Document Object Model (DOM)

Jean-Claude Charr

Maître de conférences

IUT de Belfort – Montbéliard

Université de Franche Comté

Description générale

- Définit un standard pour accéder aux documents structurés comme HTML ou XML et modifier leurs contenus
- C'est une spécification indépendante du langage de programmation
- Définit les propriétés des éléments et les méthodes pour accéder aux éléments

HTML DOM

- Tous les éléments d'un document HTML sont des noeuds :
  - Tout le document est un "document node"
  - Tout élément html est un "element node"
  - Le texte dans un élément HTML un "text node"
  - Les attributs d'un élément HTML sont des "attribute nodes"

Ex: <html>

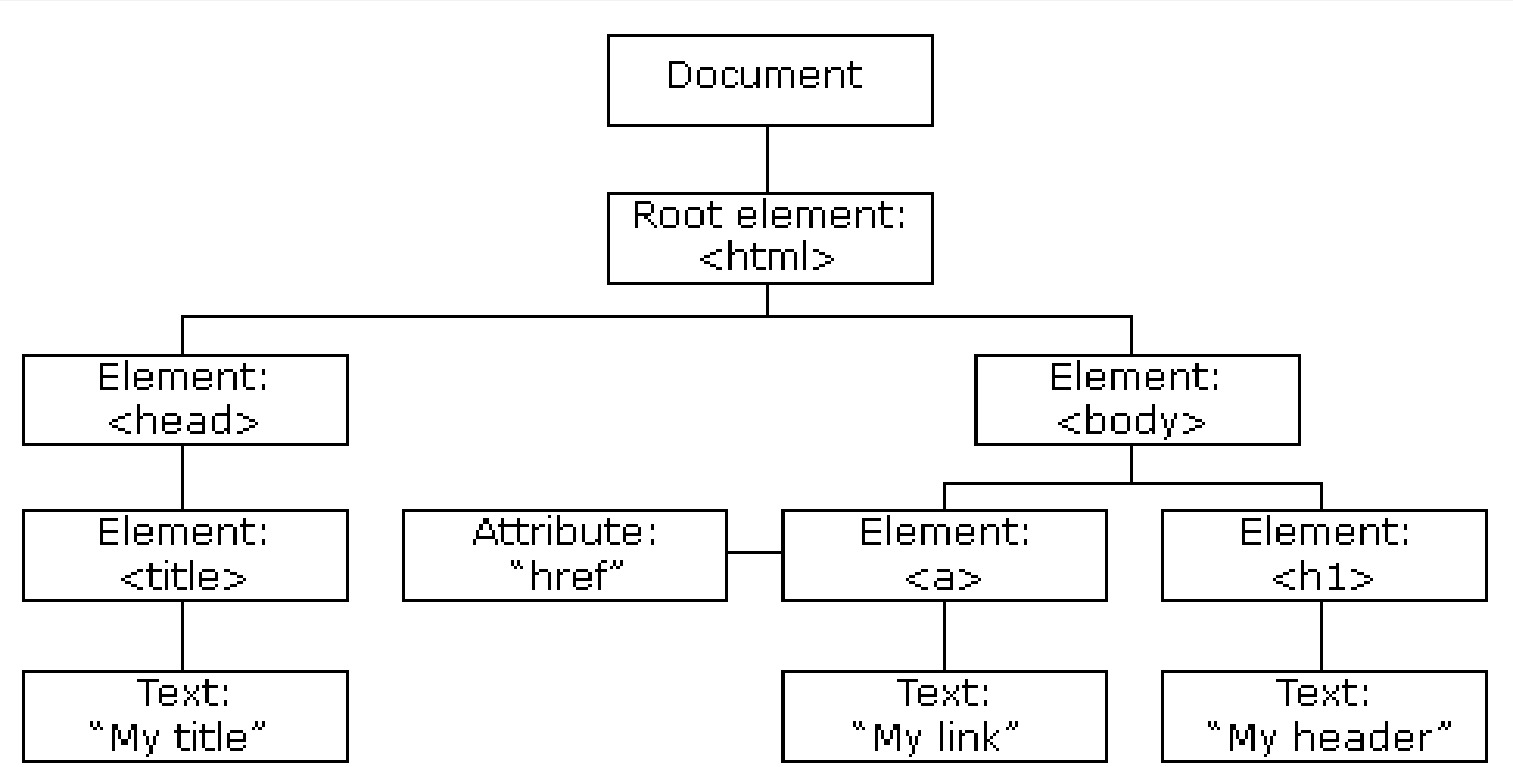
<head> <title>DOM Tutorial</title> </head>

