : l'emplacement du fichier *async* : true (asynchrone) ou false (synchrone) *user* : nom d'utilisateur facultatif *psw* : mot de passe facultatif
send()	Envoie une requête au serveur. Utilisé pour les requêtes GET
send(*string*)	Envoie une requête au serveur. Utilisé pour les requêtes POST
setRequestHeader()	Ajoute une paire étiquette/valeur à l'en-tête à envoyer



L'objet XMLHttpRequest

Propriétés de l'objet XMLHttpRequest

Propriétés	Description
onreadystatechange	Définit une fonction à appeler lorsque la propriété readyState change
readyState	Définit l'état de l'objet XMLHttpRequest. 0: requête non initialisée 1: connexion au serveur établie 2: requête reçue 3: traitement de la requête en cours 4: requête terminée et la réponse est prête
responseText	Renvoie la réponse sous forme de chaîne
responseXML	Renvoie la réponse sous forme de données XML
Status	Renvoie le numéro d'état d'une requête 200: "OK" 403: "Interdit" 404: "Pas trouvé" Pour une liste complète, **https://www.w3schools.com/tags/ref_httpmessages.asp**
statusText	envoie le texte d'état (par exemple "OK" ou "Not found")

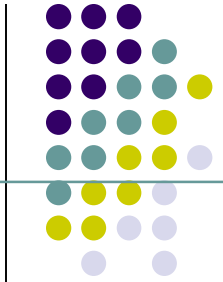


L'objet XMLHttpRequest

Pour d'anciennes versions d'IE (IE5 et IE6)

- Les anciennes versions d'Internet Explorer (IE5 et IE6) ne prennent pas en charge l'objet XMLHttpRequest.
- Pour gérer IE5 et IE6, on vérifie si le navigateur prend en charge l'objet XMLHttpRequest ou bien on crée un objet ActiveXObject:

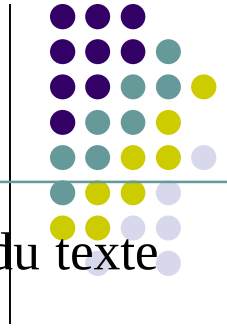
```
if (window.XMLHttpRequest) {
    // code pour navigateurs modernes
    xmlhttp = new XMLHttpRequest();
 } else {
    // code pour les anciens navigateurs de IE
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```



L'objet XMLHttpRequest

Exemple

```
<script>
function loadXMLDoc() {
   var xmlhttp;
   if (window.XMLHttpRequest) {
            xmlhttp = new XMLHttpRequest();
   } else {// code pour les anciens navigateurs de IE
            xmlhttp = new ActiveXObject("Microsoft.XMLHTTP ");
   }
   xmlhttp.onreadystatechange = function() {
       if (this.readyState == 4 && this.status == 200) {
            document.getElementById("demo").innerHTML = this.responseText;
       }
   };
   xmlhttp.open("GET", "xmlhttp_info.txt", true);
   xmlhttp.send();
}
</script>

<p id="demo"></p>
<button onclick="loadXMLDoc()">Testez</button>
```



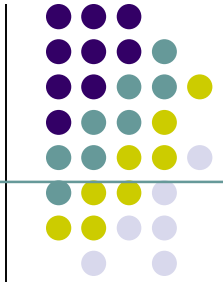
Le parseur XML

- Les principaux navigateurs ont un analyseur XML intégré pour convertir du texte en un objet DOM XML, y accéder et le manipuler.
- Le DOM XML (Document Object Model) définit les propriétés et les méthodes d'accès et d'édition d'un document XML.
- L'exemple suivant analyse une chaîne de caractères dans un objet DOM XML et extrait les informations avec JavaScript:

Exemple

```
<p id="demo"></p>
<script>
var text, parser, xmlDoc;

text = "<bookstore><book>" +
"<title>JavScript de base</title>" +
"<author>Moussa Lo</author>" +
"<year>2005</year>" +
"</book></bookstore>";
```

```
parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].child
Nodes[0].nodeValue;
</script>
```



Le parseur XML

Explication de l'exemple :

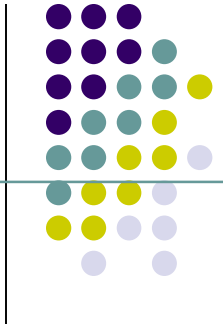
- Une chaîne de caractères est définie:

```
text = "<bookstore><book>" +
"<title>JavScript de base</title>" +
"<author>Moussa Lo</author>" +
"<year>2005</year>" +
"</book></bookstore>";
```

- Un analyseur DOM XML est créé par :
  parser = new DOMParser();

- Le parseur crée un nouvel objet DOM XML à l'aide de la chaîne de caractères :

  xmlDoc = parser.parseFromString(text,"text/xml");



Le parseur XML

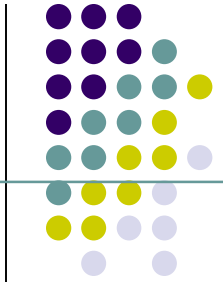
Pour les anciennes versions d'Internet Explorer

- Les anciennes versions d'Internet Explorer (IE5, IE6, IE7, IE8) ne prennent pas en charge l'objet DOMParser.

- Pour les prendre en compte, on vérifie si le navigateur prend en charge l'objet DOMParser ou on crée un objet ActiveXObject:

Exemple

```
if (window.DOMParser) {  // code pour navigateurs modernes
        parser = new DOMParser();
        xmlDoc = parser.parseFromString(text,"text/xml");
}
else {  // code pour anciens navigateurs de IE
        xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
        xmlDoc.async = false;
        xmlDoc.loadXML(text);
}
```



Le parseur XML

Parseur intégré de l'objet XMLHttpRequest

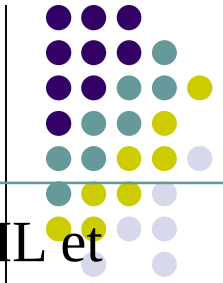
L' objet XMLHttpRequest possède un analyseur XML intégré.

- La **propriété responseText** renvoie la réponse sous forme de chaîne.
- La **propriété responseXML** renvoie la réponse en tant qu'objet DOM XML.

Si on souhaite utiliser la réponse en tant qu'objet DOM XML, on peut utiliser la propriété responseXML.

Exemple

