XML

(v 0.3)

PXC

Manel Guerrero <guerrero@ac.upc.edu>
René Serral <rserral@ac.upc.edu>
Alberto Cabellos <acabello@ac.upc.edu>

Contents

- HTML
- XML
- RSS and XHTML
- DTD and XML Schema
- CSS (for HTML and for RSS)
- XSLT and XPATH
- DOM and SAX

Sources

(That is, places from which we've done merciless cut 'n' pastes)

- David Carlson: "Modeling XML Applications with UML", Ed. Addison-Wesley.
- www.wikipedia.org
- www.webopedia.com
- Other places from the Internet

HTML

- HyperText Markup Language (HTML) is a markup language designed for the creation of web pages and other information viewable in a browser. HTML is used to structure information (denoting certain text as headings, paragraphs, lists and so on) and can be used to define the semantics of a document.
- Originally defined by Tim Berners-Lee and further developed by the IETF with a simplified SGML syntax, HTML is now an international standard (ISO/IEC 15445:2000). The HTML specification is maintained by the World Wide Web Consortium (W3C).
- HTML defines the structure and layout of a Web document by using a variety of tags and attributes. The correct structure for an HTML document starts with <HTML><HEAD>(with the title and other stuff)</HEAD><BODY> and ends with </BODY></HTML>. All the information you'd like to include in your web page fits in between the <BODY> and </BODY> tags.

HTML: Version history

- There is no official standard HTML 1.0 specification because there were multiple informal HTML standards at the time.
- HTML 2.0 — published November 1995 as IETF RFC 1866, and declared obsolete/historic by RFC 2854 in June 2000
- HTML 3.2 — published January 14, 1997 as a W3C Recommendation
- HTML 4.0 — published December 18, 1997 as a W3C Recommendation
- HTML 4.01 (minor fixes) — published December 24, 1999 as a W3C Rec.
- Minor editorial revisions to the HTML 4.0 specification were published as HTML 4.01. Due to the advent of XHTML, there will not be any more new versions of HTML. The most common extension for HTML is '.html,' however, previous operating systems limited file extensions to three letters, so a '.htm' extension was also once used, and works with most browsers.

HTML: Markup elements

- **Structural markup:** Describes the purpose of text. For example, "<h2>Golf</h2>" directs the browser to render "Golf" as a second-level heading. Structural markup does not denote any specific rendering, but most web browsers have standardised on how elements should be formatted.
- **Presentational markup:** Describes the appearance of the text, regardless of its function. For example, "boldface" will render "boldface" in bold text. In the majority of cases, using presentational markup is inappropriate, and presentation should be controlled by using CSS.
- **Hypertext markup:** Links parts of the document to other documents. For example, "Wikipedia" will render the word Wikipedia as a hyperlink to the specified URL.

HTML: Document Type Definition

- In order to specify which version of the HTML standard they conform to, all HTML documents should start with a Document Type Declaration (informally, a "DOCTYPE"), which makes reference to a Document Type Definition (DTD). For example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.w3.org/TR/html4/strict.dtd">
```

- This declaration asserts that the document conforms to the Strict DTD of HTML 4.01, which is purely structural, leaving formatting to Cascading Style Sheets. In some cases, the presence or absence of an appropriate DTD may influence how a web browser will display the page.
- Let's not worry too much about how DTDs work right now. We will see them soon.

HTML example

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN" "strict.dtd">
<HTML>
  <HEAD>
    <TITLE>UML Headlines</TITLE>
    <META NAME="managingEditor" CONTENT="editor@xmlmodeling.com">
  </HEAD>
  <BODY>
    <H1>UML Headlines</H1>
    <P>Recent news about the Unified Modeling Language (UML).</P>
    <UL>
      <LI><A HREF="http://www.omg.org">UML version 1.3 adopted by the
            OMG</A></LI>
      <LI><A HREF="http://www.rational.com">Rational Rose 2000e
            released</A></LI>
      <LI><A HREF="http://www.togethersoft.com">TogetherJ 4.0
            released</A></LI>
    </UL>
  </BODY>
</HTML>
```

