XML

e**X**tensible **M**arkup **L**anguage

Reda Bendraou
Reda.Bendraou@inria.fr

http://pagesperso-systeme.lip6.fr/Reda.Bendraou/Enseignements.htm

XML

(Part III)

Layout/Formatting, Processing and Transforming XML documents

Agenda

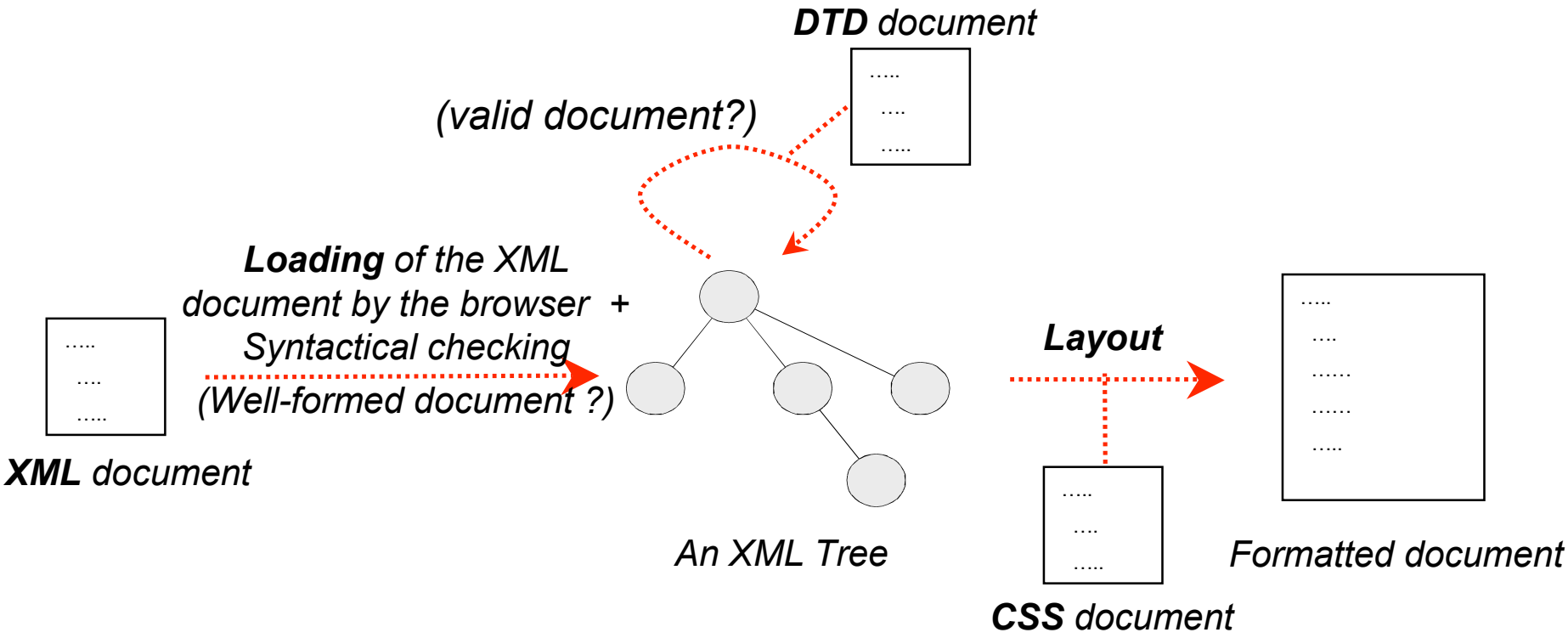
- CSS: a brief reminder
- XML documents presentation and layout with CSS
- Accessing and manipulating XML documents with DOM (Document Object Model)

What is CSS?

- **CSS** => **C**ascading **S**tyle **S**heets
- Powerful and extensible way to specify **how to display** HTML elements
- Can be Inline, Internal or External to your HTML/XML document
- **External Style Sheets** can save you a lot of work!!
  - Sharing style sheets across multiple documents or an entire website
  - External Style Sheets are stored in **CSS files**
- Multiple style definitions will **cascade** into one
  1. Browser default
  2. External style sheet
  3. Internal style sheet (inside the <head> tag)
  4. Inline style (inside an HTML element) (the highest priority)

How to use CSS with XML documents

- **Displaying an XML document using CSS:**



Syntax of a CSS Rule

- General form:

HTML element / Tag style attribute value for the attribute

selector {property: value;}

Or

selector [, selector]* { property1: value1;
property2: value2;
….
propertyN: valueN; }

- Example

H1 {text-align : center; color : blue;}

Internal Style Sheet

- Style definition within the <HEAD> tag :
  - use <style> tag with type="texte/css"

- **Example**

```
<html>
 <head>
  <style type="text/css">
   p {color: white; }
   body {background-color: black; }
  </style>
 </head>
 <body>
    <p>White text on a black background!</p>
</body>
</html>
```

Key words

CSS rules

External Style Sheet

- Declaring CSS link by:
  <link rel="stylesheet" type="text/css" href="Path_to my_Css_File.css" />