```
xmlDoc = xmlhttp.responseXML;
txt = "";
x = xmlDoc.getElementsByTagName("ARTIST");
for (i = 0; i < x.length; i++) {
    txt += x[i].childNodes[0].nodeValue + "<br>";
}
document.getElementById("demo").innerHTML = txt;
```



Le DOM XML

Le DOM définit une norme pour accéder à des documents comme XML et HTML:

"Le modèle Objet de document (DOM) du W3C est une plate-forme et une interface indépendante du langage qui permet aux programmes et aux scripts d'accéder et de mettre à jour dynamiquement le contenu, la structure et le style d'un document."

Le DOM est divisé en 3 parties / niveaux :

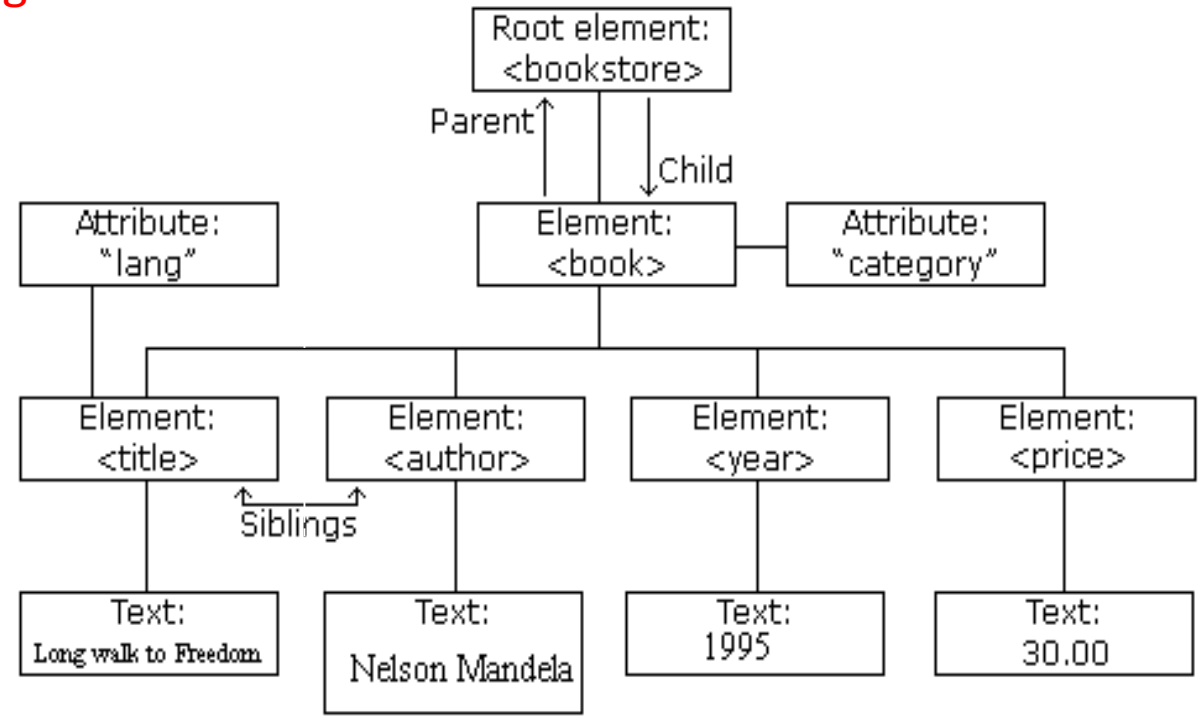
- Core DOM - modèle standard pour tout document structuré
- XML DOM - modèle standard pour les documents XML
- HTML DOM - modèle standard pour les documents HTML

Le DOM définit les **objets** et **propriétés** de tous les éléments du document, ainsi que les méthodes (**interface**) pour y accéder.

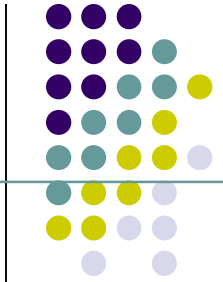
Le DOM XML

- Le DOM XML présente un document XML comme arborescence.
- Le DOM HTML présente un document HTML comme arborescence.
- Comprendre le DOM est nécessaire pour travailler avec HTML ou XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="History">
    <title lang="en">
      Long walk to freedom
    </title>
    <author>
      Nelson Mandela
    </author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  ...
</bookstore>
```



Le DOM XML



Le DOM HTML

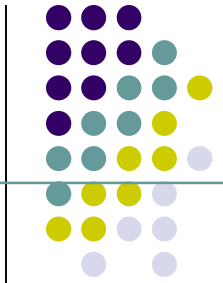
- Tous les éléments HTML sont accessibles via le DOM HTML.
- Le DOM HTML définit les objets, les propriétés et les méthodes de tous les éléments HTML.