XML

- HTML, although originally designed independently, was later reformulated (at version 2.0) to be an application of SGML metalanguage, although there's some debate on whether it ever actually became one.
- The Extensible Markup Language (XML) is a W3C-recommended general-purpose markup language for creating special-purpose markup languages. It is a simplified subset of SGML, capable of describing many different kinds of data. Its primary purpose is to facilitate the sharing of data across different systems. Languages based on XML (for example, RSS, MathML (Mathematical Markup Language), XHTML, SVG(Scalable Vector Graphics), etc) are defined in a formal way, allowing programs to modify and validate documents in these languages without prior knowledge of their form.
- Therefore, SGML (Standard Generalized Markup Language) is a metalanguage in which one can define markup languages like HTML, and XML is another metalanguage which has been used to define RSS, XHTML, etc.

Correctness in an XML document

- For an **XML document** to be **correct**, it must be:
  - **Well-formed**. A well-formed document conforms to all of XML's syntax rules. For example, if a non-empty element has an opening tag with no closing tag, it is not well-formed. A document that is not well-formed is not considered to be XML; a parser is required to refuse to process it.
  - **Valid**. A valid document has data that conforms to a particular set of user-defined content rules that describe correct data values and locations (schema). For example, if an element in a document is required to contain text that can be interpreted as being an integer numeric value, and it instead has the text "hello", is empty, or has other elements in its content, then the document is not valid.
- An XML schema is a description of a type of XML document, typically expressed in terms of constraints on the structure and content of documents of that type. Examples are DTD and XML Scheme.

XML Strenghts

- The features of XML that make it well-suited for data transfer are:
  - Simultaneously human- and machine-readable format
  - Support for Unicode, allowing the use of any human language
  - Ability to represent records, lists and trees
  - Self-documenting format that describes structure and field names
  - Strict syntax allows simple, efficient, and consistent parsing algorithms
- XML is also heavily used as a format for document storage and processing:
  - Its robust, logically-verifiable format is based on international standards
  - Its hierarchical structure is suitable for most types of documents
  - It manifests as plain text files, unencumbered by licenses or restrictions
  - It's platform-independent, thus relatively immune to changes in technology

XML Weaknesses

- Its syntax is fairly verbose and partially redundant. This can hurt human readability and application efficiency, and yields higher storage costs. Though compression can reduce the problem in some cases.
- Parsers should be designed to recursively handle arbitrarily nested data structures and must perform additional checks to detect improperly formatted or differently ordered syntax or data (with its associated overhead).
- The basic parsing requirements do not support a very wide array of data types. There is no way to know if "3.14159" is a floating-point number rather than a seven-character string. XML schema languages add this functionality.
- Modelling overlapping (non-hierarchical) data structures requires extra effort.
- Mapping XML to the relational or object oriented paradigms is often cumbersome. Not everything can be mapped to XML.
- It is, arguably, not good for high volume data.

RSS and Atom

- **RSS** is a family of XML file formats for web syndication used by (amongst other things) news websites and weblogs. It has these versions:
  - Rich Site Summary (RSS 0.91)
  - RDF Site Summary (RSS 0.9 and 1.0)
  - Really Simple Syndication (RSS 2.0)
- The technology behind RSS allows you to subscribe to websites that have provided RSS feeds, these are typically sites that change or add content regularly. To use this technology you need to set up some type of aggregation service. Think of this aggregation service as your personal mailbox. You then have to subscribe to the sites that you want to get updates on.
- **Atom:** In reaction to perceived deficiencies RSS, a third group started a new syndication specification, Atom, in June 2003, and their work was later adopted by Internet Engineering Task Force (IETF).