<body> Lesson one </body>

</html>

La structure d'un document HTML en arbre (1/3)

- Un document html peut être structuré en un arbre de noeuds



La structure d'un document HTML en arbre (2/3)

- En code HTML :

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <h1>My header</h1>
    <a href=”google.com”>My link</a>
  </body>
</html>
```

La structure d'un document HTML en arbre (3/3)

- Les noeuds dans l'arbre de noeuds ont une structure hiérarchique :
  - Le premier noeud de l'arbre est appelé la racine
  - Tout noeud, sauf la racine, a un seul noeud père
  - Un noeud peut avoir plusieurs fils
  - Une feuille est un noeud sans fils
  - Les frères sont des noeuds ayant le même père

Les propriétés des noeuds (1/3)

- Tout élément HTML DOM possède des propriétés auquel on peut accéder et modifier avec un langage de programmation comme JavaScript, exemples :
  - x.innerHTML – la valeur du texte dans le noeud x
  - x.nodeName – le nom du noeud x
  - x.nodeValue – la valeur du noeud x
  - x.parentNode – le noeud père du noeud x
  - x.childNodes – les noeuds fils du noeud x
  - x.attributes – les attributs du noeud x

Les propriétés des noeuds (2/3)

- nodeName :
  - N'est pas éditable
  - Pour un élément c'est le mot clé
  - Pour un attribut c'est le nom de l'attribut
  - Pour un noeud texte c'est #text
  - Pour un noeud document c'est #document
- nodeValue :
  - Pour un élément c'est indéfini
  - Pour un attribut c'est la valeur de l'attribut
  - Pour un noeud texte c'est le texte entre les 2 balises

Les propriétés des noeuds (3/3)

- NodeType :
  - 1 pour les éléments
  - 2 pour les attributs
  - 3 pour les noeuds de texte
  - 8 pour les commentaires
  - 9 pour le document

Les méthodes des noeuds

- Tout élément HTML DOM possède des méthodes qui permettent de le séléctionner, le modifier ou ajouter un nouveau élément, exemples :
  - x.getElementById(id) – retourne l'élément ayant comme identifiant ”id”
  - x.getElementsByTagName(name) – retourne les éléments ayant le nom clé ”name”
  - x.appendChild(node) – ajouter un noeud fils x
  - x.removeChild(node) – supprimer un noeud fils x

Exemple sur les propriétés et les méthodes

```
<html>
  <body>
    <p id="intro">Hello World!</p>
    <script type="text/javascript">
      txt=document.getElementById("intro").innerHTML;
      document.write("<p>Du paragraphe intro : " + txt + "</p>");
      txt=document.getElementById("intro").childNodes[0].nodeValue;
      document.write("<p>Du paragraphe intro : " + txt + "</p>")
    </script>
  </body>
</html>
```

Accés aux noeuds

- document.getElementById("intro");
- x=document.getElementsByTagName("p");

  x.length; taille de la liste

  x[1]; deuxième élément de la liste
- parentNode, firstChild, and lastChild

  Ex : document.getElementById("intro").firstChild;
- document.documentElement; root
- document.body; body

Modifier un noeud

- document.body.bgColor="lavender";
- document.getElementById("p1").innerHTML="Newtext!";
- document.getElementById("p1").style.fontFamily="Arial";
- <input type="button" onclick = "document.body.style.backgroundColor='lavender';" value="color" />

Collections

- C'est un groupe de noeuds similaires.
- L'objet Document a les propriétés qui contiennent les collections d'images, liens, forms, … de la page HTML.
- Exemple :