- **Example**

MyCssFile.css

```
body{ background-color: gray }
p { color: blue; }
h3 { color: white; }
```

MyHtmlFile.html

```
<html>
 <head>
  <link rel="stylesheet" type="text/css" href=" MyCssFile.css " />
 </head>
 <body> <h3> A White Header </h3>
<p> This paragraph has a blue font. The background color of this page is gray because we
    changed it with CSS! </p>
</body>
</html>
```

CSS Inline - An HTML Attribute

- CSS is built in to every HTML tag !!!
  - To add a style inside an HTML element => specify the desired CSS properties with the *style* HTML attribute

- Example

```
<p style="background: blue; color: white;">A new background and font color with inline CSS</p>
```

The Class Selector

- Helps you to define different styles for the same type of HTML element

- Syntax
  - In your CSS
    - **TagName**.**ClassName** {property1:value1;…}

    Or
    - **.ClassName** {property1:value1;…} //by omitting the tag name
  - In your HTML
    - <tagName class="ClassName">….</tagName>

Key word

The Class Selector: Example

- In your CSS document

```
p.first{ color: blue; }
p.second{ color: red; }
.title {font-style: italic; font-weight: bold;}
```

- In your HTML

```
<html>
<body>
<p>This is a normal paragraph.</p>
<p class="first">This is a paragraph that uses the p.first CSS code!</p>
<p class="second">This is a paragraph that uses the p.second CSS code!</p>
<h4 class="title"> This is a title </h1>
<h5 class="title"> This is a title </h2>
 …
</body>
</html>
```

- Display

This is a normal paragraph. → text in black

This is a paragraph that uses the p.first CSS code! → text in blue

This is a paragraph that uses the p.second CSS code! → text in red

This is a title

This is a title

Some CSS Styles

- All titles of level 1 and 2 <**H1**> et <**H2**>

```
H1, H2 { color: blue; text-decoration:underline; }
```

- All <**B**> tags within <**P**>

```
P B {background-color: #CCCCCC; font-weight: bold }
```

- All <**P**> tags with the "***plain***" class & all tags of class "**c1**"

```
P.plain {font-size:12 pt; line-height: 14pt;}
.c1 {font-size:12 pt; line-height: 14pt;}
```

- Tags with **ID**="*fancy*"

```
#fancy {font-family: Arial; font-style: italic;}
```

- <**book**> tags with attribute **categorie**="*SF*" *(only in CSS2)*

```
book[ category="SF"] {display : none;}
```

Displaying your XML Files with CSS

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
 <CD>
   <TITLE>Empire Burlesque</TITLE>
   <ARTIST>Bob Dylan</ARTIST>
   <COUNTRY>USA</COUNTRY>
   <COMPANY>Columbia</COMPANY>
   <PRICE>10.90</PRICE>
   <YEAR>1985</YEAR>
</CD>
<CD> <TITLE>Hide your heart</TITLE>
   <ARTIST>Bonnie Tyler</ARTIST>
   <COUNTRY>UK</COUNTRY>
   <COMPANY>CBS Records</COMPANY>
   <PRICE>9.90</PRICE>
   <YEAR>1988</YEAR>
</CD>
. . . .
</CATALOG>
```