Exemple :

Modifier la valeur d'un élément HTML
Cet exemple modifie la valeur d'un élément HTML avec id = "demo":

```
<h1 id="demo">Ceci est une entête</h1>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

Le DOM XML



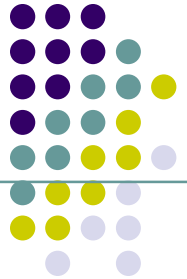
Le DOM HTML

Exemple : Cet exemple modifie la valeur du premier élément <h1> dans un document HTML:

```
<h1>Ceci est une entête</h1>

<h1>Ceci est une entête</h1>

<script>
document.getElementsByTagName("h1")[0].innerHTML =
"Hello World!";
</script>
```

Remarque: Même si le document HTML ne contient qu'un seul élément <h1>, vous devez toujours spécifier l'index du tableau [0], car la méthode getElementsByTagName () renvoie toujours un tableau.

Le DOM XML



Le DOM XML

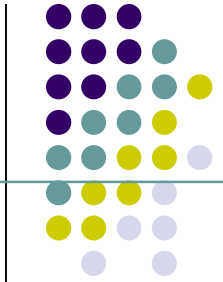
- Tous les éléments XML sont accessibles via le DOM XML.
- Le DOM XML définit les **objets**, les **propriétés** et les **méthodes** de tous les éléments XML.

Le DOM XML est :

- Un modèle d'objet standard pour XML
- Une interface de programmation standard pour XML
- Plate-forme indépendant du langage
- Une norme W3C

En d'autres termes: Le DOM XML est un standard pour accéder, modifier, ajouter ou supprimer des éléments XML.

Le DOM XML



Interface de programmation

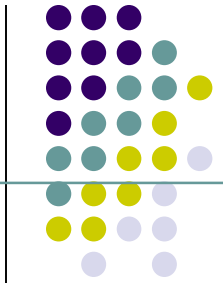
- Le DOM modélise XML comme un ensemble d'objets de nœud. Les nœuds peuvent être accédés avec JavaScript ou d'autres langages de programmation.

- L'interface de programmation pour le DOM est définie par un ensemble de propriétés et de méthodes standard.

- Les **propriétés** désignent de quoi est composé un objet (ex: nodeName est "book").

- Les **méthodes** désignent souvent ce que fait l’objet (supprimer un "livre" par exemple).

Le DOM XML



Propriétés du DOM XML

Voici quelques propriétés DOM typiques:

- x.nodeName - le nom de x
- x.nodeValue - la valeur de x
- x.parentNode - le nœud parent de x
- x.childNodes - les nœuds enfants de x
- x.attributes - les attributs noeuds de x

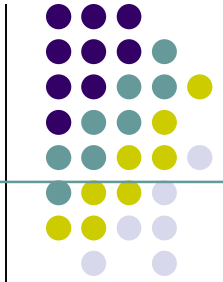
Remarque: Dans la liste ci-dessus, x est un objet nœud..

Méthodes du DOM XML

- x.getElementsByTagName(*name*) : récupère tous les elts ayant un nom de balise name
- x.appendChild(*node*) - insère un nœud enfant en x
- x.removeChild(*node*) - supprime un nœud enfant de x

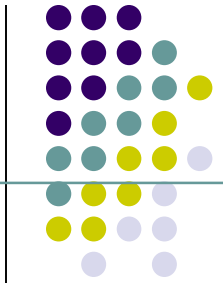
Remarque: Dans la liste ci-dessus, x est un objet nœud.

Le DOM XML



```
<!DOCTYPE html>
<html><body>
<p id="demo"></p>
<script>
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
        myFunction(xhttp);
    }
};
xhttp.open("GET", "books.xml", true);
xhttp.send();

function myFunction(xml) {
    var xmlDoc = xml.responseXML;
    document.getElementById("demo").innerHTML =
    xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
}
</script>
</body>
</html>
```

Le DOM XML



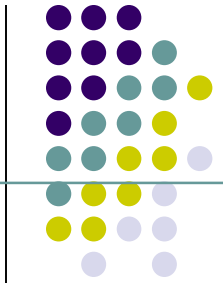
Exemple expliqué

- **xmlDoc -** L'Objet DOM XML créé par l'analyseur.
- **getElementsByTagName("title")[0] -** le 1e element <title>
- **childNodes[0] -** le premier enfant de l'élt <title> (le nœud texte)
- **nodeValue -** la valeur du nœud (le texte lui-même)

Exemple 2: Chargement d'une chaîne XML

Cet exemple charge une chaîne de texte dans un objet DOM XML et extrait les informations de celui-ci avec JavaScript:

Le DOM XML



```
<html>
<body>
<p id="demo"></p>
<script>
var text, parser, xmlDoc;
text = "<bookstore><book>" +
"<title>Long walk to freedom</title>" +
"<author>Nelson Mandela</author>" +
"<year>1993</year>" +
"</book></bookstore>";

parser = new DOMParser();
xmlDoc = parser.parseFromString(text,"text/xml");

document.getElementById("demo").innerHTML =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
</script>
</body>
</html>
```

Les nœuds du DOM XML

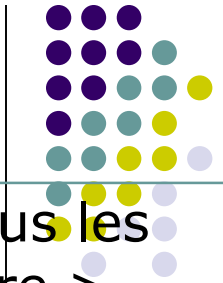
Selon le DOM, tout dans un document XML est un nœud.

- Le document entier est un nœud document
- Chaque élément XML est un nœud élément
- Le texte dans les éléments XML sont des nœuds texte
- Chaque attribut est un nœud attribut
- Les commentaires sont des nœuds commentaire

Exemple: Considérons ce document XML suivant:

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
 <book category="History">
   <title lang="en">Long walk to freedom</title>
   <author>Nelson Mandela</author>
   <year>2005</year>
   <price>30.00</price>
 </book>
  ...
 </bookstore>
```

Les nœuds du DOM XML



- Le nœud racine du document est nommé <bookstore>. Tous les autres nœuds du document sont contenus dans < bookstore >.
- Le nœud racine <bookstore> contient quatre nœuds <book>.
- Le premier nœud <book> contient quatre nœuds: <title>, <author>, <year> et <price>, qui contiennent chacun un nœud texte, "Long walk to freedom", "Nelson Mandela" et "30.00".