```
var linkslist = document.links;
for ( var i = 0; i <linklist.length ; i++ )
    alert(linklist[i].innerHTML);
```

eXtensible Markup Language

- Document XML => Document structuré avec des balises comme HTML
- XML pour stocker et transporter les données et HTML pour afficher les données
- XML et HTML sont complémentaires
- Les balises XML ne sont pas prédéfinies, chacun peut inventer ses propre balises
- XML est une recommandation de W3C et indépendant des systèmes utilisés

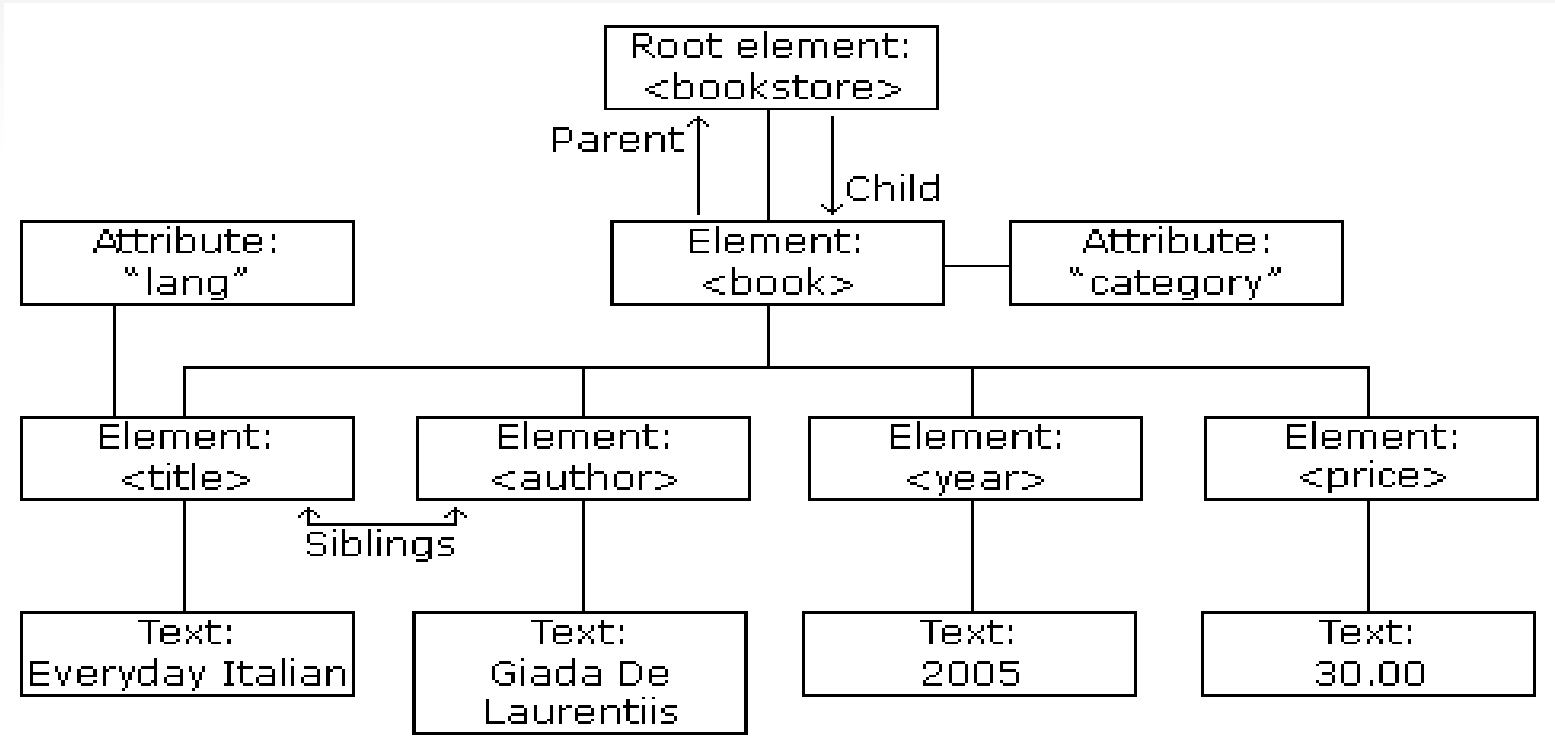
Pourquoi utiliser XML?

- Pour séparer les données des fichiers HTML
- Modifier dynamiquement les données affichées par un fichier HTML sans devoir l'éditer
- Format texte simple permet de transférer facilement les données entre les systèmes incompatibles.
- La majorité des applications peuvent lire les fichiers XML => Plusieurs applications peuvent accéder au fichier de données sans devoir modifier son format

Premier exemple de fichier XML (1/2)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
    <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>
</bookstore>
```

