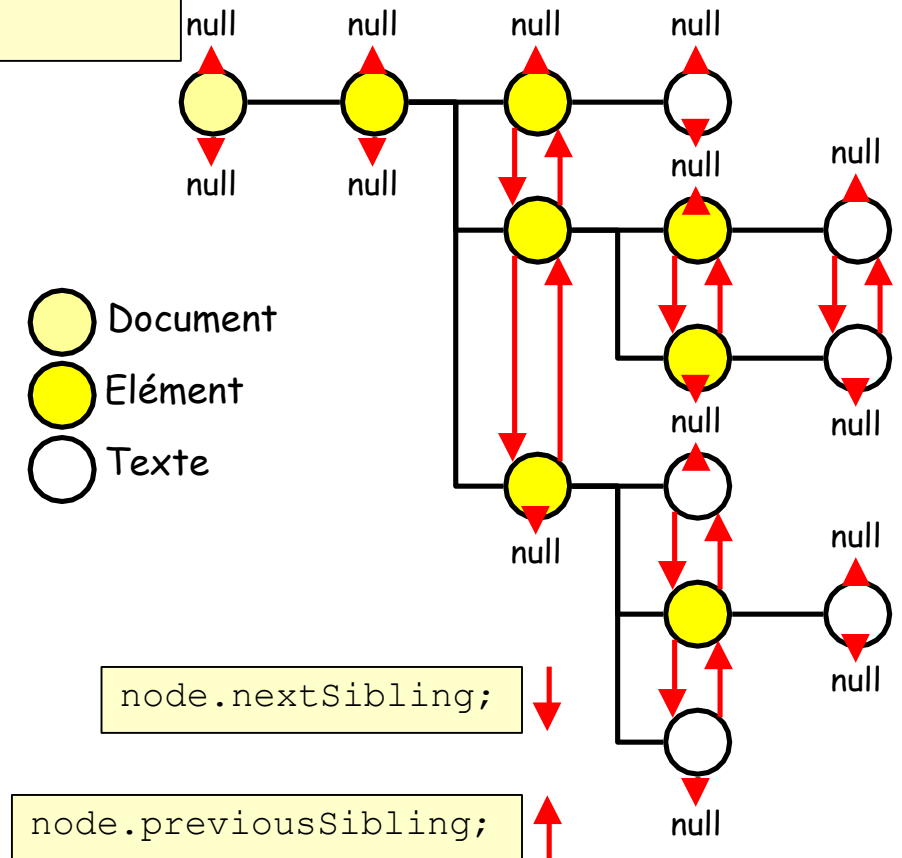
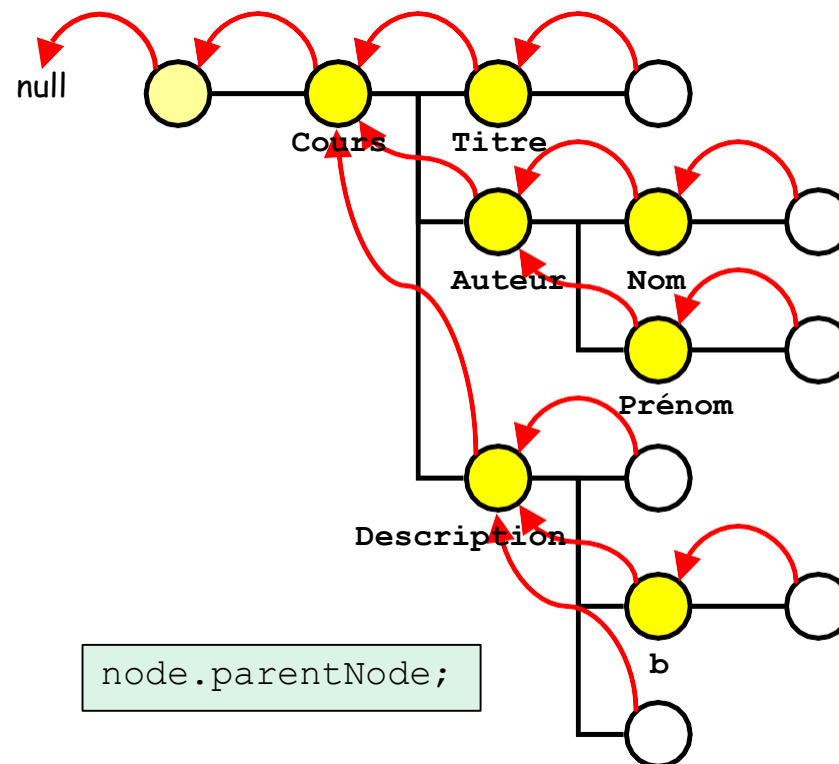


# DÉPLACEMENT DANS L'ARBRE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Cours>
  <Titre>Cours XML</Titre>
  <Auteur>
    <Nom>Poulard</Nom>
    <Prénom>Philippe</Prénom>
  </Auteur>
  <Description>
    Ce cours aborde les <b>concepts</b> de
    base mis en œuvre dans XML.
  </Description>
</Cours>
```

```
Node#parentNode;
Node#hasChildNodes();
Node#firstChild;
Node#lastChild;
Node#previousSibling;
Node#nextSibling;
```



# OBJETS DOM

## NodeList

Représente une collection de nœuds ordonnés.

Propriétés :

`length`

Méthodes :

`item`

# PARCOURS DE L'ARBRE

```
for (Node child = node.getFirstChild();
     child != null;
     child = child.getNextSibling()) {
    // do something
}
```

ou

```
NodeList nodeList = node.getChildNodes();
for (int i = 0;
     i < nodeList.getLength();
     i++) {
    Node child = nodeList.item(i);
    // do something
}
```

## Parcours récursif

```
public void processNodeRecursively(Node node) {
    // do something
    for (Node child = node.getFirstChild();
         child != null;
         child = child.getNextSibling()) {
        processNodeRecursively(child);
    }
}
```

**DOM level 2**

traversal

```
DocumentTraversal t;
NodeIterator ni;
Node child;
t = (DocumentTraversal) node.getOwnerDocument();
ni = t.createNodeIterator(node, NodeFilter.SHOW_ALL, null, false);
while ((child = ni.nextNode()) != null) {
    // do something
}
ni.detach();
```

# OBJETS DOM



## Document

Représente le document XML ou HTML.

Propriétés :

`doctype`, `documentElement`, `implementation`  
`attributes`, `childNodes`, `firstChild`, `lastChild`, `localName`, `namespaceURI`,  
`nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `ownerDocument`, `parentNode`, `prefix`,  
`previousSibling`

Méthodes :

`createAttribute`, `createAttributeNS`, `createCDATASection`, `createComment`,  
`createDocumentFragment`, `createElement`, `createElementNS`, `createEntityReference`,  
`createProcessingInstruction`, `createTextNode`, `getElementById`,  
`getElementsByName`, `getElementsByNameNS`, `importNode`  
`appendChild`, `cloneNode`, `hasAttributes`, `hasChildNodes`, `insertBefore`,  
`isSupported`, `normalize`, `removeChild`, `replaceChild`

# OBJETS DOM



## Element

Représente un nœud d'élément dans un document XML.

Propriétés :

**tagName**

`attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling`

Méthodes :

`getAttribute, getAttributeNS, getAttributeNode, getAttributeNodeNS, getElementsByTagName, getElementsByTagNameNS, hasAttribute, hasAttributeNS, removeAttribute, removeAttributeNS, removeAttributeNode, setAttribute, setAttributeNS, setAttributeNode, setAttributeNodeNS, appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild`

Un nœud créé fait partie du document  
mais n'est pas attaché à l'arbre.

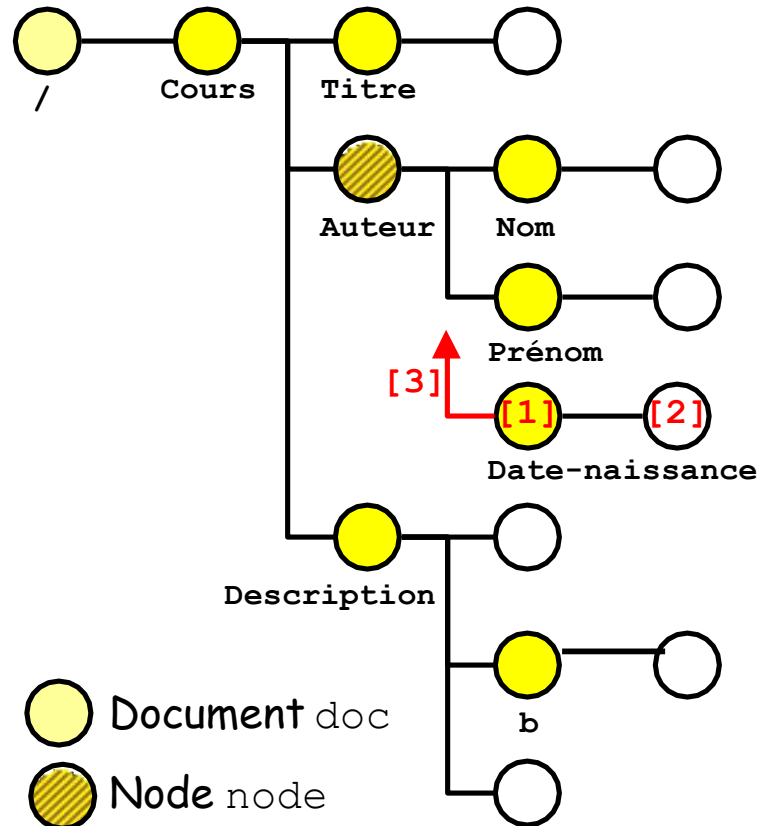
Les ajouts se font donc en 2 étapes :

- création
- ajout