RSS example (1/2)

```
<?xml version="1.0"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN"
        "rss-0.91.dtd">
<rss version="0.91">
  <channel>
    <title>UML Headlines</title>
    <description>Recent news about the Unified Modeling Language (UML).
        </description>
    <language>en-us</language>
    <link>http://xmlmodeling.com</link>
    <managingEditor>editor@xmlmodeling.com</managingEditor>
    <skipDays>
      <day>Saturday</day><day>Sunday</day>
    </skipDays>
    <pubDate>July 1, 2000</pubDate>
    <image>
      <title>UML Headlines</title>
      <url>http://xmlmodeling.com/images/xmlmodeling.jpg</url>
      <link>http://xmlmodeling.com</link>
      <width>88</width>
      <height>31</height>
    </image>
```

RSS example (2/2)

```
[Continued]

    <item>
      <title>UML version 1.3 adopted by the OMG</title>
      <link>http://www.omg.org</link>
      <description>The OMG's UML specification is the industry standard for
          analysis and design.</description>
    </item>
    <item>
      <title>Rational Rose 2000e released</title>
      <link>http://www.rational.com</link>
      <description>Rational announced the release of Rational Rose
          2000e.</description>
    </item>
    <item>
      <title>TogetherJ 4.0 released</title>
      <link>http://www.togethersoft.com</link>
      <description>The Together 4.0 product line is now shipping.</description>
    </item>
  </channel>
</rss>
```

Document Type Definition (DTD)

- A DTD is a set of declarations that conform to a particular markup syntax and that describe a class, or "type", of SGML or XML documents, in terms of constraints on the structure of those documents.
- A DTD specifies, in effect, the syntax of an "application" of SGML or XML, such as the derivative language HTML or XHTML.
- In a DTD, the structure of a class of documents is described via element and attribute-list declarations. Element declarations name the allowable set of elements within the document, and specify whether and how declared elements and runs of character data may be contained within each element. Attribute-list declarations name the allowable set of attributes for each declared element, including the type of each attribute value, if not an explicit set of valid value(s).
- A DTD may also declare default attribute values, named entities and their replacement text, and other constructs that are not always required to establish document structure, but that may affect the interpretation of some documents.

RSS DTD

```
<!ELEMENT rss (channel)>
<!ATTLIST rss version     CDATA #REQUIRED> <!-- must be "0.91"> -->

<!ELEMENT channel (title | description | link | language
    | managingEditor? | pubDate? | image? | skipDays? | item+ )*>
<!ELEMENT image (title | url | link | width? | height?
    | description?)*>
<!ELEMENT item (title | link | description)*>

<!ELEMENT title (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ELEMENT link (#PCDATA)>
<!ELEMENT language (#PCDATA)>
<!ELEMENT managingEditor (#PCDATA)>
<!ELEMENT pubDate (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ELEMENT width (#PCDATA)>
<!ELEMENT height (#PCDATA)>
<!ELEMENT skipDays (day+)>
<!ELEMENT day (#PCDATA)>
```

DTD (1/2)

- <!ELEMENT e >: Element description.
- <!ATTLIST e ats>: Description of the attributes of an element.
- #PCDATA: (Parsed Character DATA) Text that cannot contain reserved chars ('<', '&', etc). The 'element content' betwen the start-tag and end-tag.
- CDATA: (Character data) Text that you don't want to be parsed (cannot contain ']]>'). In XML, the element 'comparison' with value "6 is < 7 & 7> 6" would be: "<comparison><![CDATA[6 is < 7 & 7 > 6]]></comparison>"
- "a (b)" denotes that 'b' is nested in 'a' or that the data type of 'a' is 'b'.
- "(a | b)" denotes 'a' or 'b' and "(a,b)" denotes 'a' followed by 'b'.
- "*" denotes there can be 0 or many elements and "+" denotes 1 or more.
- "?" indicates that an element is optional (0 or 1 element).