Premier exemple de fichier XML (2/2)



Syntaxe XML

- Tous les éléments xml doivent avoir une balise fermante
- Les balises sont sensibles à la casse
- Les éléments XML doivent être proprement imbriqués
- Un document XML doit contenir une seule racine
- Les valeurs des attributs doivent être écrits entre guillemets
- Commentaire : <!-- commentaire -->
- Les espaces ne sont pas tronqués en XML
- Caractères spéciaux : ”<” = < , ”>” = >, ”&”=&, ””””=", ”'”='

Eléments XML

- Un élément XML est composé de la balise ouvrante, de la balise fermante et de tout ce qui est compris entre ces deux balises.
- Il peut contenir des attributs et d'autres éléments XML
- Règles de nomination des balises :
  1. Les noms peuvent contenir des lettres, des chiffres et des lettres spéciaux (”-”, ”.” et ”:” sont déconseillés)
  2. Ils ne peuvent pas commencer par un chiffre ou un caractère de ponctuation ou ”XML” ou ”Xml”
  3. Ils ne peuvent pas contenir des espaces

Attributs

- Contiennent des informations supplémentaires sur les éléments Ex : <file type="gif">computer.gif</file>
- Les attributs ne doivent pas contenir des données car :
  - Les attributs ne peuvent pas contenir plusieurs valeurs
  - Les attributs ne peuvent pas contenir des données structurées en arbre
  - Les attributs ne sont pas extensibles
- Utiliser les éléments pour les données et les attributs pour les méta-données

XML DOM

- XML DOM est un standard pour accéder, ajouter, modifier ou supprimer des éléments XML
- Tous les éléments d'un document XML sont des noeuds :
  - Tout le document est un "document node"
  - Tout élément html est un "element node"
  - Le texte dans les éléments XML son des "text nodes"
  - Les attributs XML sont des "attribute nodes"

Parseur XML

- La majorité des navigateurs contiennent un parseur XML. Il permet de convertir le fichier XML en objet XML DOM accessible par JavaScript.
- XML DOM contient des méthodes pour parcourir l'arbre XML et manipuler et modifier les éléments XML
- Exemple de conversion du fichier books.xml en objet DOM:

```
xhttp=new XMLHttpRequest(); //créer la requête

xhttp.open("GET","books.xml",false);  //spécifier le fichier XML

xhttp.send();  //envoyer une requête pour récupérer le fichier XML

xmlDoc=xhttp.responseXML;  // affecter la réponse à un objet DOM
```

Propriétés

- Comme dans HTML DOM, les éléments XML DOM possèdent des propriétés auquel on peut accéder et modifier avec un langage de programmation comme JavaScript, exemples :
  - x.nodeName – le nom du noeud x
  - x.nodeValue – la valeur du noeud x
  - x.parentNode – le noeud père du noeud x
  - x.childNodes – les noeuds fils du noeud x
  - x.attributes – les attributs du noeud x