```
Document#createXXX()  
Node#insertBefore(newChild, refChild)  
Node#appendChild(newChild)  
Node#replaceChild(newChild, refChild)  
Node#removeChild(oldChild)
```

## MODIFICATIONS DE L'ARBRE

```
[1] Element newChild = doc.createElement("Date-naissance");  
[2] newChild.appendChild(doc.createTextNode("10-06-1969"));  
[3] node.appendChild(newChild);
```



Les objets sont réactifs aux  
modifications de l'arbre qui peuvent avoir des  
effets de bord inattendus.

```
for (Node child = node.getFirstChild();  
     child != null;  
     child = child.getNextSibling()) {  
    node.removeChild(child);  
}
```

Après suppression du premier nœud de `node`,  
`child.getNextSibling()` retourne `null`,  
et les autres nœuds de `node` ne seront pas  
supprimés.

Pour supprimer tous les fils d'un nœud :

```
while (node.hasChildNodes()) {  
    node.removeChild(node.getFirstChild());  
}
```

# OBJETS DOM



## DocumentType

Représente la déclaration de type de document indiqué par la balise `<!DOCTYPE>`

### Propriétés :

**entities, internalSubset, name, notations, publicId, systemId**

attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling

### Méthodes :

appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild

# OBJETS DOM

## DocumentFragment

Représente une partie de l'arborescence d'un document. Ce fragment pouvant ne pas être bien formé, est utilisé généralement pour des opérations d'insertion.

### Propriétés :

`attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling`

### Méthodes :

`appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild`

C'est un "espace de travail" pratique...



# OBJETS DOM



**Attr**

Représente un attribut d'un objet Element.

Propriétés :

`name, ownerElement, specified, value`  
`attributes, childNodes, firstChild, lastChild, localName, namespaceURI,`  
`nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix,`  
`previousSibling`

Méthodes :

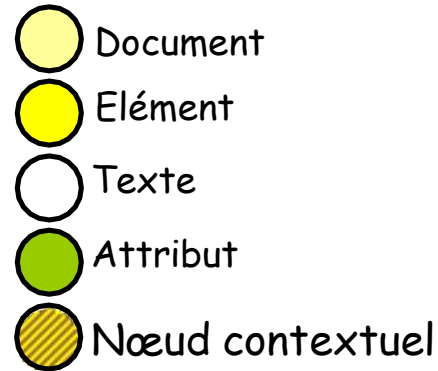
`appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported,`  
`normalize, removeChild, replaceChild`

Les attributs sont des `Nodes`, mais pas connectés à l'arbre par une dépendance hiérarchique.

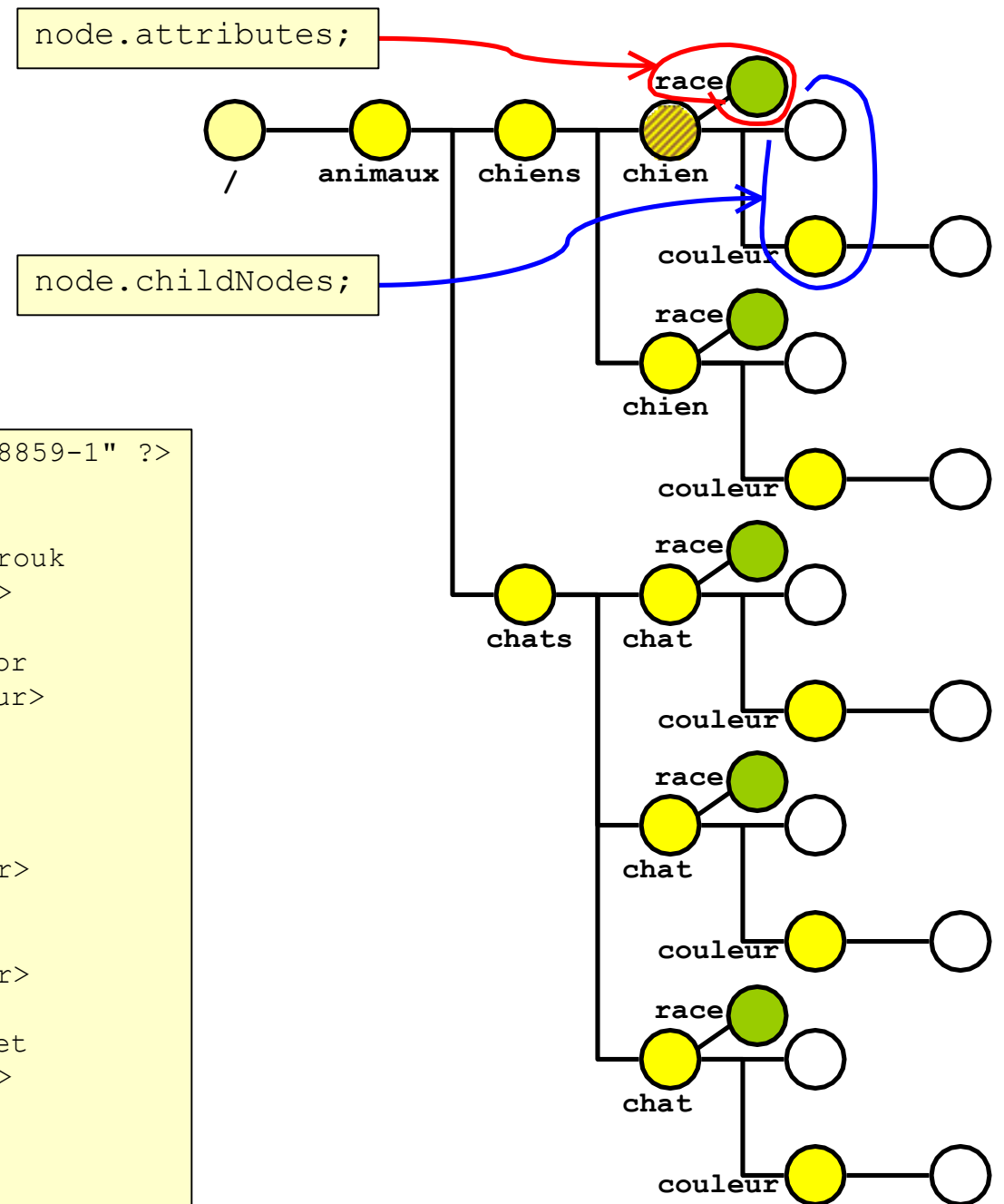


( Pour un `Node` élément, `Node#childNodes` ne retourne pas d'attributs, il faut utiliser `Node#attributes` )

# ATTRIBUTS DANS LE DOM



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<animaux>
  <chiens>
    <chien race="Labrador">Mabrouk
      <couleur>noir</couleur>
    </chien>
    <chien race="Labrador">Médor
      <couleur>marron</couleur>
    </chien>
  </chiens>
  <chats>
    <chat race="Siamois">Félix
      <couleur>crème</couleur>
    </chat>
    <chat race="Birman">Tom
      <couleur>crème</couleur>
    </chat>
    <chat race="Abyssin">Rominet
      <couleur>gris</couleur>
    </chat>
  </chats>
</animaux>
```



