



XML

XML

+ Exemple KML

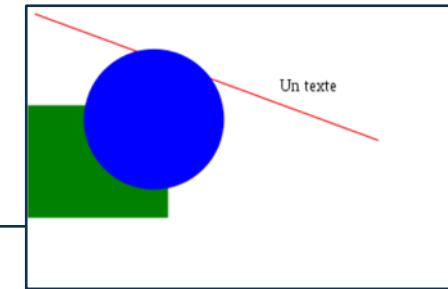


https://developers.google.com/kml/documentation/kml_tut?hl=FR

XML

+ Exemple : SVG Scalable Vector Graphics

```
<?xml version="1.0" encoding="utf-8"?>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="300" height="200">
  <title>Exemple simple de figure SVG</title>
  <desc>
    Cette figure est constituée d'un rectangle, d'un segment de droite et d'un cercle.
  </desc>
  <rect width="100" height="80" x="0" y="70" fill="green" />
  <line x1="5" y1="5" x2="250" y2="95" stroke="red" />
  <circle cx="90" cy="80" r="50" fill="blue" />
  <text x="180" y="60">Un texte</text>
</svg>
```



http://fr.wikipedia.org/wiki/Scalable_Vector_Graphics

XML

+ Exemple : x3D



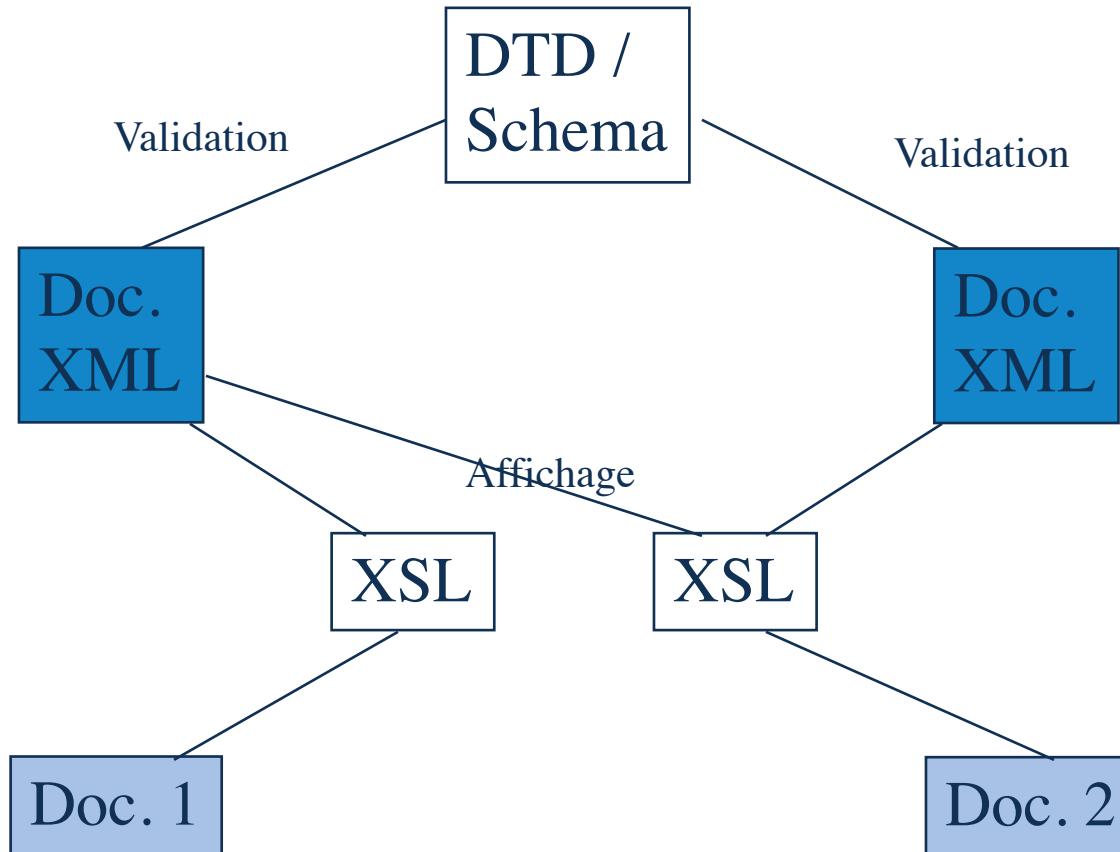
<http://www.web3d.org/case-studies/schalke-04-stadium-marketing/sports-stadium-marketing-platform>