DTD (2/2)

- Attribute modifiers:
  - #REQUIRED: The value must be provided
  - #IMPLIED: It has no default value
  - #FIXED "Foobar": It's value is constant (is "Foobar"). Not very used. If the value is different the parser will return an error.

- Specifying a Default attribute value and Empty elements:

```
<!ELEMENT square EMPTY>
<!ATTLIST square width CDATA "0">
```

  - The "square" element is defined to be an empty element with a "width" attribute of type CDATA. If no width is specified, it's default value is '0'.

XML Schema

- XML schema languages: DTD, XML Schema, etc.
- XML Schema, published as a W3C Recommendation in May 2001, is one of several XML schema languages. It was the first separate schema language for XML to achieve Recommendation status by the W3C.
- An XML Schema instance is an XML Schema Definition (XSD) and typically has the filename extension ".xsd".
- After XML Schema-based validation, it is possible to express an XML document's structure and content in terms of the data model that was implicit during validation. The XML Schema data model includes:
  - the vocabulary (Element/Attribute names)
  - the content model (Relationships/Structure)
  - and data types.

XML Schema Example

```
Schema:
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="country">
 <xs:complexType>
   <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="pop" type="xs:decimal"/>
   </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>

XML:
<country
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="country.xsd">
  <name>France</name>
  <pop>59.7</pop>
</country>
```

XML Schema More Examples (1/2)

- minOccurs and maxOccurs:

```
<xs:element name="minister" type="xs:string"
     minOccurs="0" maxOccurs="unbounded"/>
```

- choice:

```
<xs:choice>
  <xs:element name="president" type="xs:string"/>
  <xs:element name="monarch" type="xs:string"/>
</xs:choice>
```

- list:

```
<xs:simpleType name="listOfMyIntType">
  <xs:list itemType="myInteger"/>
</xs:simpleType>

Instance document: <listOfMyInt>20003 15037 95977 95945</listOfMyInt>
```

XML Schema More Examples (2/2)

- Defining myInteger, **Range** 10000-99999

```
<xsd:simpleType name="myInteger">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="10000"/>
    <xsd:maxInclusive value="99999"/>
  </xsd:restriction>
</xsd:simpleType>
```

- Using the **Enumeration** Facet:

```
<xsd:simpleType name="USState">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AK"/>
    <xsd:enumeration value="AL"/>
    <xsd:enumeration value="AR"/>
    <!-- and so on ... -->
  </xsd:restriction>
</xsd:simpleType>
```

XHTML

- Extensible HyperText Markup Language, or XHTML, is a markup language that has the same expressive possibilities as HTML, but a stricter syntax. Whereas HTML is an application of SGML, a very flexible markup language, XHTML is an application of XML, a more restrictive subset of SGML. Because they need to be well-formed (syntactically correct), XHTML documents allow for automated processing to be performed using a standard XML library — unlike HTML, which requires a relatively complex, lenient, and generally custom parser.
- XHTML 1.0 became a World Wide Web Consortium (W3C) Recommendation on January 26, 2000.

XHTML (2)

- The need for a more strict version of HTML was felt primarily because World Wide Web content now needs to be delivered to many devices (like mobile devices) apart from traditional computers, where extra resources cannot be devoted to support the additional complexity of HTML syntax.
- Most of the recent versions of popular web browsers render XHTML properly, and many older browsers will also render XHTML as it is mostly compatible with HTML and most browsers do not require valid HTML. Similarly, almost all web browsers that are compatible with XHTML also render HTML properly. Some argue this compatability is slowing the switch from HTML to XHTML.

XHTML: differences with HTML

- Documents must be well-formed: all elements must either have closing tags or use the special form "<foobar />" and that all the elements must nest properly. <u>wrong</u>
- Element and attribute names must be in lower case (because XML is case-sensitive). not <LI>
- For non-empty elements, end tags are required. Foobar.
- Attribute values must always be quoted.<td rowspan="3">
- XML does not support attribute minimization. <dl compact="compact"> is correct and <dl compact> is incorrect.
- Empty elements must either have an end tag or the start tag must end with "/>".
<hr/>
- And some others.