# ATTRIBUTS DANS LE DOM

```
NamedNodeMap Element#attributes
String Element#getAttribute(String name)
void Element#setAttribute(String name, String value)
void Element#removeAttribute(String name)
Attr Element#getAttributeNode(String name)
void Element#setAttributeNode(Attr attr)
void Element#removeAttributeNode(Attr attr)
```

```
NamedNodeMap map = elem.getAttributes();
for (int i = 0; i < map.getLength(); i++) {
    Attr attr = (Attr) map.item(i);
    // do something
}
```



Si l'attribut n'existe pas :

```
String value = elem.getAttribute("style");
```

Retourne ""

```
Attr attr = elem.getAttributeNode("style");
```

Retourne null

# OBJETS DOM

## **CDATASection**

Représente une section de données textuelles (Character Data Section).

### Propriétés :

`data, length, attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling`

### Méthodes :

`appendData, deleteData, insertData, replaceData, substringData, appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild, splitText`

# OBJETS DOM

## CharacterData

Est une extension de l'interface Node qui permet d'accéder aux données textuelles dans le modèle d'objet.

### Propriétés :

**data, length**

attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling

### Méthodes :

**appendData, deleteData, insertData, replaceData, substringData**

appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild

# OBJETS DOM



## Comment

Représente un commentaire dans un document XML ou HTML

```
<!-- Commentaire -->
```

### Propriétés :

`data, length, attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling`

### Méthodes :

`appendData, deleteData, insertData, replaceData, substringData, appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild`

# OBJETS DOM

## DOMImplementation

Fournit des méthodes qui sont indépendantes de n'importe quelles instances particulières du Modèle d'Objet du Document

Propriétés :

Aucune

Méthodes :

`createDocument, createDocumentType, hasFeature`

# OBJETS DOM

○ **Entity** Représente une entité analysée ou non-analysée dans un document XML.

Propriétés :

```
<!ENTITY entité SYSTEM "blah-blah">
```

**notationName, publicId, systemId**

attributes, childNodes, firstChild, lastChild, localName,  
namespaceURI, nextSibling, nodeName, nodeType, nodeValue,  
ownerDocument, parentNode, prefix, previousSibling

Méthodes :

appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore,  
isSupported, normalize, removeChild, replaceChild



# OBJETS DOM



## EntityReference

Contient le nom de l'entité

&entité;

### Propriétés :


`attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling`


### Méthodes :

`appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild`

# ENTITÉS DANS LE DOM

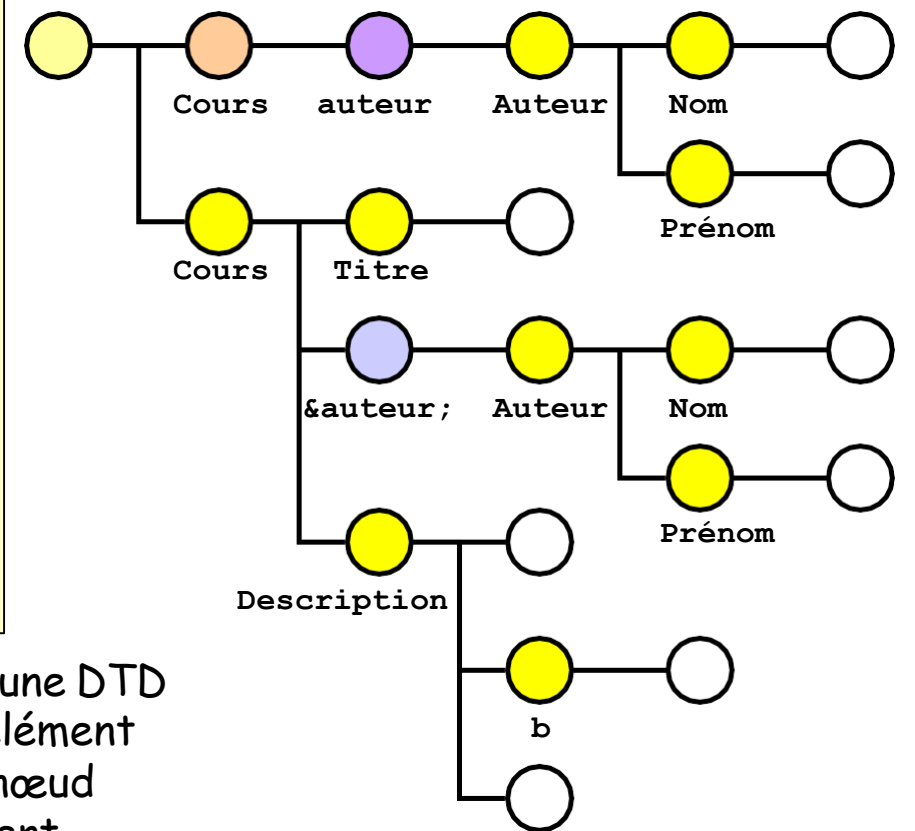
```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Cours [
  <!ENTITY auteur
    "<Auteur>
      <Nom>Poulard</Nom>
      <Prénom>Philippe</Prénom>
    </Auteur>">
]>
<Cours>
  <Titre>Cours XML</Titre>
  &auteur;
  <Description>
    Ce cours aborde les <b>concepts</b>
    de base mis en &#339;uvre dans XML.
  </Description>
</Cours>
```


 Ce document pourrait être conforme à une DTD dans laquelle l'élément `Auteur` suivrait l'élément `Titre`. Pourtant, dans le DOM, le premier nœud suivant (de type `ELEMENT_NODE`) de l'élément `Titre` n'est pas l'élément `Auteur`.

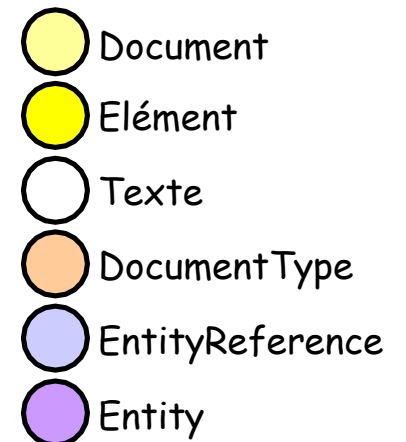
 Certains parseurs proposent des options pour intégrer directement les contenus d'entités sans les références

**DOM level 2**

Les vues (views) permettent d'obtenir une représentation du document sans les références d'entités.



 Le premier objet d'un document n'est pas nécessairement un élément



# OBJETS DOM

## NamedNodeMap

Représente des collections de nœuds qui peuvent être accédées par un nom.

Propriétés :

`length`

Méthodes :

`getNamedItem, getNamedItemNS, item, removeNamedItem,  
removeNamedItemNS, setNamedItem, setNamedItemNS`

# OBJETS DOM

## Notation

Représente une notation `<!NOTATION notation>` déclarée dans la DTD.