```
CATALOG { background-color: #ffffff; width: 100%; }

CD { display: block; margin-bottom: 30pt; margin-left: 0; }

TITLE { color: #FF0000; font-size: 20pt; }

ARTIST { color: #0000FF; font-size: 20pt; }

COUNTRY,PRICE,YEAR,COMPANY { display: block; color: #000000; margin-left: 20pt; }
```

Example: XML & CSS using *Display* property

Displaying only SF category books of an XML document using a CSS Style Sheet

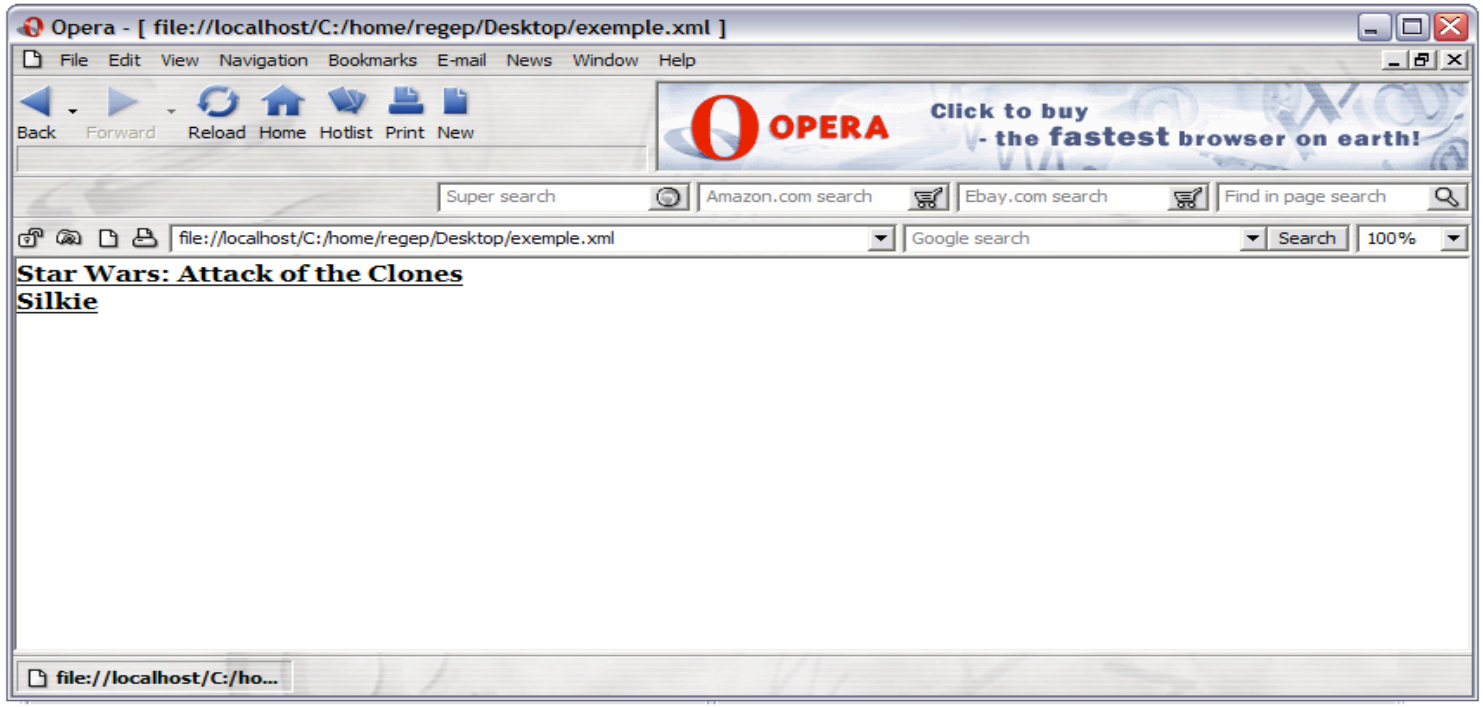
File: example.xml

```
<?xml version="1.0"?>
<?xml-stylesheet href="example.css" type="text/css" ?>
<books>
  < book category ="SF">Star Wars: Attack of the Clones</ book >
  < book category ="Literature">Le pendule de Foucault</ book >
  < book category ="SF">Silkie</ book >
  < book category ="Technical">Professional XML</ book >
</ books >
```

File : example.css

```
book[ category ="Literature"] { display : none; }
book[ category ="Technique"] { display : none;}
book[ category ="SF"] { font-family : serif; font-size: 12pt;  line-height:14pt; font-
        weight : bold; text-decoration : underline; display : block;}
```

Result!


Opera - [file://localhost/C:/home/regep/Desktop/exemple.xml]
File Edit View Navigation Bookmarks E-mail News Window Help
Back Forward Reload Home Hotlist Print New
OPERA
Click to buy
- the fastest browser on earth!
Super search
Amazon.com search
Ebay.com search
Find in page search
file://localhost/C:/home/regep/Desktop/exemple.xml
Google search
Search
100%
Star Wars: Attack of the Clones
Silkie
file://localhost/C:/ho...

Questions

- What it is the need behind separating data and presentation aspects within XML?

- Could you give some use cases of CSS styles?

The XML Document Object Model (XML DOM)