Le texte est toujours stocké dans les nœuds texte

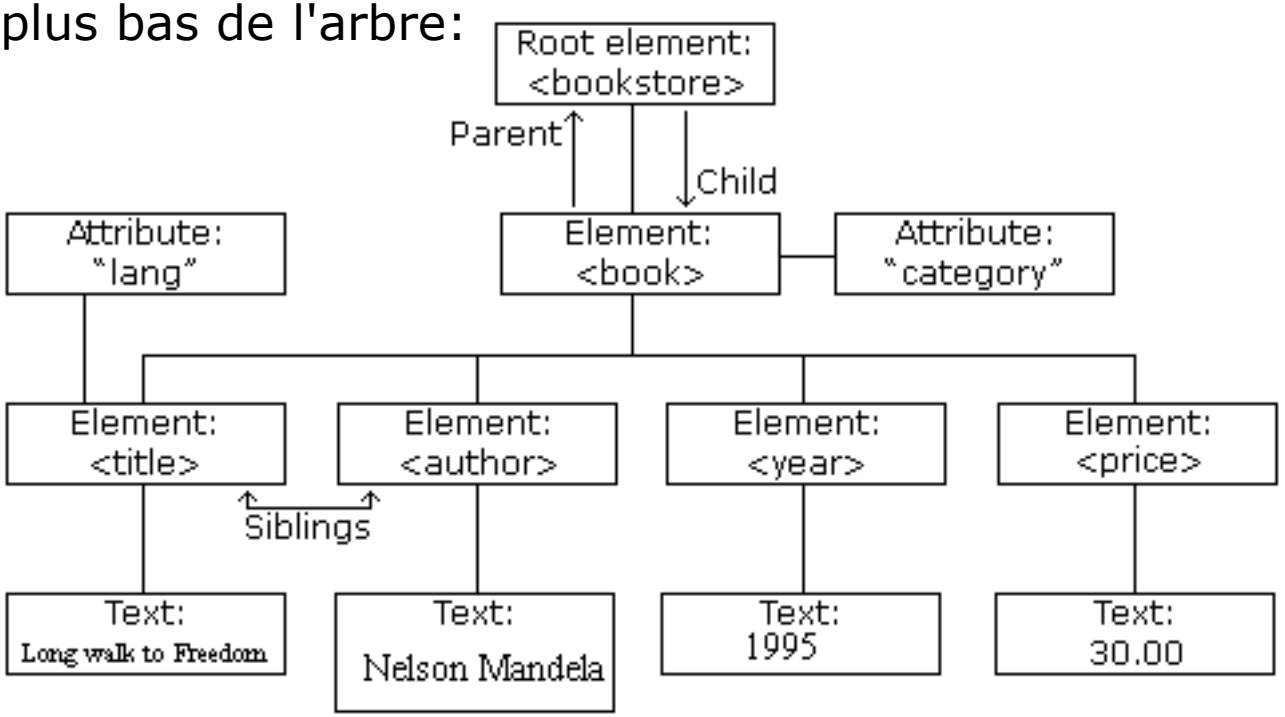
- Une erreur courante dans le traitement DOM est de s'attendre à ce qu'un nœud élément contienne du texte.
- Cependant, le texte d'un nœud élément est stocké dans un nœud texte.
- Dans l'exemple: <year> 2005 </year>, le nœud élément <year> contient un nœud de texte avec la valeur "2005".

"2005" **n'est pas la valeur** de l'élément <year>!

Les nœuds du DOM XML

L'arborescence du DOM XML

- Le DOM XML visualise un document XML sous la forme d'une arborescence appelée nœud-arbre.
- Tous les nœuds peuvent être accédés par l'arbre. Leur contenu peut être modifié ou supprimé, et de nouveaux éléments peuvent être créés.
- L'arbre de nœud montre l'ensemble des nœuds, et les connexions entre eux. L'arbre commence par le nœud racine et se ramifie vers les nœuds texte au niveau le plus bas de l'arbre:



Les nœuds du DOM XML

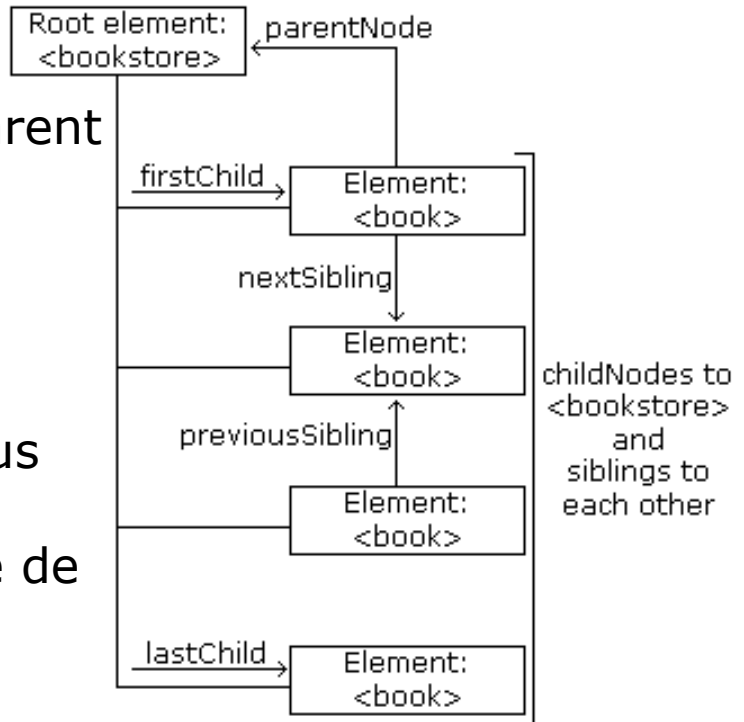
Nœud Parents, enfants et frères

Les nœuds de l'arborescence ont une relation hiérarchique entre eux.