### Propriétés :

**publicId, systemId**

attributes, childNodes, firstChild, lastChild, localName, namespaceURI, nextSibling, nodeName, nodeType, nodeValue, ownerDocument, parentNode, prefix, previousSibling

### Méthodes :

appendChild, cloneNode, hasAttributes, hasChildNodes, insertBefore, isSupported, normalize, removeChild, replaceChild

# OBJETS DOM



## ProcessingInstruction

Représente une instruction de traitement

```
<?Nom_Instruction Cible?>
```

Propriétés :

**data, target**

attributes, childNodes, firstChild, lastChild, localName,  
namespaceURI, nextSibling, nodeName, nodeType, nodeValue,  
ownerDocument, parentNode, prefix, previousSibling

Méthodes :

appendChild, cloneNode, hasAttributes, hasChildNodes,  
insertBefore, isSupported, normalize, removeChild, replaceChild

# OBJETS DOM

○ **Text** Représente le contenu textuel d'un attribut ou d'un élément.

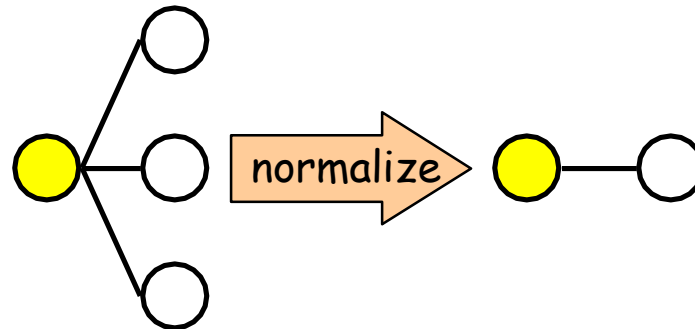
Propriétés :

`data`, `length`, `attributes`, `childNodes`, `firstChild`, `lastChild`, `localName`, `namespaceURI`, `nextSibling`, `nodeName`, `nodeType`, `nodeValue`, `ownerDocument`, `parentNode`, `prefix`, `previousSibling`

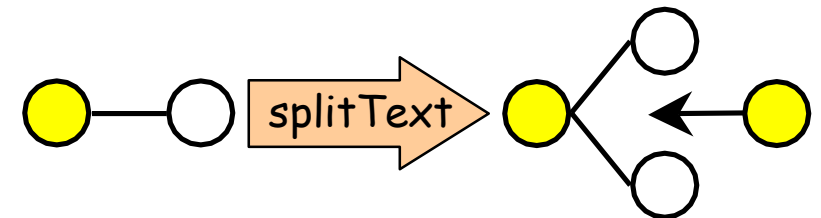
Méthodes :

**splitText**

`appendData`, `deleteData`, `insertData`, `replaceData`, `substringData`, `appendChild`, `cloneNode`, `hasAttributes`, `hasChildNodes`, `insertBefore`, `isSupported`, `normalize`, `removeChild`, `replaceChild`



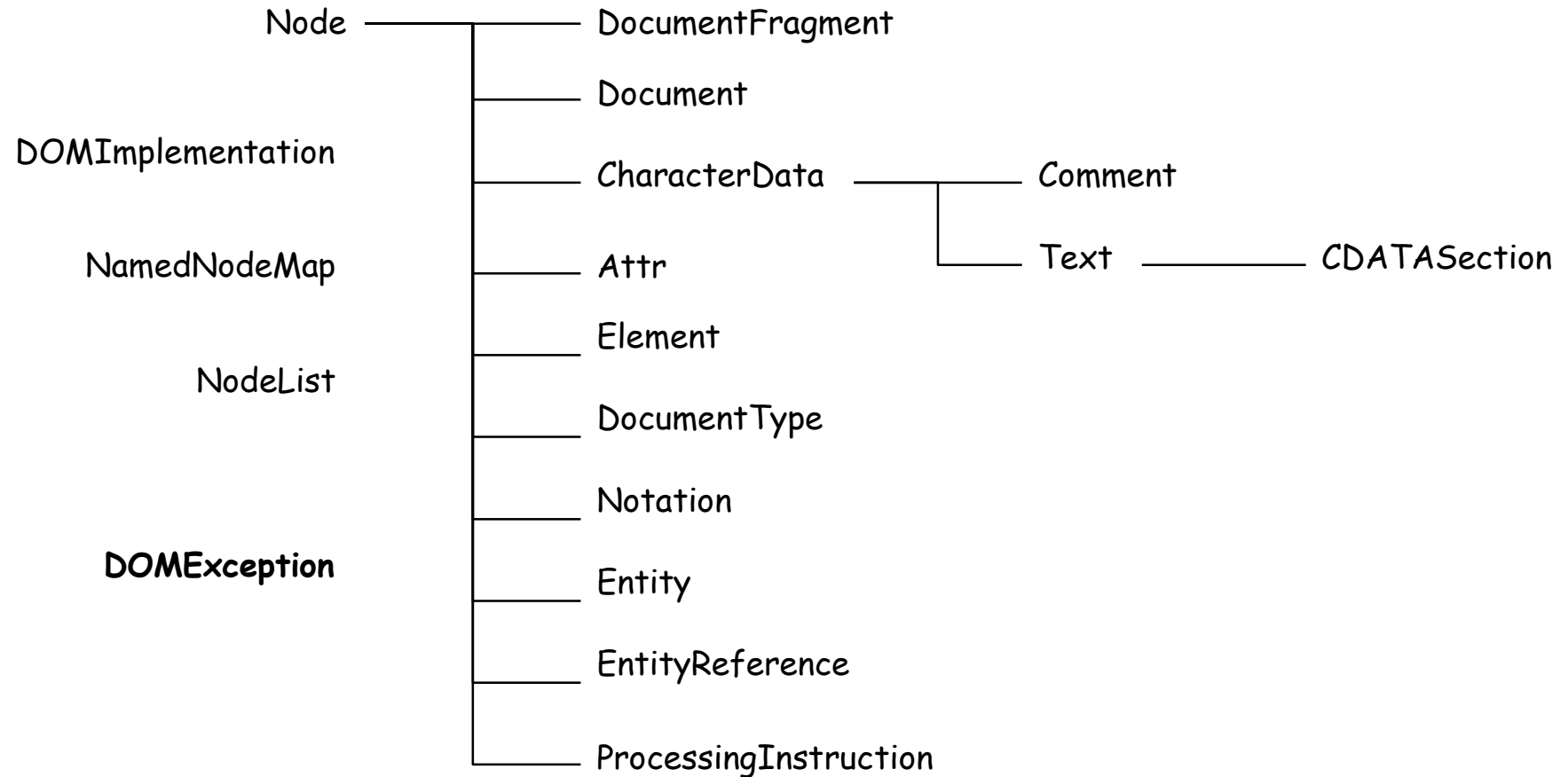
`Text.splitText(offset);`



Possibilité  
d'inclusion d'un  
élément au  
milieu du texte

```
try {  
    ...  
    root.appendChild(document.createTextNode("Un peu"));  
    root.appendChild(document.createTextNode(" "));  
    root.appendChild(document.createTextNode("de texte"));  
    document.getDocumentElement().normalize();  
} catch (ParserConfigurationException pce) {  
    ...  
}
```