Méthodes

- Comme dans HTML DOM, tout élément XML DOM possède des méthodes qui permettent de le séléctionner, le modifier ou ajouter un nouveau élément, exemples :
  - x.getElementsByTagName(name) – retourne les éléments ayant le nom clé ”name”
  - x.appendChild(node) – ajouter un noeud fils x
  - x.removeChild(node) – supprimer un noeud fils x

Accéder aux noeuds

- Avec la méthode getElementsByTagName()

  Ex: listeTitre=xmlDoc.getElementsByTagName("title");

  text=listeTitre[0].childNodes[0].nodeValue;

- Parcourir l'arbre XML avec les propriétés

  Ex : x=xmlDoc.documentElement.childNodes;

  y=xmlDoc.documentElement.firstChild;

  for (i=0;i<x.length;i++)

  if (y.nodeType==1) {//Process only element nodes (type 1)

  document.write(y.nodeName + "
");

  y=y.nextSibling;}

Accéder aux attributs

Exemple:

```
xmlDoc=loadXMLDoc("books.xml");
//Récupérer la liste d'attribut du peremier livre
x=xmlDoc.getElementsByTagName("book")[0].attributes;
y=xmlDoc.getElementsByTagName("book")[0];
//afficher la valeur de l'attribut "category"
document.write(x.getNamedItem("category").nodeValue);
document.write(y.getAttributeNode("category").nodeValue);
document.write(y.getAttribute("category"));
//afficher la valeur du premier attribut
document.write(x[0].nodeValue);
```

Tronquer les espaces

- Certains navigateurs comme firefox traitent les espaces entre les éléments XML comme des éléments.

  Pour éliminer ce problème il faut tester le type de l'élément

  Ex : Pour rectifier la propriété nextSibling

```
function get_nextSibling(n){
        y=n.nextSibling;
        while (y.nodeType!=1)
                y=y.nextSibling;
      return y;
      }
```

Changer les valeurs

- Pour un noeud :

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

- Pour un attribut :

```
x=xmlDoc.getElementsByTagName(”book”);
x[0].setAttribute("category","food");
```

ou

```
x=xmlDoc.getElementsByTagName("book")[0]
y=x.getAttributeNode("category");
y.nodeValue="food";
```

Supprimer des noeuds

- Quand un noeud est supprimé tous ses fils sont supprimés
- Supprimer un noeud fils :

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

- Supprimer le noeud courant :

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

- Supprimer un noeud attribut

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

Créer des noeuds

- Créer un noeud élément :

```
xmlDoc=loadXMLDoc("books.xml");
newel=xmlDoc.createElement("edition");
```

- Créer un noeud attribut :

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

- Créer un noeud texte :

```
newtext=xmlDoc.createTextNode("first");
```

- Créer un commentaire :

```
newComment=xmlDoc.createComment("Revised March 2008");
```

Ajouter des noeuds au fichier XML

- Ajouter un noeud fils (élément, texte ou commentaire):

  x.appendChild(newElement);

- Ajouter un élément avant un autre élément :

  x=xmlDoc.documentElement;

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

  x.insertBefore(newNode,y);

- Ajouter un attribut :

  x[0].setAttribute("edition","first");

  x[0].setAttributeNode(newatt);

- Ajouter du texte à un texte Node : x.insertData(0,"Easy ");

Remplacer des noeuds dans le fichier XML

- Exemple :

```
x=xmlDoc.documentElement;
newNode=xmlDoc.createElement("book");
newTitle=xmlDoc.createElement("title");
newText=xmlDoc.createTextNode("A Notebook");
newTitle.appendChild(newText);
newNode.appendChild(newTitle);
y=xmlDoc.getElementsByTagName("book")[0]
x.replaceChild(newNode,y);
```

Copier un noeud

```
oldNode=xmlDoc.getElementsByTagName('book')[0];
newNode=oldNode.cloneNode(true);
xmlDoc.documentElement.appendChild(newNode);
y=xmlDoc.getElementsByTagName("title");
for (i=0;i<y.length;i++){
    document.write(y[i].childNodes[0].nodeValue);
    document.write("<br />");
}
```