XHTML: Common errors (1/3)

- Not closing empty elements (elements without closing tags)
  - Incorrect:
  Correct:
- Not closing non-empty elements
  - Incorrect: This is a paragraph. This is another paragraph.
  - Correct: This is a paragraph. This is another paragraph.
- Improperly nesting elements (elements must be closed in reverse order)
  - Incorrect: This is some text.
  - Correct: This is some text.
- Not putting quotation marks around attribute values
  - Incorrect: <td rowspan=3> Correct: <td rowspan="3">

XHTML: Common errors (2/3)

- Not specifying alternate text for images (using the alt attribute)
  - Incorrect:
  - Correct:
- Putting text directly in the body of the document
  - Incorrect: <body>Welcome to my page.</body>
  - Correct: <body> Welcome to my page. </body>
- Nesting block-level elements within inline elements
  - Incorrect: <h2>Introduction</h2>
  - Correct: <h2>Introduction</h2>

XHTML: Common errors (3/3)

- Using the ampersand outside of entities (use & instead)
  - Incorrect: <title>Cars & Trucks</title>
  - Correct: <title>Cars & Trucks</title>
- Using uppercase tag names and/or tag attributes
  - Incorrect: <BODY><P>The Best Page Ever</P></BODY>
  - Correct: <body> The Best Page Ever </body>
- Attribute minimization
  - Incorrect: <textarea readonly>READ-ONLY</textarea>
  - Correct: <textarea readonly="readonly">READ-ONLY</textarea>

XML Example

The following is an example of XHTML 1.0 Strict:

```
8<----------

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <title>XHTML Example</title>
    </head>
    <body>
        <p>This is a tiny example of an XHTML document.</p>
    </body>
</html>

---------->8
```

CSS

- Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML, but the language can be applied to any application of XML.
- CSS is used by both the authors and readers of web pages to define colors, fonts, layout, and other aspects of document presentation. It is designed primarily to enable the separation of document structure (written in HTML or a similar markup language) from document presentation (written in CSS). This separation provides a number of benefits, including improved content accessibility, greater flexibility and control in the specification of presentational characteristics, and reduced complexity of the structural content.
- CSS is also capable of controlling the document's style separately in alternative rendering methods, such as on-screen, in print, by voice (when read out by a speech-based browser or screen reader) and on braille-based, tactile devices.

CSS stylesheet for HTML

```
BODY {
  font-family: "Times New Roman";
  font-size: 12pt;
}

H1 {
  font-family: Arial;
  font-weight: bold;
  text-align: center;
  color: blue;
  font-size: 14pt;
}

LI {
  font-family: "Arial";
  font-size: 10pt;
}
```

• You can specify styles in the html file that only apply to one element:

```
<LI STYLE="color: red">
  <A HREF="http://www.debian.org">
    Debian forever</A>
</LI>
```

CSS stylesheet for HTML

- The stylesheet can be embedded in the HTML document:

```
<head>
[...]
<style type="text/css">
  body { color: black; background: white; }
</style>
[...]
</head>
```

- Or it can be in a separated file:

```
<link type="text/css" rel="stylesheet" href="style.css">
```

(So different HTML documents can refer to the same stylesheet.)