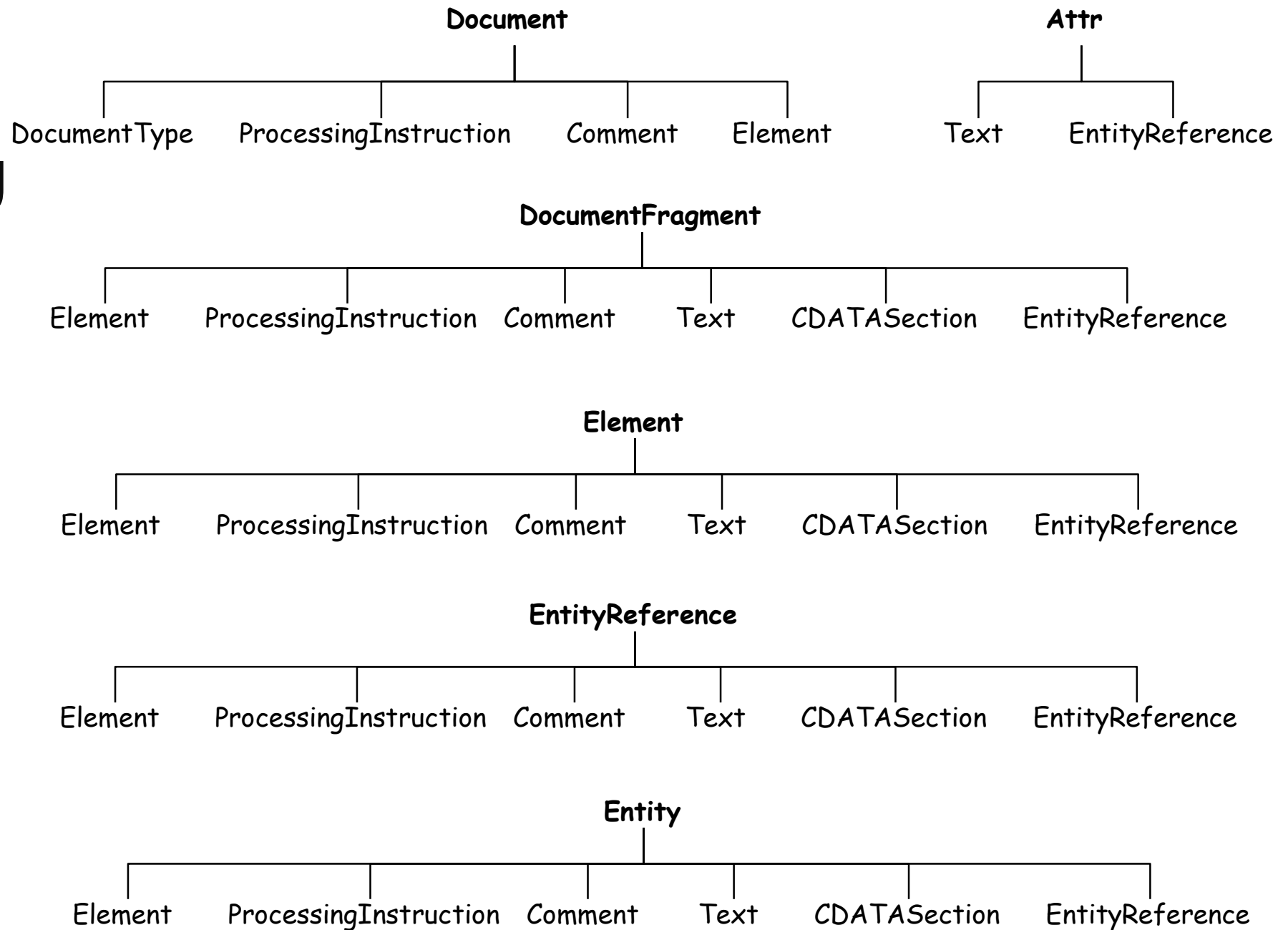
# HÉRITAGE DES OBJETS DOM



Pour obtenir les nœuds textuels, il faut penser à prendre en compte les sections CDATA

```
If (node.getNodeType() == Node.TEXT_NODE ||  
    node.getNodeType() == Node.CDATA_SECTION_NODE) {  
    // traitement des nœuds texte  
}
```

# HIÉRARCHIE DES NOEUD DU DOM

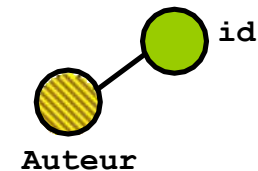




# CLONAGE DE NŒUDS DANS LE DOM

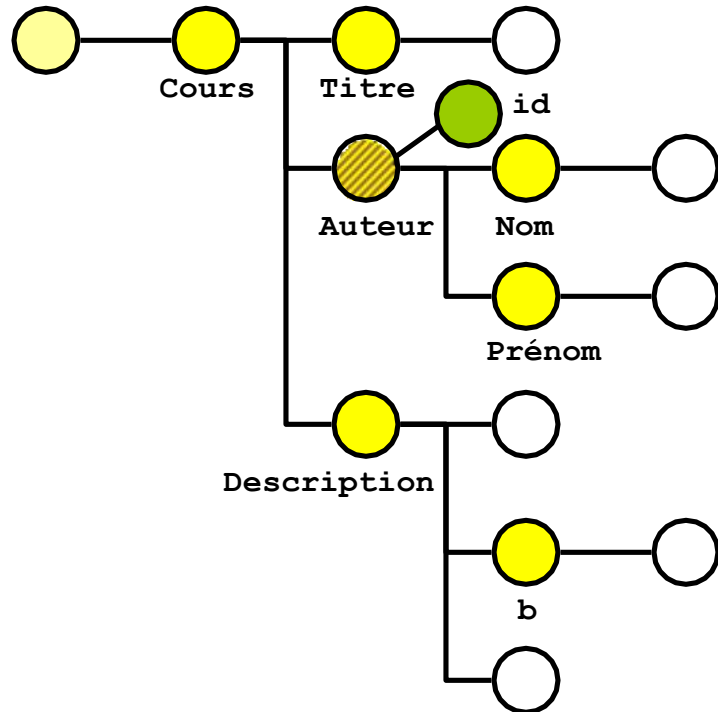
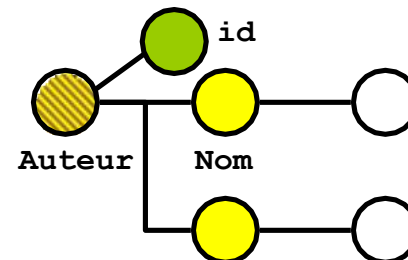
## Clonage en surface

```
Element newElement = elem.cloneNode(false);
```



## Clonage en profondeur

```
Element newElement = elem.cloneNode(true);
```



● Élément `elem`  
● Élément `newElement`

# ECHANGE DE NŒUDS ENTRE DOCUMENTS

Document    Document#ownerDocument  
Node        Document#importNode()

Les nœuds, même non attachés à l'arbre, appartiennent au document.

```
DOMParser parser = new DOMParser();
parser.parse(.../...);
Document doc1 = parser.getDocument();
parser.parse(.../...);
Document doc2 = parser.getDocument();
.../...
Element el1 = doc1.getDocumentElement();
Element el2 = doc2.getDocumentElement();
el1.appendChild(el2);
```



Provoque une exception  
DOMException

```
DOMParser parser = new DOMParser();
parser.parse(.../...);
Document doc1 = parser.getDocument();
parser.parse(.../...);
Document doc2 = parser.getDocument();
.../...
Element el1 = doc1.getDocumentElement();
Element el2 = doc2.getDocumentElement();
Node node = doc1.importNode(el2, true);
el1.appendChild(node);
```



Importation correcte d'un  
nœud

# ESPACES DE NOMMAGE DANS LE DOM

## DOM level 2

Propriétés des Element et Attr : namespaceURI  
localName  
prefix

Vérifier que votre parser prend en compte les espaces de nommage.  
Le cas échéant, activer la fonctionnalité.



DOM ne réalise aucune déclaration d'espace de nommage automatiquement.

```
Element elem = doc.createElementNS(  
    "http://www.foo.com",  
    "foo:bar");
```

← URI de l'espace de nom

← Nom qualifié de l'élément

Crée un élément `bar` dont l'URI de l'espace de nom est `http://www.foo.com` et dont le préfixe est `foo`.

La déclaration de l'espace de nommage `xmlns:foo="http://www.foo.com"` n'est pas créée automatiquement.