Les termes parent, enfant et frère sont utilisés pour décrire les relations. Les nœuds parents ont des enfants. Les enfants du même niveau sont appelés frères.

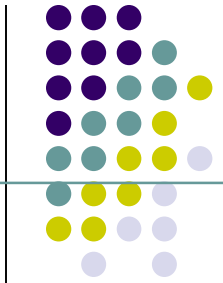
- Dans l'arborescence, le nœud supérieur est appelé racine
- Chaque nœud, sauf la racine, possède exactement un nœud parent
- Un nœud peut avoir 0 ou plusieurs enfants
- Une feuille est un nœud sans enfants
- Les frères sont des nœuds avec le même parent

L'image suivante illustre une partie de l'arborescence et la relation entre les nœuds:



Comme les données XML sont structurées sous forme d'arborescence, elles peuvent être parcourues sans connaître la structure exacte de l'arborescence et sans connaître le type de données contenues.

Les nœuds du DOM XML



Premier enfant – Dernier enfant

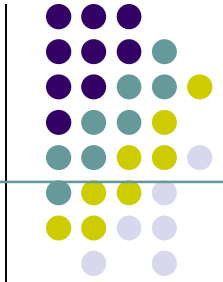
Regardez le fragment XML suivant::

```
<bookstore>
  <book category="history">
    <title lang="en">Long walk to freedom</title>
    <author>Nelson Mandela</author>
    <year>1993</year>
    <price>30.00</price>
  </book>
</bookstore>
```

- Dans le code ci-dessus, l'élément <title> est le premier enfant de l'élément <book> et l'élément <price> est le dernier enfant de l'élément <book>.

- De plus, l'élément <book> est le nœud parent des éléments <title>, <author>, <year> et <price>.

Accès aux noeuds



On peut accéder à un nœud de trois façons:

1. En utilisant la méthode getElementsByTagName ()
2. En bouclant sur l'arbre des nœuds.
3. En navigant dans l'arborescence des nœuds, en utilisant les relations entre les nœuds.

- **La méthode getElementsByTagName ()**

Elle renvoie tous les éléments avec un nom de balise spécifié.
Syntaxe : *noeud* .getElementsByTagName(*"tagname"*);

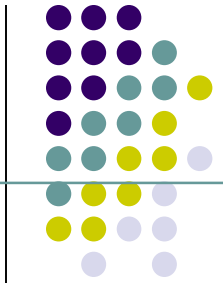
Exemple : x.getElementsByTagName("title");

renvoie tous les éléments <title> sous l'élément x:
Notez que l'exemple ci-dessus ne renvoie que des éléments <title> sous le nœud x.

Pour renvoyer tous les éléments <title> dans le document XML, utilisez:
xmlDoc.getElementsByTagName("title");

Où **xmlDoc** est le document lui-même (nœud document).

Accès aux noeuds



- **La méthode getElementsByTagName ()**

Types de nœuds

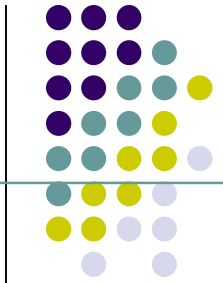
La propriété **documentElement** du document XML est le nœud racine.

La propriété **nodeName** d'un nœud est le nom du nœud.

La propriété **nodeType** d'un nœud est le type du nœud.

D’autres propriétés d’un nœud seront vus dans la suite.

Accès aux noeuds



- **La méthode getElementsByTagName ()**

Liste des nœuds:

La méthode getElementsByTagName () renvoie une liste de nœuds. Une liste de nœuds est un tableau de nœuds.

```
x = xmlDoc.getElementsByTagName("title");
```

Les éléments <title> de x peuvent être accédés par leur indice qui commencent par 0. y = x[2] donne le troisième élément <title>;

Longueur de la liste des nœuds :

La propriété length donne la longueur d'une liste de nœuds (le nombre de nœuds). Il peut être utilisé pour parcourir une liste de nœuds.

Exemple

```
var x = xmlDoc.getElementsByTagName("title");

for (i = 0; i <x.length; i++) {
  // code
  }
```

Accès aux noeuds

- **La méthode d'accès transversal**

Le code suivant passe en revue les nœuds enfants du nœud racine:

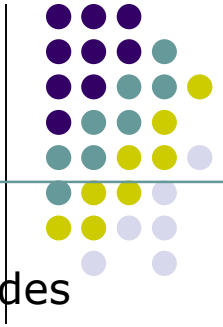
Exemple :

```
txt = "";
x = xmlDoc.documentElement.childNodes;

for (i = 0; i <x.length; i++) {
  // Process only element nodes (type 1)
  if (x[i].nodeType == 1) {
    txt += x[i].nodeName + "<br>";
  }
}
```

Explication:

1. On charge "books.xml" dans xmlDoc
2. On accède aux nœuds enfants de l'élément racine (xmlDoc)
3. Et pour chaque nœud enfant, on vérifie son type. Si le type de nœud est "1", il s'agit d'un nœud élément :
4. Alors on affiche le nom du nœud s'il s'agit d'un nœud élément

Accès aux noeuds



- **La méthode par navigation dans les relations entre les nœuds**

Le code suivant permet de naviguer dans l'arborescence des nœuds à l'aide des relations entre nœud:

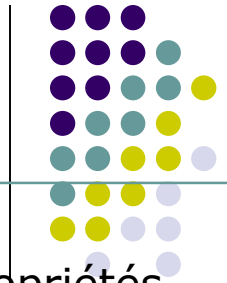
Exemple :

```
x = xmlDoc.getElementsByTagName("book")[0];
xlen = x.childNodes.length;
y = x.firstChild;

txt = "";
for (i = 0; i <xlen; i++) {
  // Process only element nodes (type 1)
  if (y.nodeType == 1) {
    txt += y.nodeName + "<br>";
  }
  y = y.nextSibling;
}
```

Explication : Supposons que " books.xml " soit chargé dans xmlDoc

1. On obtient les nœuds enfants du premier élément de livre
2. Puis on définit la variable "y" comme étant le premier nœud enfant du premier élément de livre
3. Pour chaque nœud enfant (commençant par le premier nœud enfant "y"):
4. On vérifiez le type de nœud. Si le type est "1", il s'agit d'un nœud élément
5. et on affiche le nom du nœud s'il s'agit d'un nœud élément
6. On définit la variable "y" comme étant le nœud frère suivant et on boucle

Les propriétés des nœuds



Propriétés d'un nœud

Dans le DOM XML, chaque nœud est un **objet** qui a des méthodes et des propriétés, accessibles et manipulées par JavaScript.