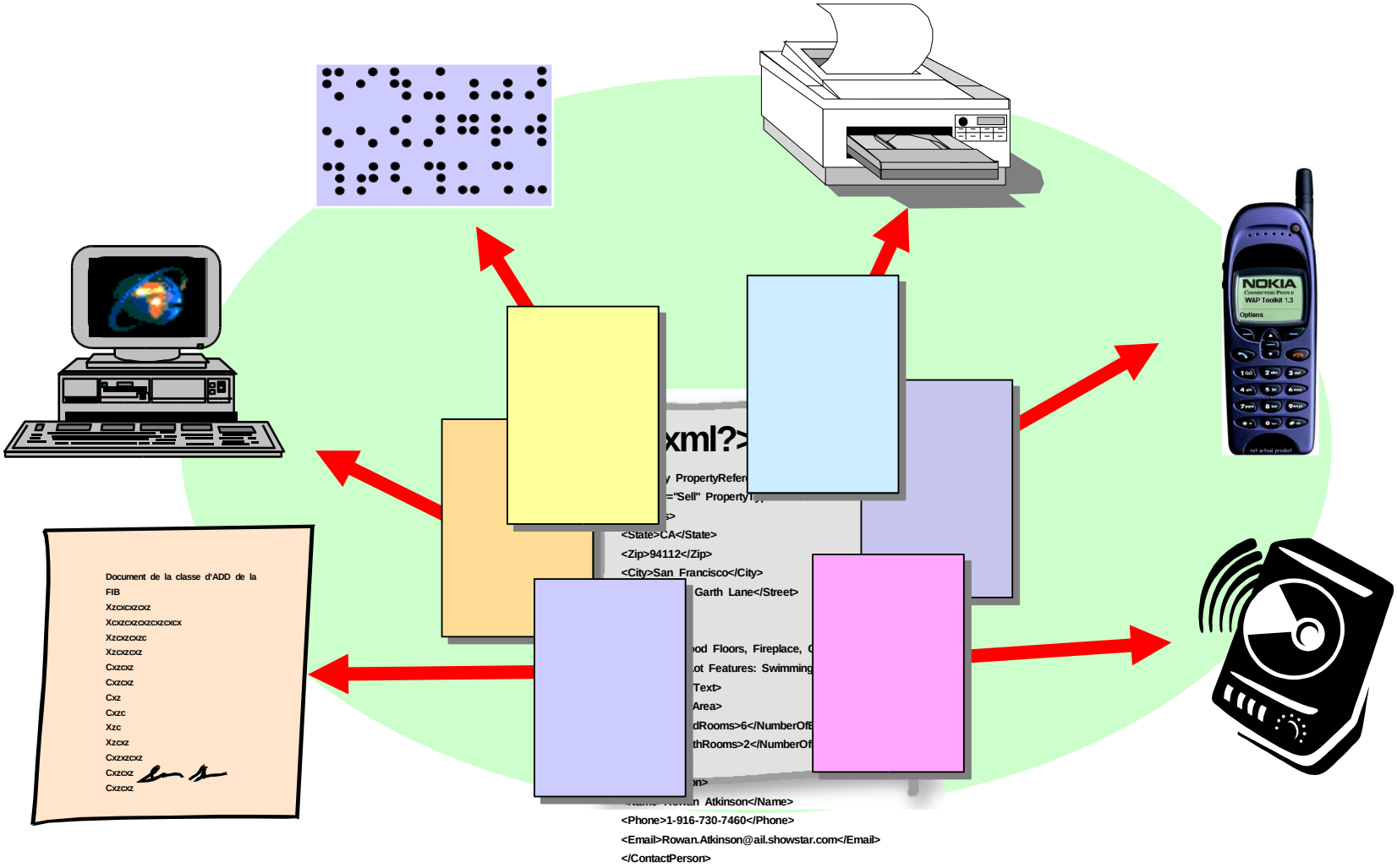
CSS stylesheet for RSS

```
rss, channel, item, title, description, link {
  display: block;
}
image, language, managingEditor, pubDate, skipDays {
  display: none;
}
channel title {
  font-family: Arial;
  font-weight: bold;
  text-align: center;
  color: blue;
  font-size: 14pt;
}
item title {
  font-family: Arial;
  font-weight: normal;
  text-align: left;
  color: black;
  font-size: 10pt;
}
```

```
[Continued]

item description {
  display: none;
}
link {
  text-decoration: underline;
  color: blue;
  margin-left: 1em;
}
```