Habituellement, on n'associe pas d'espaces de nommage aux attributs, puisqu'ils dépendent de leur élément parent. Lorsque ce n'est pas le cas, on peut les déclarer ainsi :

```
elem.setAttributeNS(  
    "http://www.w3.org/1999/xlink",  
    "xlink:type",  
    "simple");  
elem.setAttributeNS(  
    "http://www.w3.org/1999/xlink",  
    "xlink:href",  
    "myDocument.html");
```

← URI de l'espace de nom

← Attribut

← Valeur

← URI de l'espace de nom

← Attribut

← Valeur

# DÉCLARATION DES ESPACES DE NOMMAGE

```
elem.setAttributeNS(  
    "http://www.w3.org/2000/xmlns/",  
    "xmlns:foo",  
    "http://www.foo.com");
```

← URI de l'espace de nom xmlns  
← Attribut  
← Valeur

```
elem.setAttributeNS(  
    "http://www.w3.org/2000/xmlns/",  
    "xmlns:xlink",  
    "http://www.w3.org/1999/xlink");
```

← URI de l'espace de nom xmlns  
← Attribut  
← Valeur

D'une manière générale :

```
ancestorElem.setAttributeNS(  
    "http://www.w3.org/2000/xmlns/",  
    "xmlns:" + elem.getPrefix(),  
    elem.getNamespaceURI());
```

← URI de l'espace de nom xmlns  
← Attribut  
← Valeur

Les déclarations doivent se faire sur l'élément ou l'un de ses ancêtres.



Les déclarations d'espace de nommage (ou l'absence de déclaration) peuvent rendre le DOM inconsistant. Il appartient au programmeur de réaliser les déclarations idoines.

```
Element elem = doc.createElementNS(  
    "http://www.foo.com", "foo:bar");  
elem.setAttributeNS(  
    "http://www.w3.org/2000/xmlns/", "xmlns:foo", "http://www.blah.com");
```

Non

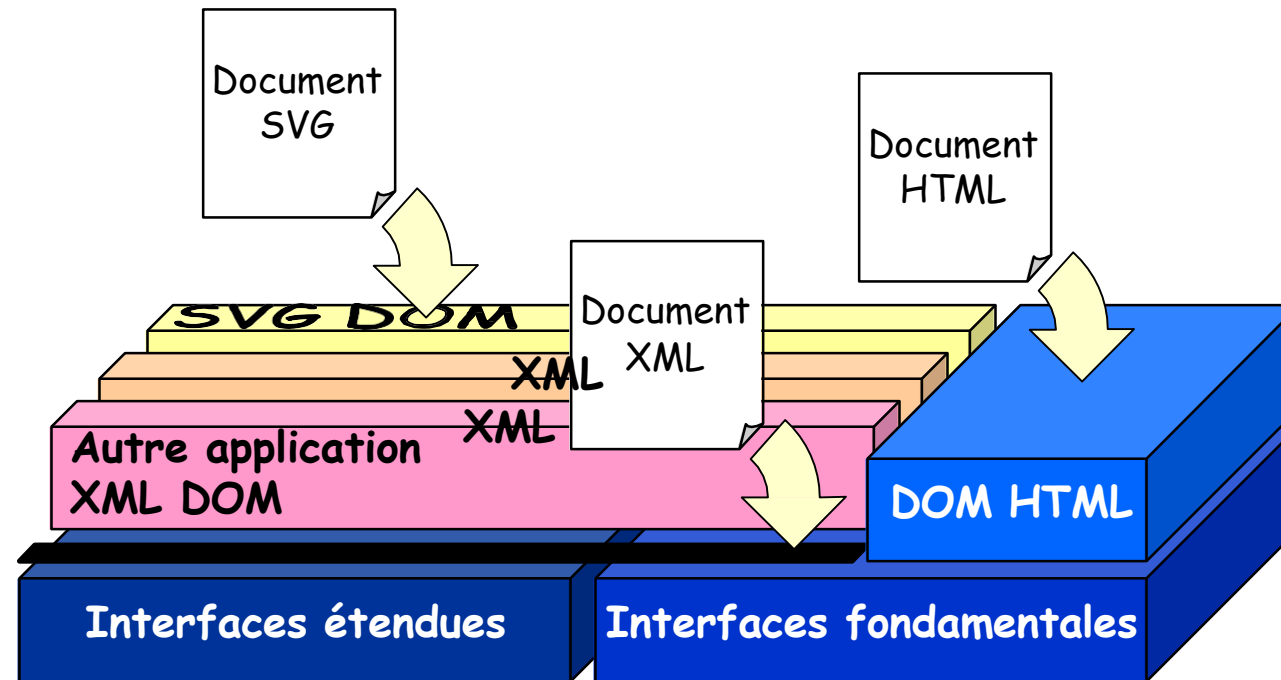
# ESPACES DE NOMMAGE DANS LE DOM LEVEL 3

Le DOM level 3 permet de lire et écrire des documents XML

La méthode `write` s'assure de l'intégrité des déclarations d'espace de nommage avec le contenu

# SPÉCIALISATION DES APIS DOM

DOM Core  
+  
spécialisation  
par domaine

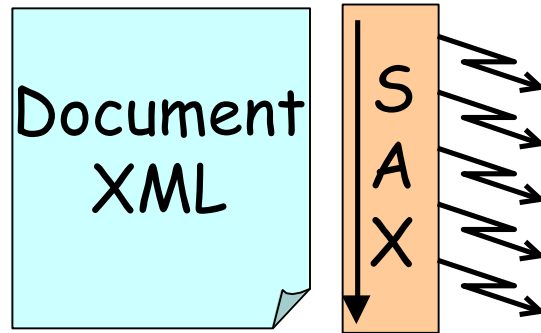


Exemple : SVG DOM

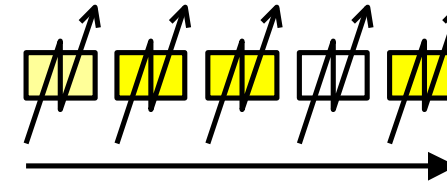
SVG DOM	→	<code>Point.setX(12);</code>
DOM Core	→	<code>Element.setAttribute("x", "12");</code>

# SAX

API simple pour XML :  
une API pour **réagir** sur  
le contenu et la structure d'un document



Le document est vu comme une suite  
d'événements qui surviennent lors de la  
lecture séquentielle du document

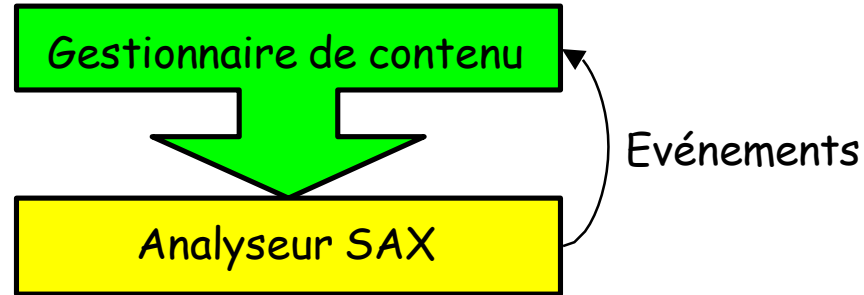


Support des espaces de nommage, support des filtres SAX, nouvelles interfaces, changement du nom de certaines classes et interfaces, mécanisme générique pour lire ou modifier les propriétés et fonctionnalités du parser

## Simple API for XML

SAX, standard issu de xmldev mailing list  
implémenté par David Megginson

# FONCTIONNEMENT DE SAX



A chaque événement, un callback est appelé dans l'ordre d'apparition dans le document

- Début de document
- Début d'élément
- Caractères
- ...
- Fin de document

Le gestionnaire de contenu est une classe qui effectue les traitements appropriés à chaque appel d'un callback

Cette classe doit être enregistrée auprès de l'analyseur SAX



# PRINCIPE DE SAX

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Cours>
  <Titre>Cours XML</Titre>
  <Auteur>
    <Nom>Poulard</Nom>
    <Prénom>Philippe</Prénom>
  </Auteur>
  <Description>
    Ce cours aborde les <b>concepts</b> de
    base mis en œuvre dans XML.
  </Description>
</Cours>
```