What it is XML DOM?

- The XML DOM is the Document Object Model for XML
- It is a platform- and language-independent API
  - Defines **a class hierarchies for** XML nodes processing
- **Defines a standard way to manipulate XML documents**
  - **XML document contents can be modified or deleted, and new elements can be created**
- The XML DOM is a W3C standard

XML DOM Nodes

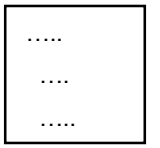
- Everything in an XML document is a node

- According to DOM:
  - The entire document is a Document node
  - Every XML tag is an Element node
  - The texts contained in the XML elements are Text nodes
  - Every XML attribute is an Attribute node
  - Comments are Comment nodes

DOM Node Hierarchy

- All nodes in an XML document form a document tree
- Each element, attribute, text, etc. in the XML document represents a node in the tree
- The tree starts at the Document node
- The tree finishes at Text nodes, the lowest level of the tree
- The terms "parent" and "child" are used to describe the relationships between nodes
- **Because the XML data is structured in a tree form :**
  - **it can't be traversed without knowing the exact structure of the tree**
  - **and without knowing the type of data contained within**

XML Tree Construction

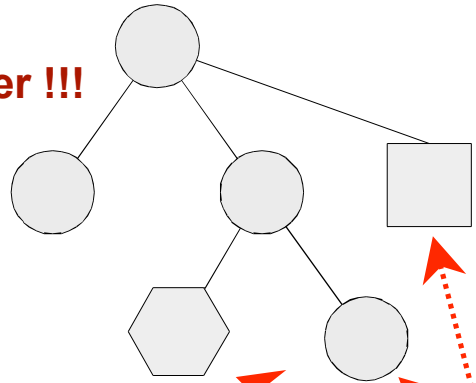


***XML** document*

Load

You need an XML Parser !!!

(well-formed ?)



XML** Tree= A **node** **hierarchies of different types

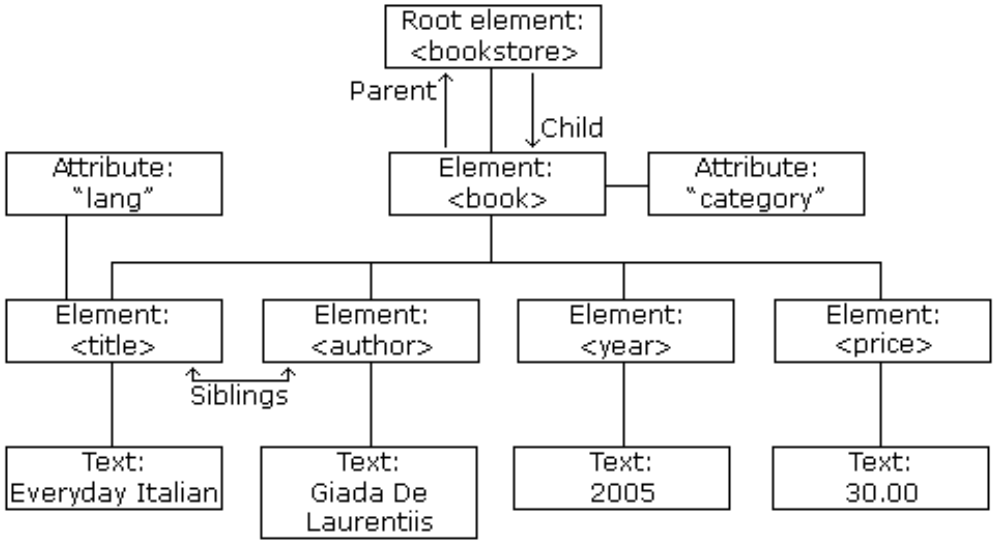
- **One root** (the document it self),
- **Processing instruction** (<? ….?>)
- **comments**, (<! --….-->)
- **elements**, (tags)
- **text**, (Tag's PCDATA)
- **attribute**, (Tag's attribute)

….. etc.

DOM Node Hierarchy Example