XSL (Extensible Style Language)



XSL

- Why two Style Sheet languages?
- The fact that W3C has started developing XSL in addition to CSS has caused some confusion. Why develop a second style sheet language when implementors haven't even finished the first one? The answer can be found in the table below:

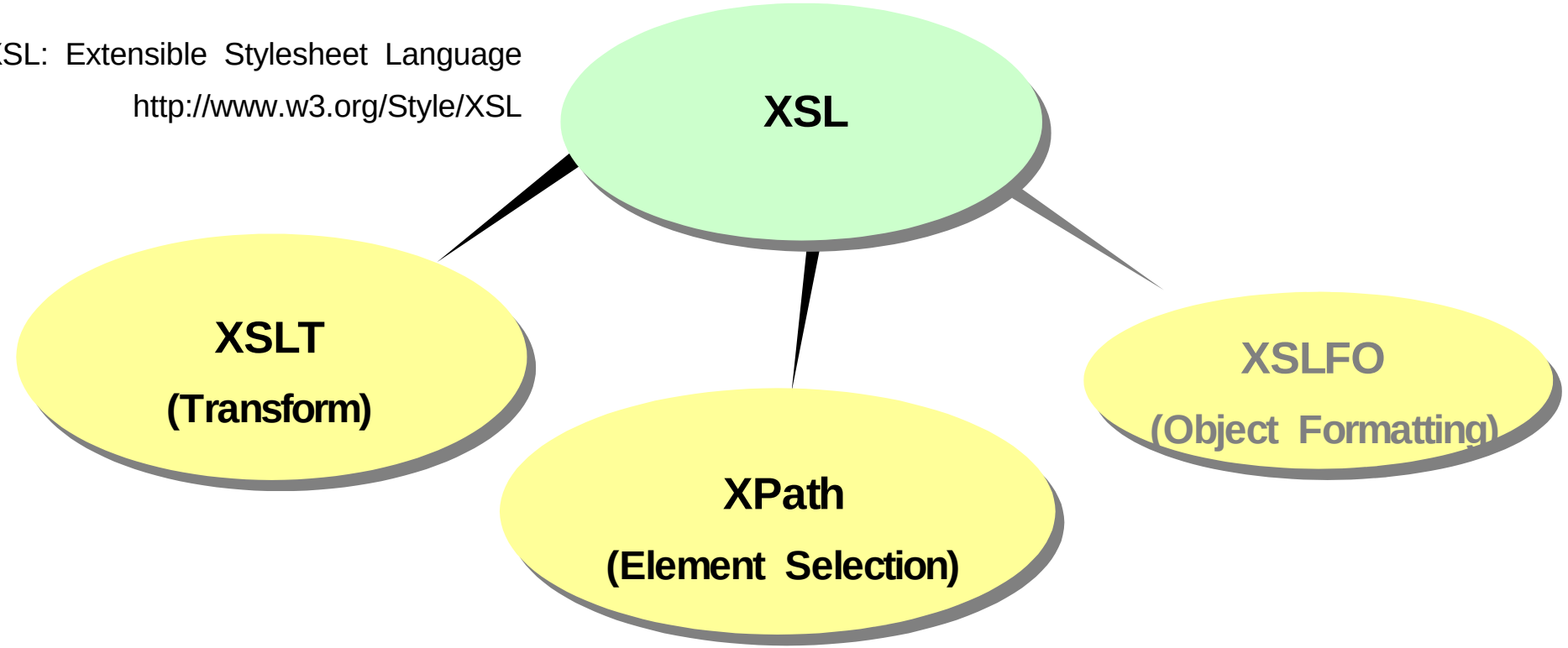
	CSS	XSL
Can be used with HTML?	Yes	No
Can be used with XML?	Yes	Yes
Transformation language?	No	Yes
Syntax	CSS	XML

- The unique features are that CSS can be used to style HTML & XML documents. XSL, on the other hand, is able to transform documents. For example, XSL can be used to transform XML data into HTML/CSS documents on the Web server. This way, the two languages complement each other and can be used together.