```
startDocument
startElement: Cours
startElement: Titre
characters: "Cours XML"
endElement: Titre
startElement: Auteur
startElement: Nom
characters: "Poulard"
endElement: Nom
startElement: Prénom
characters: "Philippe"
endElement: Prénom
endElement: Auteur
startElement: Description
characters: "Ce cours aborde les "
startElement: b
characters: "concepts"
endElement: b
characters: " de base mis en œuvre dans XML."
endElement: Description
endElement: Cours
endDocument
```

# PRINCIPAUX HANDLERS

Les applications doivent enregistrer le handler d'événements auprès d'une instance d'un parser qui implémente l'interface `XMLReader`.

En Java :

## Interfaces

Parser : `org.xml.sax.XMLReader`

Handlers : `org.xml.sax.ContentHandler`

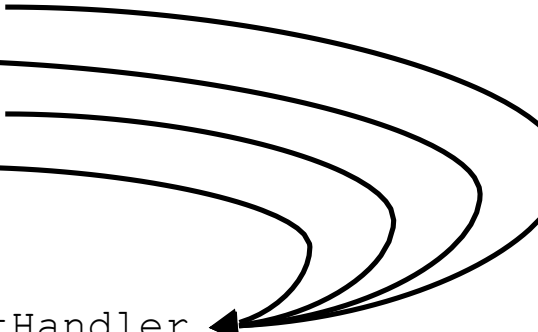
`org.xml.sax.DTDHandler`

`org.xml.sax.EntityResolver`

`org.xml.sax.ErrorHandler`

Implémentation par défaut (qui ne fait rien) :

`org.xml.sax.helpers.DefaultHandler`



Avec MSXML2.4.0 :

## Interfaces

Parser : `ISAXXMLReader`

Handlers : `ISAXContentHandler`

`ISAXDTDHandler`

`ISAXEntityResolver`

`ISAXErrorHandler`

# L'INTERFACE CONTENT HANDLER

<code>startDocument()</code>	Notification du début du document
<code>endDocument()</code>	Notification de la fin du document
<code>startElement(String namespaceURI, String localName, String qName, Attributes atts)</code>	Notification du début d'un élément
<code>endElement(String namespaceURI, String localName, String qName)</code>	Notification de la fin d'un élément
<code>characters(char[] ch, int start, int length)</code>	Notification de données caractères
<code>ignorableWhitespace(char[] ch, int start, int length)</code>	Notification d'espaces blancs ignorables dans le contenu d'un élément
<code>startPrefixMapping(String prefix, String uri)</code>	Notification du début de la portée d'un espace de nommage
<code>endPrefixMapping(String prefix)</code>	Notification de la fin de la portée d'un espace de nommage
<code>processingInstruction(String target, String data)</code>	Notification d'une instruction de traitement
<code>skippedEntity(String name)</code>	Notification d'une entité non résolue
<code>setDocumentLocator(Locator locator)</code>	Permet d'enregistrer un <code>Locator</code> qui délivre des informations sur la localisation d'un événement SAX (numéro de ligne, colonne...)

# ENREGISTREMENT D'UN HANDLER

```
String fileName = "...";
ContentHandler myContentHandler = new ...;
.../...
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(myContentHandler);
parser.parse(fileName);
```

```
String fileName = "...";
ContentHandler myContentHandler = new ...;
DTDHandler myDTDHandler = new ...;
ErrorHandler myErrorHandler = new ...;
EntityResolver myEntityResolver = new ...;
.../...
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(myContentHandler);
parser.setDTDHandler(myDTDHandler);
parser.setErrorHandler(myErrorHandler);
parser.setEntityResolver(myEntityResolver);
parser.parse(fileName);
```

```
String fileName = "...";
DefaultHandler handler = new ...;
.../...
XMLReader parser = XMLReaderFactory.createXMLReader();
parser.setContentHandler(handler);
parser.setDTDHandler(handler);
parser.setErrorHandler(handler);
parser.setEntityResolver(handler);
parser.parse(fileName);
```

# EXEMPLE

Liste des éléments rencontrés :

```
import java.io.File;
import java.io.IOException;
import org.xml.sax.XMLReader;
import org.xml.sax.SAXException;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;

public class MySAXParser extends DefaultHandler {
    public MySAXParser() {}
    static public void main(String[] args) {
        try {
            DefaultHandler handler = new MySAXParser();
            XMLReader parser = XMLReaderFactory.createXMLReader();
            parser.setContentHandler(handler);
            parser.parse(args[0]);
        } catch (SAXException e) {e.printStackTrace();}
        } catch (IOException e) {e.printStackTrace();}
    }

    public void startElement(String uri, String localName, String qName, Attributes atts) {
        System.out.println("Elément rencontré : " + qName);
    }
}
```

Invite de commandes

```
C:\>java MySAXParser cours.xml
Elément rencontré : Cours
Elément rencontré : Titre
Elément rencontré : Auteur
Elément rencontré : Nom
Elément rencontré : Prénom
Elément rencontré : Description
Elément rencontré : b
C:\>
```

# EVÉNEMENTS CHARACTERS()

La spécification SAX permet aux parsers de décomposer un segment de texte en plusieurs événements caractères de l'interface `ContentHandler`

```
<foo>
  Les APIs
  DOM &#x26; SAX
</foo>
```

```
startDocument
startElement : foo
characters : "\n  Les APIs\n  DOM & SAX\n"
endElement : foo
endDocument
```

## Parser SAX de Xerces

```
startDocument
startElement : foo
characters : "\n  Les APIs\n  DOM "
characters : "&"
characters : " SAX\n"
endElement : foo
endDocument
```

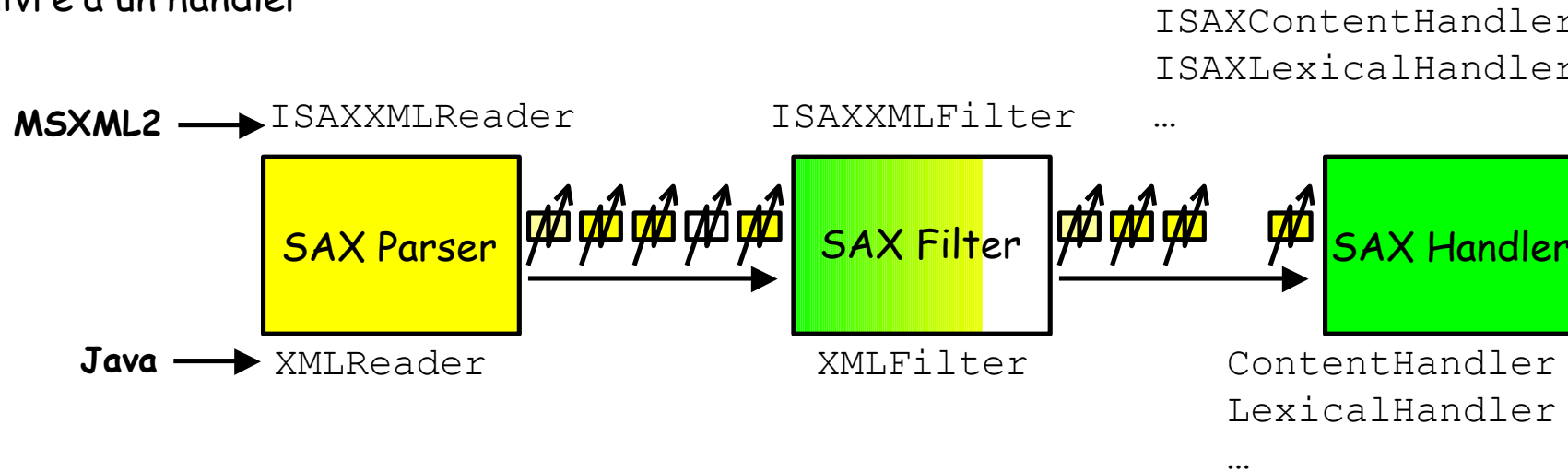
## Parser SAX de Crimson