```
<?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>
<book category="CHILDREN">
   <title lang="en">Harry Potter</title>
   <author>J K. Rowling</author>
   <year>2005</year>
   <price>29.99</price>
 </book>
  …
<bookstore>
```

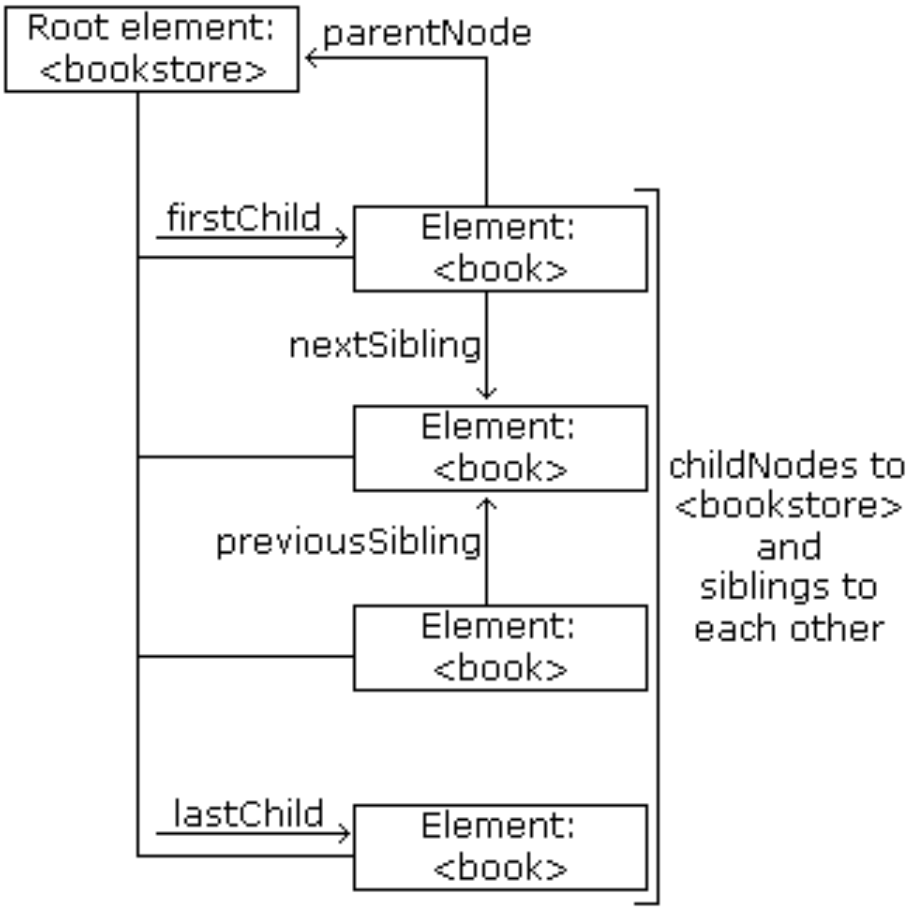
The corresponding Tree



XML DOM Node Tree

- **What is a Node Tree?**

- **We can navigate between nodes by using their relationship to each other:**
  - parentNode
  - childNodes
  - firstChild
  - lastChild
  - nextSibling
  - previousSibling

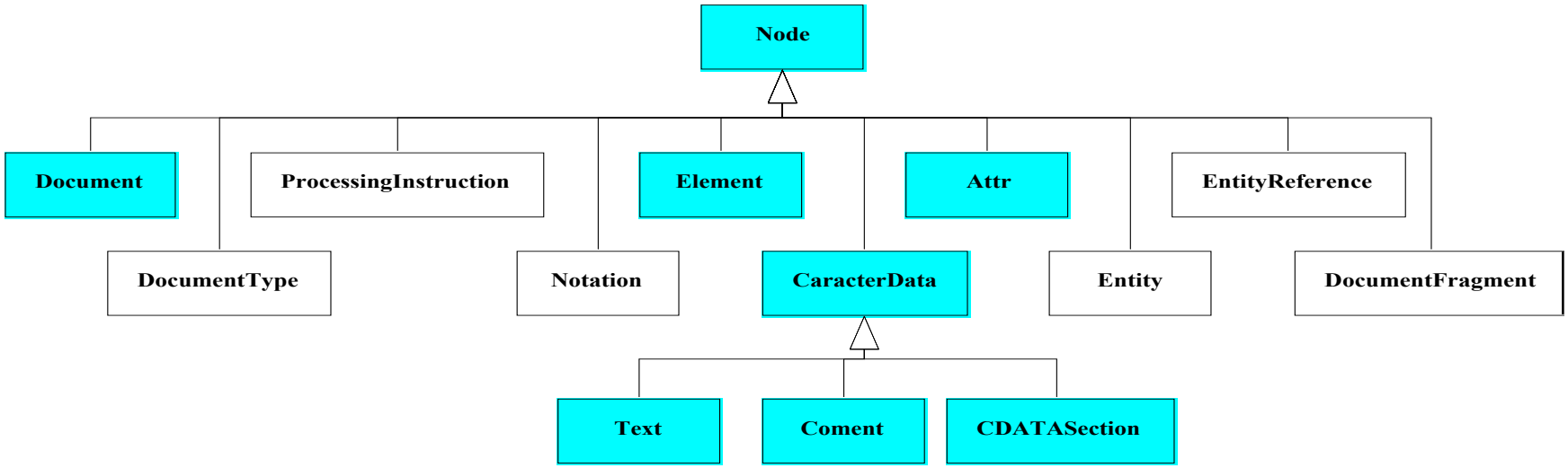


XML DOM Node Information

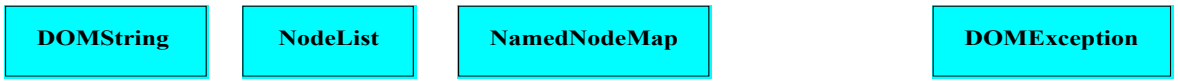
- Every node has properties:
  - **nodeName**
    - The nodeName of an element node is the tag name
    - The nodeName of an attribute node is the attribute name
    - The name of a text node is always #text
    - The name of the document node is always #document
    - **nodeName always contains the uppercase tag name of an XML element**.
  - **nodeValue**
    - On text nodes, the nodeValue property contains the text.
    - On attribute nodes, the nodeValue property contains the attribute value.
    - **The nodeValue property is not available on document and element nodes**.
  - **nodeType**
    - The nodeType property returns the type of node.
    - The most important node types are:
    - **Element (=1) Attribute (=2) Text (=3) Comment (=8) Document (=9)**

DOM Reference model (1)

- **DOM Interface hierarchies**



- **Auxiliary data types**



Interfaces explored in this lecture

DOM (2)

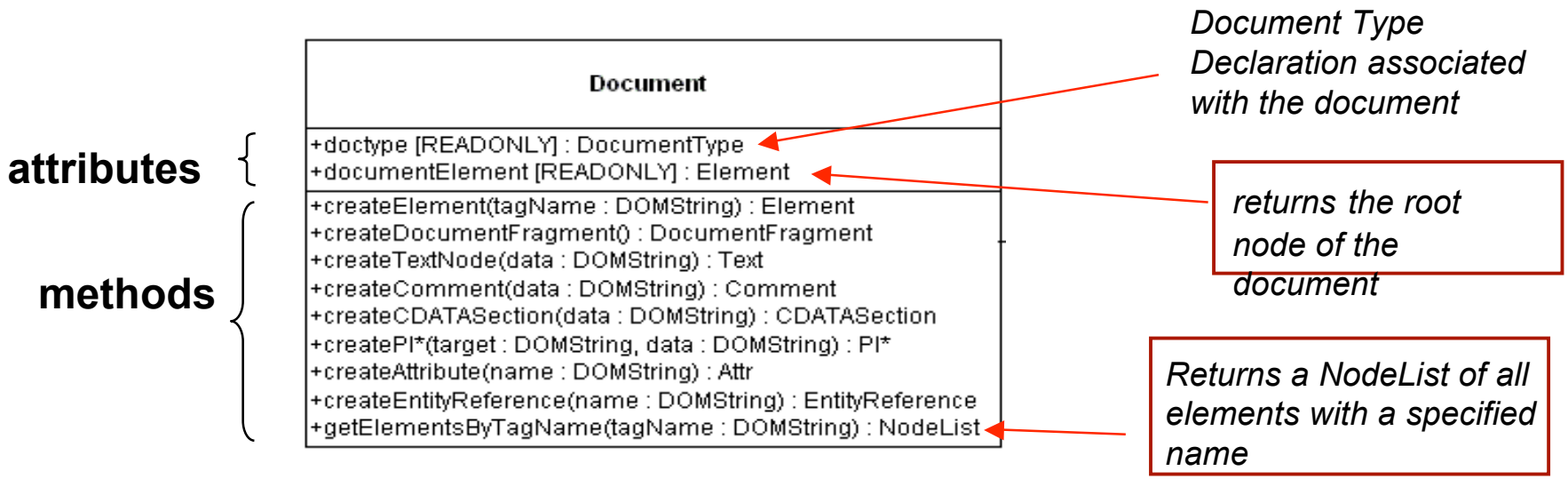
- **Primary DOM Interfaces** :
  - **Node** → a building block **class** *(every XML Tree element is a node)*
  - **Document** → the **root** of a document tree