Trois propriétés importantes d'un nœud sont:

- nodeName
- nodeValue
- nodeType

La propriété nodeName

La propriété nodeName spécifie le nom d'un nœud.

- NodeName est en lecture seule
- NodeName d'un nœud élément est le même que le nom de la balise
- NodeName d'un nœud attribut est le nom de l'attribut
- NodeName d'un nœud texte est toujours #text
- NodeName du nœud document est toujours #document

Les propriétés des nœuds

La propriété nodeValue

La propriété nodeValue spécifie la valeur d'un nœud.

- NodeValue pour les nœuds élément est indéfinie
- NodeValue pour les nœuds de texte est le texte lui-même
- NodeValue pour les nœuds attribut est la valeur d'attribut

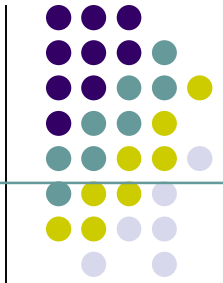
Exemple: Obtenir la valeur d'un élément
Le code suivant récupère la valeur de nœud de texte du premier élément <title>:

```
var x = xmlDoc.getElementsByTagName("title")[0].childNodes[0];
var txt = x.nodeValue;
```

Explication:

1. On suppose que " books.xml " est chargé dans xmlDoc
2. On cherche le nœud texte du premier nœud élément <title>
3. On définit la variable txt comme étant la valeur du nœud texte

Les propriétés des nœuds



Modifier la valeur d'un élément

Le code suivant modifie la valeur du nœud texte du premier élément <title>:

```
var x = xmlDoc.getElementsByTagName("title")[0].childNodes[0];
x.nodeValue = "Easy Cooking";
```

La propriété nodeType

- La propriété nodeType spécifie le type de nœud.
- NodeType est en lecture seule.
- Les types de nœuds les plus importants sont:

Type de nœud	Valeur du type
Element	1
Attribut	2
Texte	3
Commentaire	8
Le document	9

Listing des nœuds

Lors de l'utilisation des propriétés ou des méthodes comme childNodes ou getElementsByTagName (), un objet ayant la liste de nœuds est renvoyé.

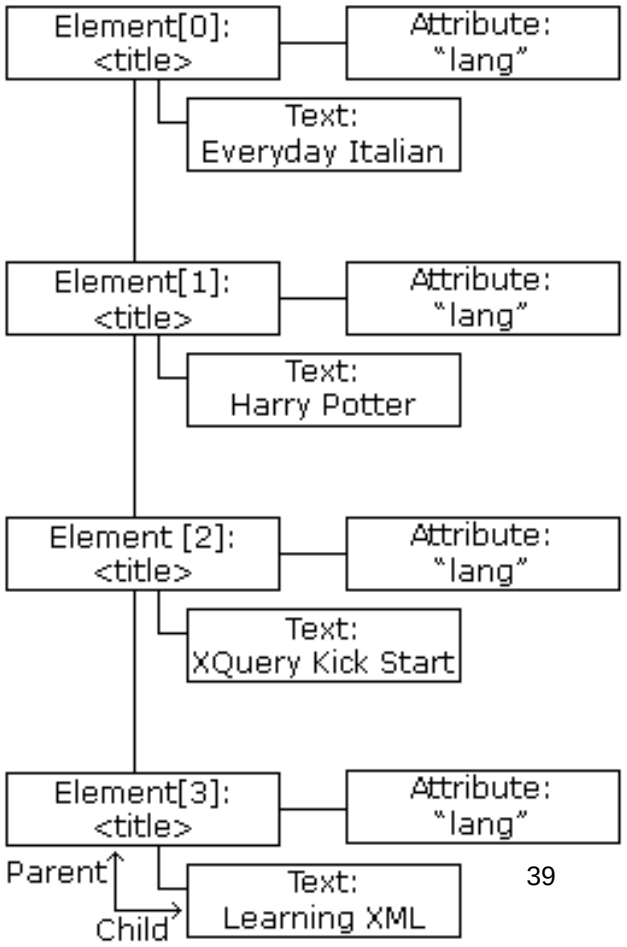
Un objet liste de nœuds représente une liste de nœuds, dans le même ordre que dans le XML.

Les nœuds de la liste de nœuds sont accédés avec des numéros d'index commençant par 0.

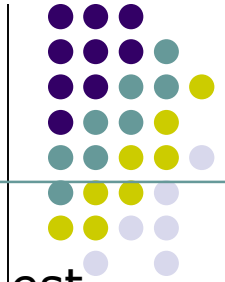
L'image ci-contre représente une liste de nœuds des éléments <title> dans " books.xml ":

Supposons "books.xml" chargé dans xmlDoc.
`x = xmlDoc.getElementsByTagName("title");`
renvoie une liste de nœuds des éléments de title.
où x est un objet de liste de nœuds.

`var txt = x[0].childNodes[0].nodeValue;`
renvoie le texte du premier élément <title> dans la liste des nœuds de x.