XML

- + eXtended Markup Language
 - + Généralisation de HTML
 - + Règles de fonctionnement strictes
 - + Structures
 - + Utilisation de balises imbriquées
 - + Attributs
 - + Les noms de balises ne sont pas limités à ceux de HTML
Possibilité de définir des règles via des DTD ou des schémas
 - + Utilisation d'outils de manipulation
 - + Fichiers XSL + XSLT
 - + API relativement complète pour de nombreux langages

XML

+ Pourquoi ce mécanisme ?



XML

```
<?xml version="1.0" encoding="UTF-8"?>
<biblio sujet="XML">
  <livre isbn="9782212090819" lang="fr" sujet="applications">
    <auteur>
      <prenom>Jean-Christophe</prenom>
      <nom>Bernadac</nom>
    </auteur>
    <auteur>
      <prenom>François</prenom>
      <nom>Knab</nom>
    </auteur>
    <titre>Construire une application XML</titre>
    <editeur>
      <nom>Eyrolles</nom>
      <place>Paris</place>
    </editeur>
    <datepub>1999</datepub>
  </livre >
  <livre isbn="9782212090529" lang="fr" sujet="général">
    <auteur>
      <prenom>Alain</prenom>
      <nom>Michard</nom>
    </auteur>
    <titre>XML, Langage et Applications</titre>
    <editeur>
      <nom>Eyrolles</nom>
      <place>Paris</place>
    </editeur>
    <datepub>1998</datepub>
  </livre >
</biblio>
```

```
<?xml version="1.0" encoding= »ISO-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>Document HTML</title>
  </head>

  <body>
    <p>Document HTML</p>
  </body>
</html>
```

XML

- + Documents bien formés
 - + 1 balise racine
 - + Toute balise ouverte doit être fermée, sauf si c'est une balise vide.
 - + Les balises sont imbriquées de manière à former un arbre
 - + Les balises peuvent comporter des attributs.
 - + Les caractères < > & " ' sont traduits en leur équivalent : > ; < ; & ; " ; '
 - + Les séquences <[et]]> ne doivent pas être utilisées

XML

- + Notion de DTD et de Schéma
 - + Définir les balises autorisées, les attributs autorisés, leur imbrication.
 - + Définir un fichier décrivant ce qui est autorisé
 - + DTD : issus de la syntaxe SGML
fichier avec l'extension .dtd
 - + Schéma : description à l'aide de balises
fichier avec l'extension .xsd
 - + Associer la DTD ou le schéma au document XML

Si le fichier XML respecte la syntaxe, il est alors valide.

XML

+ Document Type Definition

```
<?xml encoding="UTF-8"?>
<!ELEMENT biblio (livre)+>
<!ATTLIST biblio
  sujet NMTOKEN #REQUIRED>

<!ELEMENT livre (auteur+,titre,editeur,datepub)>
<!ATTLIST livre
  isbn CDATA #REQUIRED
  lang NMTOKEN #REQUIRED
  sujet NMTOKEN #REQUIRED>

<!ELEMENT auteur (prenom,nom)>
...>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE biblio SYSTEM "biblio.dtd">
<biblio sujet="XML">
  <livre isbn="9782212090819" lang="fr"
    sujet="applications">
    <auteur>
      <prenom>Jean-Christophe</prenom>
      <nom>Bernadac</nom>
    </auteur>
    <titre>Construire une application XML</titre>
    <editeur>
      <nom>Eyrolles</nom>
      <place>Paris</place>
    </editeur>
    <datepub>1999</datepub>
  </livre>
</biblio>
```