  - **CDATASection** → **section CDATA** *(no parsed text) (<![CDATA[...]]>)*
  - **Comment** → **comment** corresponds to *(e.g. <!....>)*
  - **Element** → represents an element (**Tag**) in an XML document *(e.g. <book>)*
  - **Attr** → represents an **Attribute** of an element object
  - **Text** → represents the **textual** content of an element or attribute

- **Auxiliary DOM Interfaces** :
  - **NodeList** → represents an ordered list of nodes
  - **NamedNodeMap** → represents an unordered list of nodes
  - **DOMException** → deals with exceptions

- **Other Nodes :**
  - DocumentType, ProcessingInstruction, Notation, Entity, EntityReference, DocumentFragment .....

The Document Object

- **The Document object represents the entire XML document**
  - Contains methods to create these objects



- **Syntax of:**
  - **attributes :** *visibility_attribute_Access_Mode[Mod_Acc] : Attribute Type*
  - **methods :** *visibility_Method_Name (parameters) : returned_value_type*

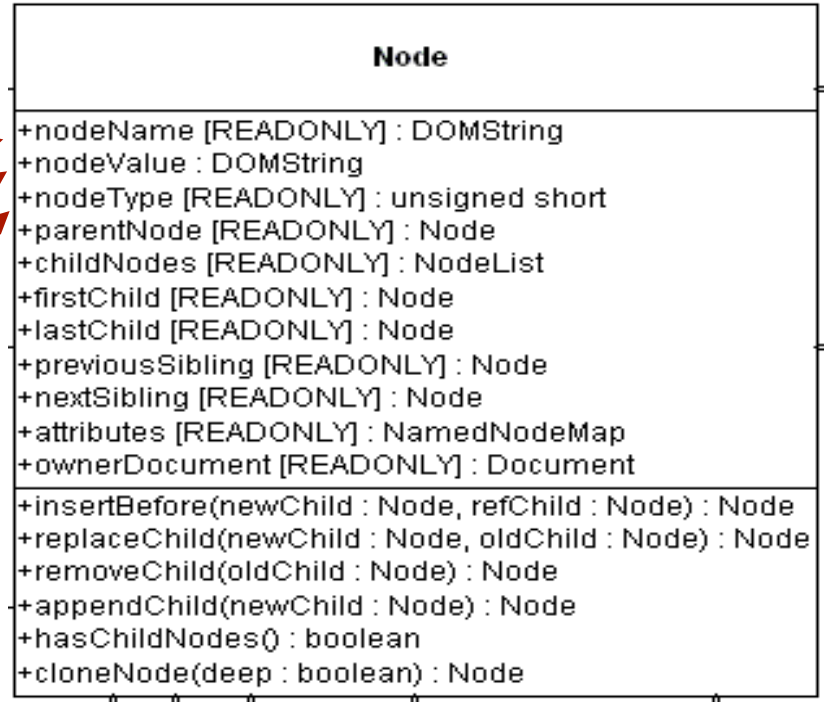
Parameter_name : Parameter_Type

The Node Object

- The Node object is the primary data type for the entire DOM

- The Node object represents a single node in the document tree (element node, an attribute node, a text node, etc.)

Node name (i.e. Tag name, attribute..)
Node Value (i.e. Tag Content, attribute value..)
Node Type (1=Element; 2=Attribute; 3=Text…)
Parent Node
List of Child Nodes
....
....

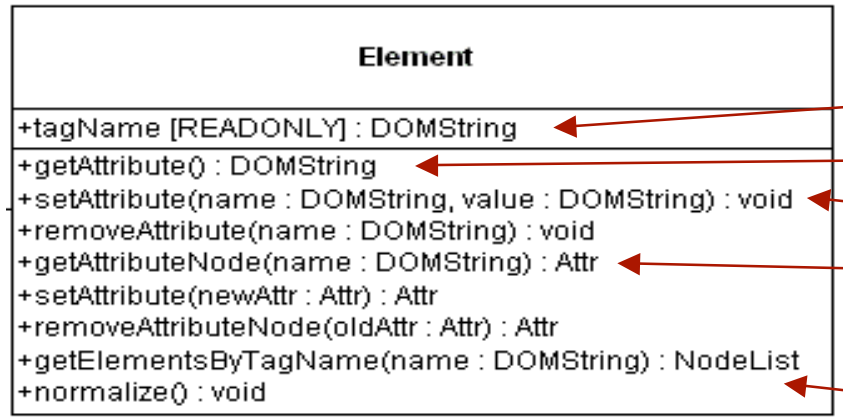


The Element Object

- The Element object represents an element (Tag) in an XML document.
  - Inherits Node

- Elements may contain attributes, other elements, or text.

- **Important** : If an element contains text, the text is represented in a text-node.

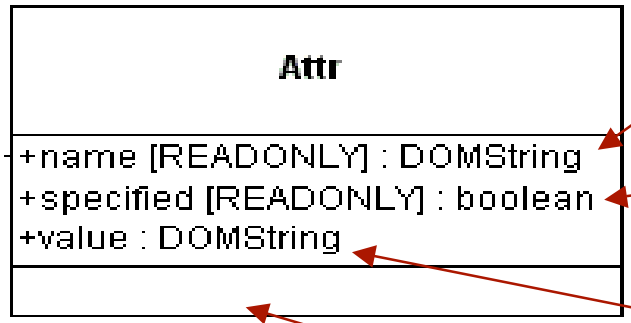


Tag Name (e.g. BOOK)