Listing des nœuds



Longueur de la liste des nœuds

Un objet liste de nœud se maintient lui-même à jour. Si un élément est supprimé ou ajouté, la liste est automatiquement mis à jour.

La propriété **length** d'une liste de nœud est le nombre de nœuds dans la liste.

Exemple: pour avoir le nombre d'éléments <title> dans "books.xml":

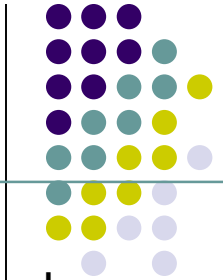
```
x = xmlDoc.getElementsByTagName('title').length;
```

La longueur de la liste des noeuds peut être utilisé pour boucler sur tous les éléments dans la liste.

Exemple

```
x = xmlDoc.getElementsByTagName('title');
xLen = x.length;

for (i = 0; i <xLen; i++) {
    txt += x[i].childNodes[0].nodeValue) + " ";
}
```

Listing des nœuds



Liste des nœuds attribut
La propriété **attributes** d'un nœud élément renvoie la liste des nœuds attribut.

Cette liste est similaire à une liste de nœuds, à l'exception de certaines différences dans les méthodes et les propriétés.

Exemple: Liste des nœuds attribut du 1e élément <book> de "books.xml":

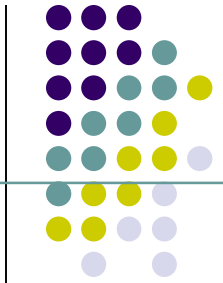
```
x = xmlDoc.getElementsByTagName('book')[0].attributes;
```

A la suite, x.length et x.getNamedItem () peuvent être utilisé pour retourner un nœud attribut.

Exemple: Obtenir la valeur de l'attribut "catégorie", et le nombre d'attributs, d'un livre:

```
x = xmlDoc.getElementsByTagName("book")[0].attributes;
txt = x.getNamedItem("category").nodeValue + " " + x.length;
```

Listing des nœuds



La valeur d'un attribut

- Avec la méthode **getAttribute ()**

Le code suivant récupère la valeur texte de l'attribut "lang" du premier élément <title>:

Exemple

```
x = xmlDoc.getElementsByTagName("title")[0];
txt = x.getAttribute("lang");
```

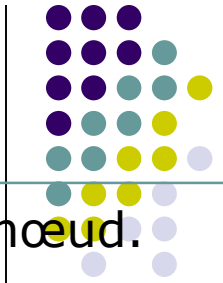
Obtenir un nœud attribut

La méthode **getAttributeNode ()** retourne un **nœud d'attribut.**
Le code suivant récupère la valeur de texte de l'attribut "lang" du premier élément <title>:

Exemple

```
x = xmlDoc.getElementsByTagName("title")[0];
y = x.getAttributeNode("lang");
txt = y.nodeValue;
```

Changer la valeur d'un nœud



- La propriété **nodeValue** est utilisée pour modifier la valeur d'un nœud.
- La méthode **setAttribute ()** est utilisée pour modifier celle d'un d'attribut.

Modifier la valeur d'un élément

- Dans le DOM, tout est nœud et un nœud élément n`a pas de valeur texte.
- La valeur texte d'un nœud élément est stockée dans un nœud enfant. Ce nœud est appelé nœud texte.
- Pour modifier la valeur texte d'un élément, vous devez modifier la valeur du nœud texte de l'élément.

Modifier la valeur d'un nœud texte

La propriété **nodeValue** est utilisé pour modifier la valeur d'un nœud texte.

Exemple

```
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue = "Vale
ur "
```

Suppression de nœuds

- La méthode **removeChild ()** supprime un nœud spécifié.
- La méthode **removeAttribute ()** supprime un attribut spécifié.

Supprimer un nœud élément

La méthode **removeChild ()** supprime un nœud spécifié.

Lorsqu'un nœud est supprimé, tous ses nœuds enfants sont également supprimés.

Exemple : Suppression du premier élément <book> du fichier xml chargé:

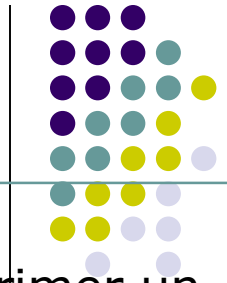
```
y = xmlDoc.getElementsByTagName("book")[0];

xmlDoc.documentElement.removeChild(y);
```

Supprimer le nœud élément courant

Exemple

```
x = xmlDoc.getElementsByTagName("book")[0];

x.parentNode.removeChild(x);
```

Suppression de nœuds



Supprimer un nœud texte

La méthode **removeChild ()** peut également être utilisé pour supprimer un nœud de texte:

Exemple

```
x = xmlDoc.getElementsByTagName("title")[0];
y = x.childNodes[0];
x.removeChild(y);
```

Effacer un nœud de texte

La propriété **nodeValue** peut être utilisé pour modifier la valeur d'un nœud texte:

Exemple

```
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue = "";
```

Supprimer un nœud d'attribut par son nom

La méthode **removeAttribute ()** supprime un nœud attribut par son nom.

Exemple:

```
x = xmlDoc.getElementsByTagName("book");
x[0].removeAttribute("category");
```

Suppression de nœuds

Supprimer un nœud attribut par objet

La méthode **removeAttributeNode ()** supprime un nœud d'attribut, en utilisant l'objet nœud en tant que paramètre.

Exemple: removeAttributeNode (x)
Ce code supprime tous les attributs de tous les éléments <book>:
Exemple

```
x = xmlDoc.getElementsByTagName("book");

for (i = 0; i < x.length; i++) {
    while (x[i].attributes.length > 0) {
        attnode = x[i].attributes[0];
        old_att = x[i].removeAttributeNode(attnode);
    }
}
```

Remplacer un nœud

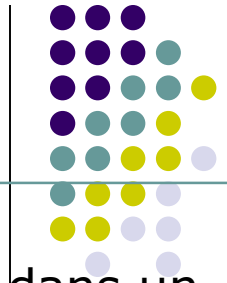
- La méthode **replaceChild ()** remplace un nœud spécifié.
- La propriété **nodeValue** remplace le texte d'un nœud texte.