XML

+ Schéma – Document Content Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xsschema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xselement name="biblio">
<xsccomplexType>
<xsssequence>
<xselement maxOccurs="unbounded" ref="livre"/>
</xsssequence>
<xssattribute name="sujet" use="required" type="xs:NCName"/>
</xsccomplexType>
</xselement>
<xselement name="livre">
<xsccomplexType>
<xsssequence>
<xselement maxOccurs="unbounded" ref="auteur"/>
<xselement ref="titre"/>
<xselement ref="editeur"/>
<xselement ref="datepub"/>
</xsssequence>
<xssattribute name="isbn" use="required" type="xs:integer"/>
<xssattribute name="lang" use="required" type="xs:NCName"/>
<xssattribute name="sujet" use="required" type="xs:NCName"/>
</xsccomplexType>
</xselement>
...
...
```

```
<?xml version="1.0" encoding="UTF-8"?>
<biblio
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="biblio.xsd">
    sujet="XML">
    ...

```

XML

- + eXtensible Stylesheet Language
 - + Objectif : transformer un document XML en autre chose
 - + Mécanisme : règles de transformation pour les balises du document XML
 - + Outil : XSLT = eXtensible Stylesheet Language Transformations

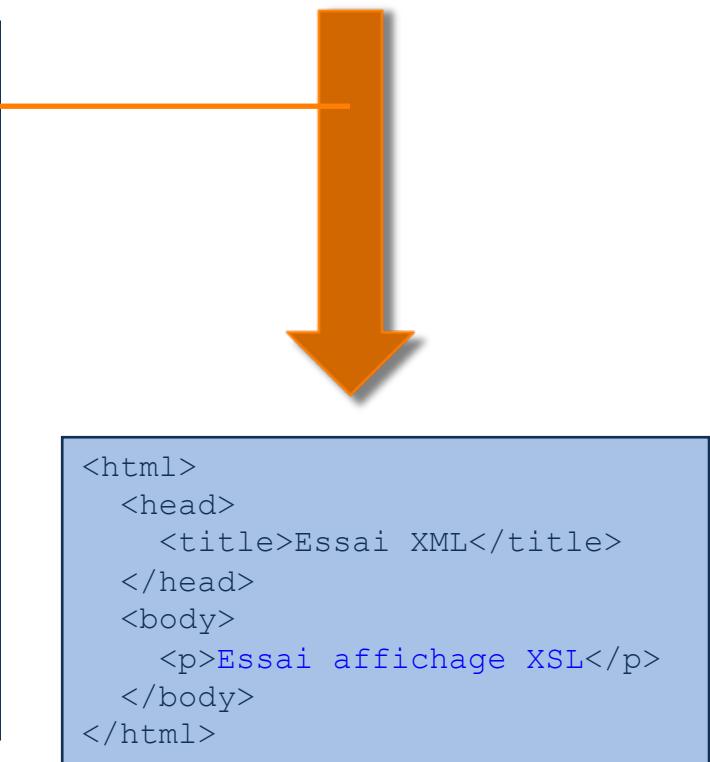
XML

+ Exemple XSL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<xsl:apply-templates select="test" />
</xsl:template>

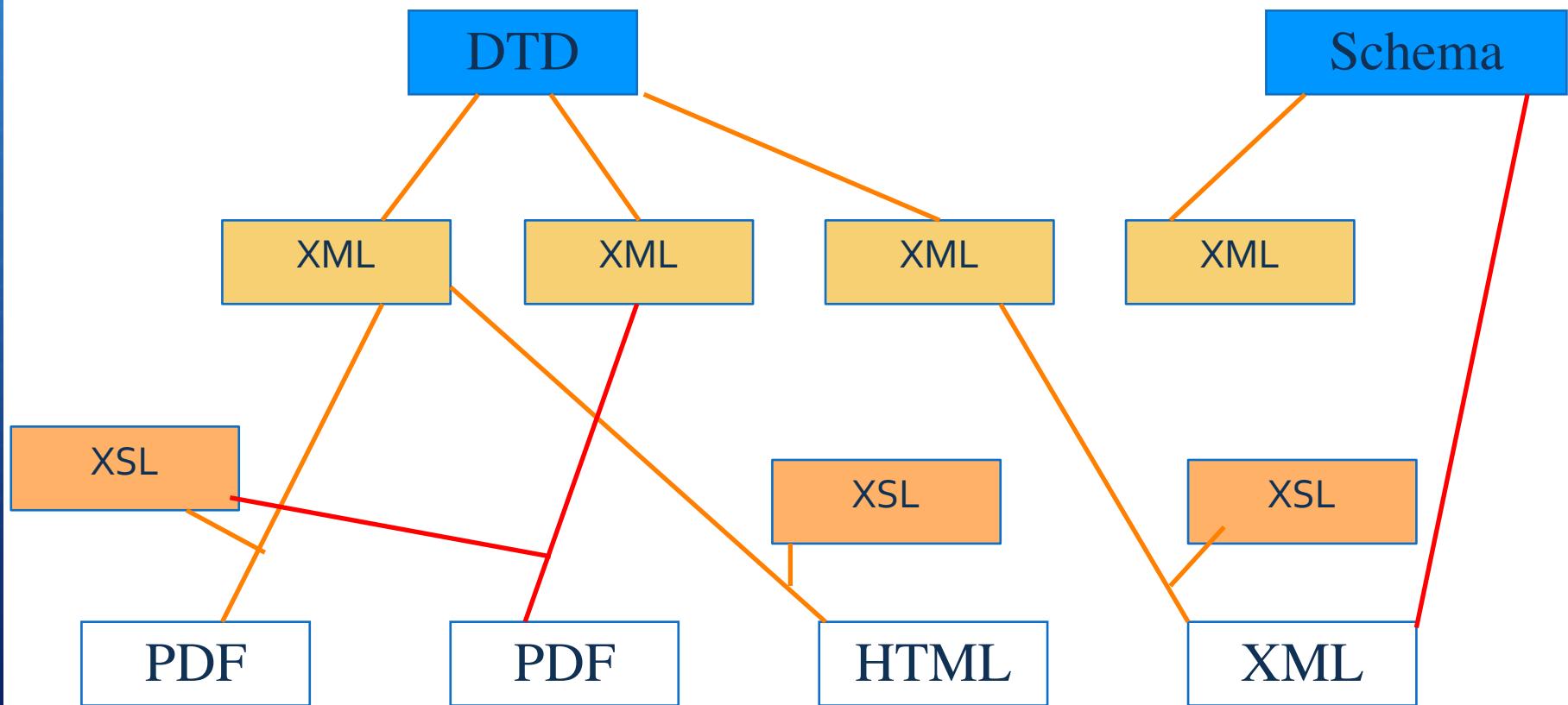
<xsl:template match="test">
<html>
  <head>
    <title>Essai XML</title>
  </head>
  <body>
    <p><xsl:value-of select=". " /></p>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<test>Essai affichage XSL</test>
```