Get an attribute value by name
Set an attribute value by name
Get an attribute node by name
…
returns a NodeList of all a elements with a specified name. A Very useful method !!!!!

The Attr (attribute) Object

- **Represents an attribute of an Element object**

- **An attribute does not have a parent node and is not considered to be a child node of an element**



Attribute Name (e.g. "style")

Returns true if the attribute value is set in the document, and false if it's a default value in a DTD/Schema

Attribute Value (e.g. "display : none" or "" if empty)

….. No methods

The Text Object

- The Text object represents the textual content of an element or attribute

- Some Text object properties:

  - **data**: Sets or returns the text of the element or attribute

  - **length** : Returns the length of the text of the element or attribute

- Some methods:

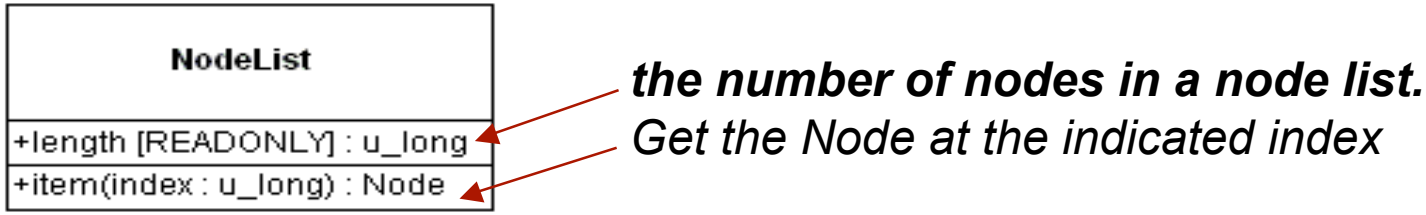
  - appendData(): Appends data to the node

  - substringData(): Extracts data from the node

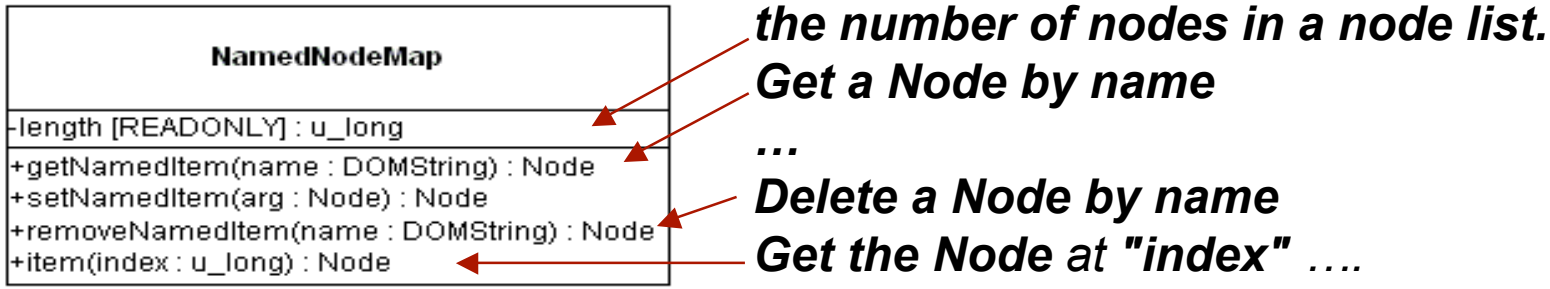
  - deleteData(): Deletes data from the node

Auxiliary DOM objects

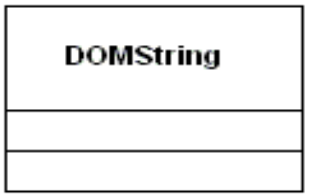
- **Node List : represents an ordered list of nodes**



- **NamdedNodeMap : represents an unordered list of nodes**



- **Represents a String in DOM** :



Parsing the XML DOM

- To read and update, create and manipulate an XML document, you will need an XML parser !!!!!
- The parser loads the document into your computer's memory.
- Once the document is loaded, its data can be manipulated using the DOM.
- The DOM treats the XML document as a tree

Steps to use DOM: Creating a Parsed Document

- **1.Import XML-related packages:**

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import java.io.*;
```