```
startDocument
startElement : foo
characters : ""
characters : "\n"
characters : "  Les APIs"
characters : "\n"
characters : "  DOM "
characters : "&"
characters : " SAX"
characters : "\n"
endElement : foo
endDocument
```

⇒ Les programmes qui traitent les caractères doivent utiliser des tampons (buffers)

# FILTRES SAX

Un filtre SAX reçoit des événements SAX d'un parser SAX, peut les modifier, et les fait suivre à un handler

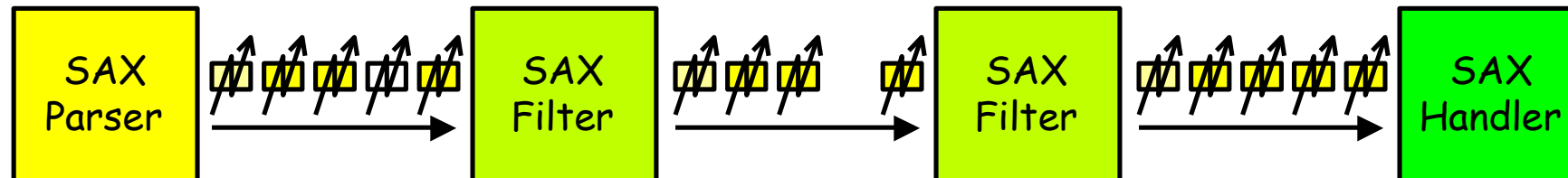


Le filtre SAX est vu par le parser SAX comme un handler SAX

Le filtre SAX est vu par le handler SAX comme un parser SAX

`XMLFilter` est dérivé de `XMLReader`

## Pipeline SAX



# UTILITÉ DES FILTRES SAX

Permet à un handler d'accepter différentes variantes de structures

```
<livre>
  <titre>Les misérables</titre>
  .../...
</livre>
```

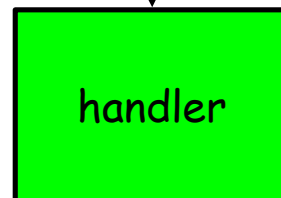
startElement : livre  
startElement : titre  
characters : "Les misérables"  
endElement : titre  
...  
endElement : livre

```
<livre titre="Les misérables">
  .../...
</livre>
```

startElement : livre  
...  
endElement : livre



startElement : livre  
startElement : titre  
characters : "Les misérables"  
endElement : titre  
...  
endElement : livre



à la réception de l'élément `livre`, le filtre vérifie s'il existe l'attribut `titre`.  
Si c'est le cas, le filtre génère les événements :  
startElement : titre  
characters : "Les misérables"  
endElement : titre



# UTILISATION DES FILTRES SAX

```
XMLReader parser = XMLReaderFactory.createXMLReader();
XMLFilter filter = new MyFilter();
filter.setParent(parser);
filter.setContentHandler(handler);
filter.parse(.../...);
```

ou

```
XMLReader parser = XMLReaderFactory.createXMLReader();
XMLFilter filter = new MyFilter(parser);
filter.setContentHandler(handler);
filter.parse(.../...);
```

## Pipeline SAX

```
XMLReader parser = XMLReaderFactory.createXMLReader();
XMLFilter myFilter = new MyFilter();
XMLFilter yourFilter = new YourFilter();
myFilter.setParent(parser);
yourFilter.setParent(myFilter);
yourFilter.setContentHandler(handler);
yourFilter.parse(.../...);
```

ou

```
XMLReader parser = XMLReaderFactory.createXMLReader();
XMLFilter yourFilter = new YourFilter(new MyFilter(parser));
yourFilter.setContentHandler(handler);
yourFilter.parse(.../...);
```

# ANALYSEURS SAX VALIDANTS

Un analyseur validant qui ne valide pas  
**n'est pas**  
un analyseur non validant

Traitement des espaces blancs

Certains analyseurs disposent de 2 implémentations :

- analyseur validant → `ignorableWhitespace()`
- analyseur non validant → `ignorableWhitespace()` ou `characters()`

Espaces blancs ignorables

```
<!ELEMENT dauphin (nom, sexe)>
```

```
<dauphin> ↵  
  <nom>Flipper</nom> ↵  
  <sexe>M</sexe> ↵  
</dauphin>
```

Truc : éviter d'avoir des espaces blancs significatifs

Si on ne peut pas faire autrement : les inclure dans une section CDATA

Pour les entités non résolues, il existe un callback spécifique :

```
public void skippedEntity(String name) throws SAXException {}
```

Mais les analyseurs non validants peuvent résoudre les appels d'entités

# ESPACES DE NOMMAGE DANS SAX

La spécification SAX2 définit les 2 fonctionnalités standards suivantes :

Introduits dans

**SAX2**

Fonction	Nom	Valeur par défaut
Namespaces	<a href="http://xml.org/sax/features/namespaces">http://xml.org/sax/features/namespaces</a>	true
Namespace-prefix	<a href="http://xml.org/sax/features/namespace-prefixes">http://xml.org/sax/features/namespace-prefixes</a>	false

Le parser peut fournir des informations sur les URI d'espace de nommage et les local names via `startElement()` et `endElement()` de `ContentHandler` et `getURI()` et `getLocalName()` de `Attributes`.

Disponibilité des noms qualifiés, selon l'implémentation

`startPrefixMapping()` et `endPrefixMapping()` de `ContentHandler` sont appelés quand les éléments qui déclarent des espaces de nommage sont rencontrés et quittés.

Il n'y a pas de déclaration d'espace de nommage (`xmlns...`) dans les instances d'`Attributes`.

Namespace -prefix  
Namespaces

false	true	x	-	x	-	Par défaut
true	true	x	x	x	x	
false	false	-	-	-	x	
true	false	-	x	-	x	

# COMPARAISON SAX ET DOM

## DOM

Un processus qui utilise le DOM ne peut traiter l'arbre qu'après la lecture entière du document

Accès aléatoire

Utilise beaucoup de mémoire

Construction de l'arbre du document  
Les objets sont réutilisables  
Richesse des fonctionnalités

Programmation aisée

Les implémentations de DOM reposent souvent sur un analyseur SAX

## SAX

Un analyseur SAX délivre les données à un processus au fur et à mesure de la lecture du document

Accès séquentiel

Utilise peu de mémoire pour son propre fonctionnement, et le strict nécessaire pour le processus traitant

Événementiel

Les objets ne sont pas stockés  
Fonctionnalités rudimentaires

Beaucoup de code à produire

# QUAND UTILISER SAX OU DOM ?

## DOM

Toutes les données doivent être utilisées

Les performances peuvent être limitées

Abondance de mémoire

## SAX

Seule une partie des données doivent être utilisées

Les performances sont critiques

Peu de mémoire



Les parsers DOM récents ont des temps de réponse à peine plus faibles que ceux de SAX : ils diffèrent la création des objets qui ne sont pas sollicités.

# AUTRES APIS

DOM pour Java :

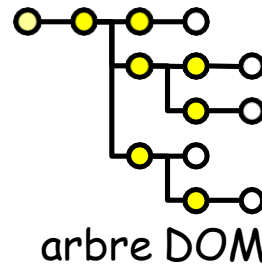
- JDOM
- DOM4J

<http://dom4j.org/>  
<http://www.jdom.org/>

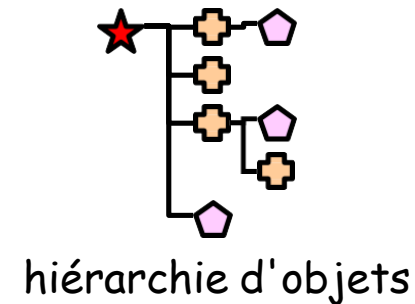
## XML Data Binding

Au lieu d'obtenir un modèle de données XML, on cherche à instancier des classes qui correspondent aux éléments rencontrés

### Modèle généraliste



### Modèle spécifique



Des techniques permettent de réaliser la désérialisation (unmarshal) automatiquement

Avec Java :

- JAXB
- Castor

**Merci pour votre attention !!**