XML

+ En résumé :



API - XML



API - XML

- + Application Programming Interface
 - + SAX
Lire le document XML
 - + DOM
Représenter le document XML en mémoire, le modifier, chercher des informations, modifier des informations
 - + JAXB - cas particulier
API spécifique JAVA de manipulation XML
 - + JiBX – amélioration de JAXB

API – XML - SAX

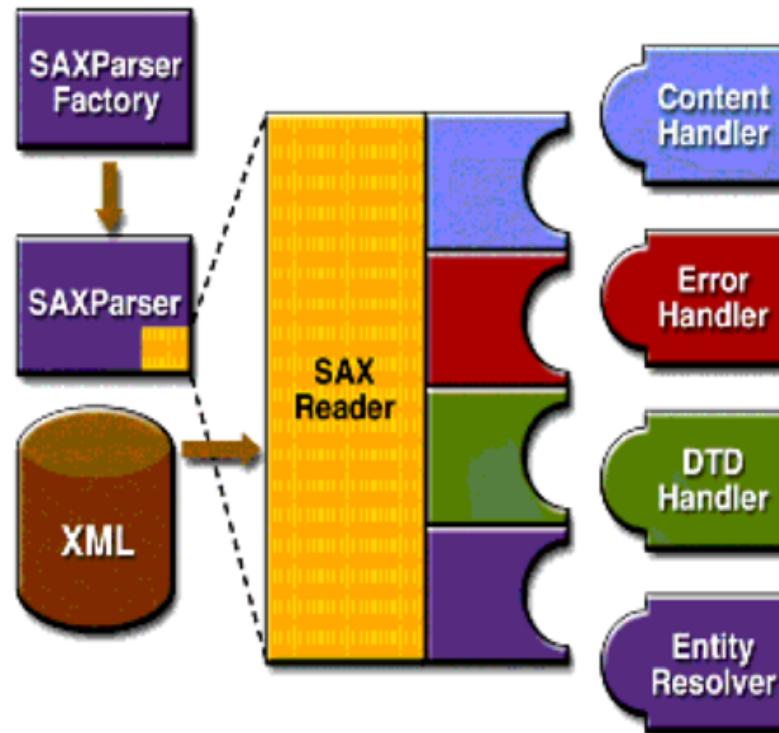
+ SAX = Simple API for XML

```
<?xml version="1.0" encoding="UTF-8"?>
<memo>
    <de>Dupont</de>
    <a>Durand</a>
    <contenu>
        <titre>Salut</titre>
        <texte>Ca va ?</texte>
    </contenu>
</memo>
```

1. StartDocument
2. StartEntity: memo
3. StartEntity: de
4. Characters: Dupont
5. EndEntity: de
6. StartEntity: a
7. Characters: Durand
8. EndEntity: a
9. StartEntity: contenu
10. StartEntity: titre
11. Characters: Salut
12. EndEntity: titre
13. StartEntity: texte
14. Characters: Ca va ?
15. EndEntity: texte
16. EndEntity: contenu
17. EndEntity: memo
18. EndDocument