Remplacer un nœud élément

Exemple: Le code suivant remplace le premier élément <book>:

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.documentElement;

//Créer un élément book, un elt title et un nœud texte
newNode=xmlDoc.createElement("book");
newTitle=xmlDoc.createElement("title");
newText=xmlDoc.createTextNode("A Notebook");

//Ajout du neoud texte au nœud title,
newTitle.appendChild(newText);

//Ajout du nœud title au nœud book
newNode.appendChild(newTitle);

y=xmlDoc.getElementsByTagName("book")[0]
//Remplacer le premier nœud book par le nouveau noeud
x.replaceChild(newNode,y);
```

Remplacer un nœud



Remplacer des données dans un nœud texte

La méthode replaceData () est utilisée pour remplacer des données dans un nœud texte.

La méthode replaceData () possède trois paramètres:

- indice - l'indice à partir duquel on remplace
- taille - le nombre de caractères à remplacer
- chaine - La chaîne à insérer

Exemple :

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];
x.replaceData(0,8,"Easy");
```

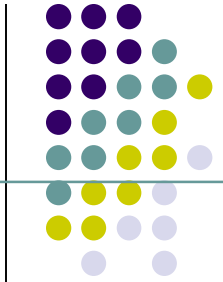
Avec la propriété nodeValue

Remplacer les données dans un nœud texte est plus facile avec nodeValue.

Exemple

```
xmlDoc=loadXMLDoc("books.xml");
x=xmlDoc.getElementsByTagName("title")[0].childNodes[0];
x.nodeValue="Easy Italian";
```

Créer un nœud



Créer un nouveau nœud élément

La méthode **createElement ()** crée un nouveau nœud élément:

Exemple

```
newElement = xmlDoc.createElement("edition");

xmlDoc.getElementsByTagName("book")[0].appendChild(newElement);
```

Créer un nouveau nœud attribut

La méthode **createAttribute ()** est utilisée pour créer un nouveau nœud attribut. Si l'attribut existe déjà, il est remplacé par le nouveau.

Exemple

```
newAtt = xmlDoc.createAttribute("edition");
newAtt.nodeValue = "first";

xmlDoc.getElementsByTagName("title")[0].setAttributeNode(newAtt);
```

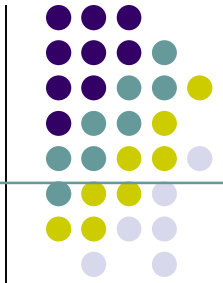
Créer un attribut à l'aide de setAttribute ()

La méthode **setAttribute ()** peut être utilisé pour créer un nouvel attribut.

Exemple

```
xmlDoc.getElementsByTagName('book')[0].setAttribute("edition","first");
```

Créer un nœud



Créer un nœud de texte

La méthode **createTextNode ()** crée un nouveau nœud texte:

Exemple

```
newEle = xmlDoc.createElement("edition");
newText = xmlDoc.createTextNode("first");
newEle.appendChild(newText);

xmlDoc.getElementsByTagName("book")[0].appendChild(newEle);
```

Créer un nœud section CDATA

La méthode **createCDATASection ()** crée une nvelle nœud section CDATA.

Exemple

```
newCDATA = xmlDoc.createCDATASection("Special Offer & Book Sale");

xmlDoc.getElementsByTagName("book")[0].appendChild(newCDATA);
```

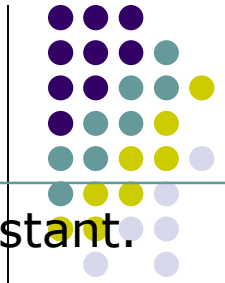
Créer un nœud de commentaire

La méthode **createComment ()** crée un nouveau nœud commentaire.

Exemple

```
newComment = xmlDoc.createComment("Revised March 2015");

xmlDoc.getElementsByTagName("book")[0].appendChild(newComment);
```

Ajout de noeuds



La méthode **appendChild ()** ajoute un nœud enfant à un nœud existant.
Le nouveau nœud est ajouté à la fin.

Exemple

```
newEle = xmlDoc.createElement("edition");

xmlDoc.getElementsByTagName("book")[0].appendChild(newEle);
```

Insérer un nœud - insertBefore ()

La méthode **insertBefore ()** insère un nœud avant un nœud enfant spécifié.

Exemple

```
newNode = xmlDoc.createElement("book");

x = xmlDoc.documentElement;
y = xmlDoc.getElementsByTagName("book")[3];

x.insertBefore(newNode,y);
```

Ajout de noeuds

Ajouter un nouvel attribut

La méthode **setAttribute ()** définit la valeur d'un attribut.

Exemple :

```
xmlDoc.getElementsByTagName('book')[0].setAttribute("edition","first");
```

Ajouter du texte à un nœud de texte - insertData ()

La méthode **insertData ()** insère des données dans un nœud texte existant. Elle possède deux paramètres:

- indice - l'indice à partir de laquelle on insère des caractères
- chaine - La chaîne à insérer

Exemple :

```
xmlDoc.getElementsByTagName("title")[0].childNodes[0].insertData(0,"Eas
y ");
```

Cloner un noeud

Copier un nœud

La méthode **cloneNode ()** crée une copie d'un nœud spécifié.
Elle a un paramètre (vrai ou faux) qui indique si le nœud cloné doit inclure tous les attributs et nœuds enfants du nœud d'origine.

Exemple

```
oldNode = xmlDoc.getElementsByTagName('book')[0];
newNode = oldNode.cloneNode(true);
xmlDoc.documentElement.appendChild(newNode);
```