- **2.Create a DocumentBuilder:**

```
DocumentBuilderFactory factory =DocumentBuilderFactory.newInstance();

DocumentBuilder builder = factory.newDocumentBuilder();
```

- **3.Create a Document from a file or stream:**

```
Document document= builder.parse(newFile(file));
```

(source ® Marty Hall slides 2006)

Steps to use DOM: Creating a Parsed Document

- **4.Extract the root element**

```
Element root = document.getDocumentElement();
```

And Then, you can:

- **5.Examine attributes**

  getAttribute("attributeName") **returns specific attribute**

  getAttributes() **returns a Map (table) of names/values**

- **6.Examine sub-elements**

  getElementsByTagName("subelementName") returns a list of subelementsof specified name

  getChildNodes() returns a list of all child nodes

  Both methods return data structure containing Node entries, not Element.

  »Node is parent interface of Element

  »Results of getElementsByTagName can be typecast to Element

  »Results of getChildNodes are Node entries of various types

Example DOM with Java : The XML document

The XML document as input :

```
<?xml version="1.0" encoding="UTF-8"?>
<dataroot>
<ref_biblio>
   <ref>Globe99</ref>
   <type>article</type>
   <author>Van Steen, M. and Homburg, P.  and Tanenbaum, A.S.</author>
   <title>Globe: A Wide-Area Distributed System</title>
  </ref_biblio>
  <ref_biblio>
   <ref>ada-rm</ref>
   <type>techReport</type>
   <author>International Organization for Standardization</author>
   <title>Ada Reference Manual</title>
  </ref_biblio>
</dataroot>
```

How to reach this text ?

Example DOM with Java : Your Java code

```
public class Example {
 public static void main(String[] args) {

    /*   instantiate the Parser Constructor  */
    DocumentBuilderFactory _factory = DocumentBuilderFactory.newInstance();

    /*   ignore comments within the parsed XML document    */
    _factory.setIgnoringComments(true);

    /*   create a Document builder (parser) that will parse the XML document to DOM
    document instances*/
    DocumentBuilder _builder = _factory.newDocumentBuilder();

    /*   Load the document using the parser   */
    String filepath = ".\\bib.xml";

    Document doc = _builder.parse(filepath);
    /*    Get the document Root    */
    Element library = doc.getDocumentElement();
```

document Root

Reusable part for each use of DOM with Java

Example DOM with Java : Your Java code

```
/*    get the list  of all nodes (Childs,  grandsons …)    */
NodeList allChilds = library .getChildNodes();

/*    go through the list  */
for (int i = 0; i <  allChilds.getLength() ; i++) {
    Node node = allChilds.item(i);

      /*   Check if the Node is an Element (Tag) and not an Attr or Text Node    */
      if (node.getNodeType() = Node.ELEMENT_NODE) {

          / *    caste Node into Element   */
          Element elt = (Element) node;

          /*   get the Element (Tag) "title"   */
          Element title = (Element) elt.getElementsByTagName("title").item(0) ;

          /*   Get its unique child which is of type Text */
          Node text = title.getFirstChild() ;

          /* get the value of the Text node which is the data we are looking for */
          String titre = text. getNodeValue();
          }
}}}
```

Of course, you can do simpler, it's just an example to illustrate the use of some DOM methods. You have to take care of exceptions!!!!

DOM Summary

- A Standardized API (W3C)
- XML Object tree construction :
  - Easy programming
  - API platform and language independent
  - A bit long to execute with huge XML docuements→ use SAX for scalability
- To simplify XML tree browsing → XPath

XML and DOM Resources

- Java API Docs
  http://java.sun.com/j2se/1.5.0/docs/api/
  http://java.sun.com/j2se/1.5.0/docs/api/org/w3c/dom/Node.html
  http://java.sun.com/j2se/1.5.0/docs/api/org/w3c/dom/Element.html
- XML 1.0 Specification
  http://www.w3.org/TR/REC-xml

- WWW consortium's Home Page on XML
  http://www.w3.org/XML/

Sun Page on XML and Java
  http://java.sun.com/xml/

O'Reilly XML Resource Center
  http://www.xml.com/