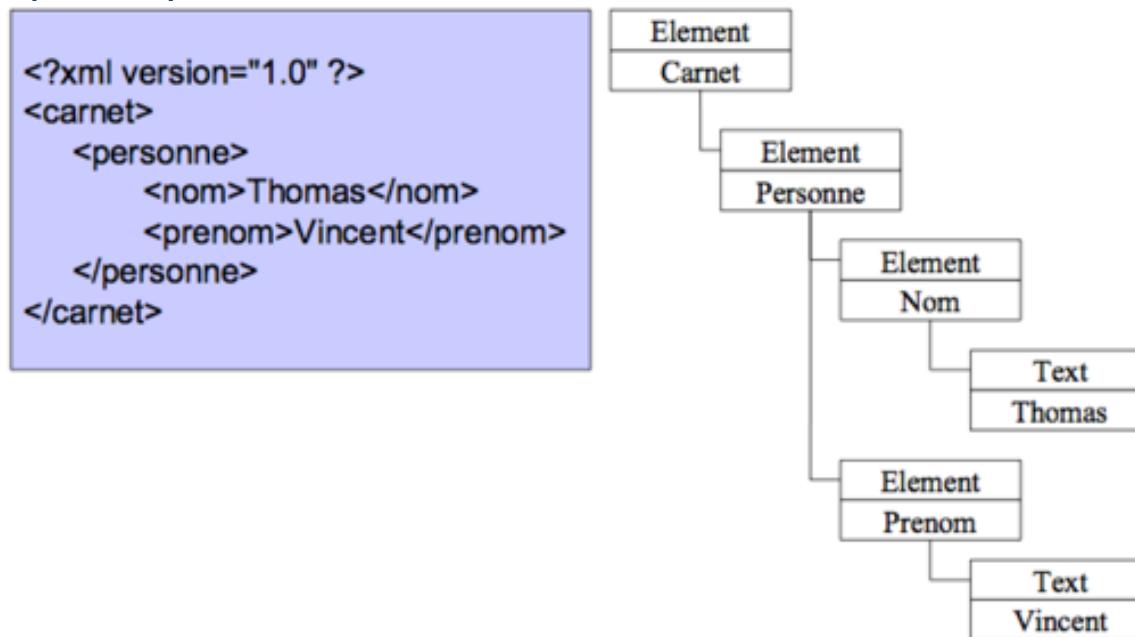
API - XML - SAX

+ SAX



API – XML - DOM

- + DOM = Document Object Model
 - + Mémoriser tout le document avec une structure de données spécifique : le DOM



API – XML - DOM

- + DOM
- + Structuration

The screenshot shows a browser's developer tools with the DOM tab selected. A specific element, an `<h1>` tag with the text "Titre", is highlighted. The left pane displays the full HTML code:

```
<!DOCTYPE html>
<html lang="fr-fr">
  <head></head>
  <body>
    <h1>Titre</h1> <span>Sous-titre avec un style local</span>
    <p id="intro">Introduction avec un id</p>
    <h2 style="color:red">Sous-titre avec un style local</h2>
    <p class="info">Ceci est un message d'information avec une classe</p>
  </body>
</html>
```

The right pane shows the properties of the selected `<h1>` element. Many properties are listed in orange, indicating they are read-only or specific to the DOM API:

- Propriétés
- clientLeft: 0
- clientTop: 0
- clientWidth: 1290
- contentEditable: "inherit"
- dataset: DOMStringMap {}
- dir: ""
- draggable: False
- firstChild: #text "Titre"
- firstElementChild: null
- hidden: false
- id: "
- innerHTML: "Titre"
- innerText: "Titre"
- isConnected: true
- isContentEditable: false
- lang: "
- lastChild: #text "Titre"
- lastElementChild: null
- localName: "h1"
- namespaceURI: "http://www.w3.org/1999/xhtml"
- nextElementSibling: <p id="intro">
- nextSibling: #text " "

- + API => Outils de manipulation
 - + JAVA : JAXP
 - + Javascript
 - + ...

API – XML - DOM

- + L'objet Document
 - + Objet de référence
 - + Permet d'accéder à l'ensemble des nœuds et des informations de l'arbre
 - + Méthodes applicables
 - + `documentElement` : la racine de l'arbre
 - + `getElementById(String id)` : permet d'accéder à un nœud à partir de son id
 - + `getElementByTag(String tagName)` : permet d'accéder aux nœuds ayant un nom de balise donné

API – XML - DOM

+ L'objet Node

- + Elément caractérisant les nœuds et les attributs de l'arbre XML
- + Il contient entre autre :
 - + nodeType : le type du noeud (ELEMENT_NODE, ATTRIBUTE_NODE, TEXT_NODE, ...)
 - + nodeName : son nom
 - + nodeValue : s'il y a lieu, sa valeur
 - + childNodes, parentNode, firstChild, lastChild, nextSibling, previousSibling : des liens vers les éléments connexes de l'arbre
 - + ownerDocument : la racine de l'arbre

API – XML - DOM

- + L'objet NodeList
 - + Liste de nœuds de l'arbre
 - + Implémente les méthodes :
 - + length() : nombre de nœuds dans la liste
 - + item(int itemNumber) : permet de récupérer un nœud de type Node

API – XML - DOM

+ L'objet Element

- + Il s'agit d'une balise
- + Il Implémente
 - + tagName : le nom de la balise
 - + getAttributes() : retourne un NodeList contenant les attributs de la balise
 - + getElementByTagName(String tagName) : retourne un NodeList contenant toutes les balises du sous-arbre dont le nom est celui indiqué.
 - + hasChildNodes()
 - + hasAttributes()

API – XML - DOM

- + Manipulation de l'arbre
 - + createElement(String tagName) : permet de créer un nouvel élément
 - + createAttribute(String attributeName) : permet de créer un attribut
 - + createTextNode(String content) : permet de créer un élément texte
- + appendChild(Node aNode)
- + insertBefore(Node nodeBefore, Node aNode)
- + removeChild(Node aChild)
- + removeAttribute(Node anAttribute)
- + removeAttributeNode(Node anAttribute)

API – XML - DOM

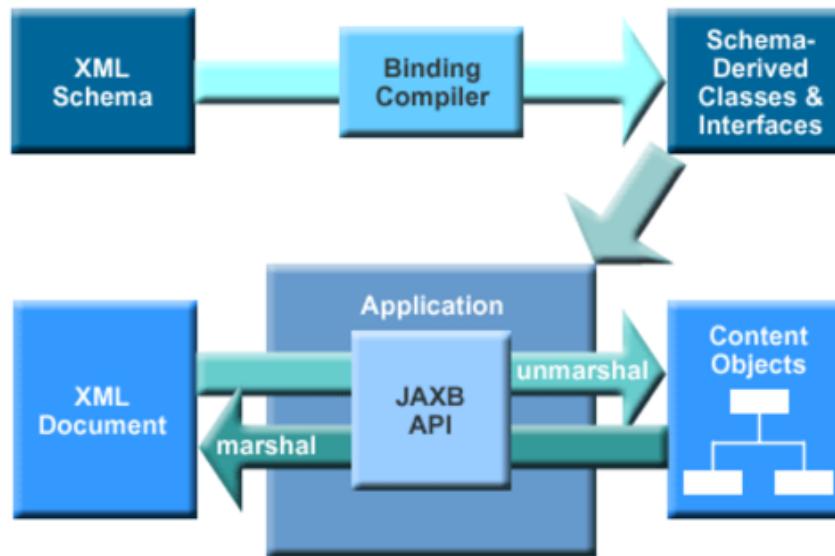
+ Exemple Javascript

```
function renumberItems(elementRef, value) {
    if ((elementRef !== null) && (elementRef.nodeType !== 3)) {
        if (elementRef.hasAttribute("name")) {
            var name = elementRef.name;
            var pos = name.indexOf("[]");
            if (pos >= 0) {
                var valueStr = name.substring(0, pos + 1) + value + "]";
                elementRef.name = valueStr;
            }
        }

        var parcours = elementRef.firstChild;
        while (parcours !== null) {
            renumberItems(parcours, value);
            parcours = parcours.nextSibling;
        }
    }
}
```

API – XML - JAXB

- + JAXB
 - + Utiliser le schema pour produire une API spécifique



```
import javax.xml.bind.JAXBContext;
import javax.xml.bind.Unmarshaller;
import java.io.File;
import java.util.List;
public class UsingJAXBTest1 {
    public static void main (String args[]) {
        try {
            JAXBContext jc = JAXBContext.newInstance("test.jaxb");
            Unmarshaller unmarshaller = jc.createUnmarshaller();
            unmarshaller.setValidating(true);

            Collection collection= (Collection) unmarshaller.unmarshal(new File( "books.xml"));
            CollectionType.BooksType booksType = collection.getBooks();
            List bookList = booksType.getBook();
            for( int i = 0; i < bookList.size();i++ ) {
                System.out.println("Book details " );
                test.jaxb.BookType book =(test.jaxb.BookType) bookList.get(i);
                System.out.println("Item id: " + book.getItemId());
                System.out.println("Book Name: " + book.getName().trim());
                System.out.println("Book ISBN: " + book.getISBN());
                System.out.println("Book Price: " + book.getPrice().trim());
                System.out.println("Book category: " + book.getBookCategory());
                System.out.println("Book promotion: " + book.getPromotion().getDiscount().trim());
                System.out.println("No of Authors " + book.getAuthors().getAuthorName().size());

                BookType.AuthorsType authors = book.getAuthors();
                for (int j = 0; j < authors.getAuthorName().size();j++) {
                    String authorName = (String) authors.getAuthorName().get(j);
                    System.out.println("Author Name " + authorName.trim());
                }
                System.out.println();
            }
        } catch (Exception e ) { e.printStackTrace(); }
    }
}
```