XSL

XSL: Extensible Stylesheet Language

http://www.w3.org/Style/XSL



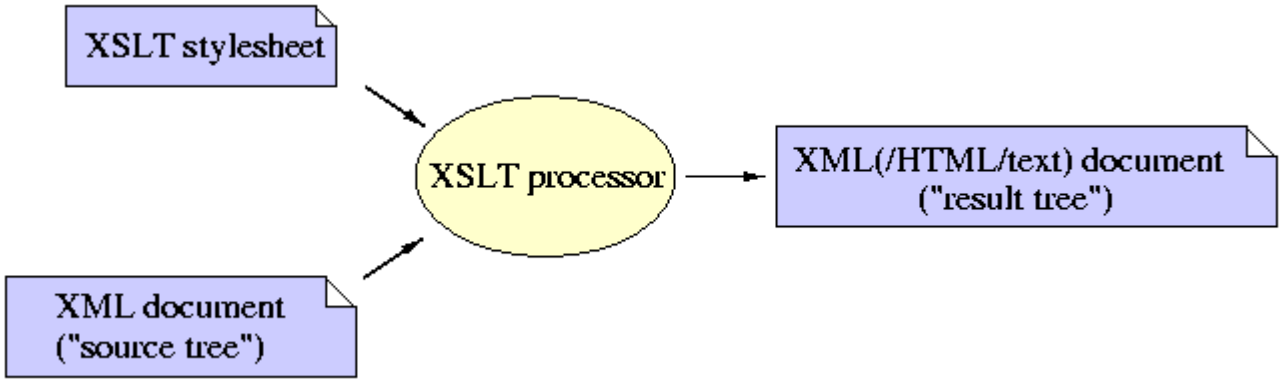
XSL standard by W3C

(XSLT and XPath) November 1999.

Complete specification in Octubre 2001.

Basics of XSL

- XSLT stylesheet:
  - Is declarative, uses pattern matching and templates for transform specification
- An easy way of describing XSL's transformation process is that it uses XSLT for transforming a XML source tree in another XML result tree.



XSLT stylesheet for RSS (.xsl)

Beginning of the Style Sheet

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0" >
  <xsl:output method="html" version="4.0" indent="yes"
              doctype-public="-//W3C//DTD HTML 4.0//EN"
              doctype-system="strict.dtd"/>
  <!-- Match the <channel> element & process all <item> children. -->
  <xsl:template match="channel">
    <HTML>
     <HEAD>
       <TITLE><xsl:value-of select="title"/></TITLE>
       <META NAME="managingEditor" CONTENT="{managingEditor}"/>
       <LINK REL="STYLESHEET" TYPE="text/css" HREF="rss-html.css"/>
     </HEAD>
    <BODY>
      <H1><xsl:value-of select="title"/></H1>
       <P><xsl:value-of select="description"/></P>
       <UL>
         <xsl:apply-templates select="item"/>
       </UL>
    </BODY></HTML>
  </xsl:template>
```

Transformation rule

XPath

```
[Continued]

  <xsl:template match="item">
    <LI> <A HREF="{link}">
     <xsl:value-of
select="title"/>
      </A> </LI>
  </xsl:template>
</xsl:stylesheet>
```

HTML generated by XSLT

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0//EN" "strict.dtd">
<HTML>
   <HEAD>
      <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
      <TITLE>UML Headlines</TITLE>
      <META NAME="managingEditor" CONTENT="editor@xmlmodeling.com">
      <LINK REL="STYLESHEET" TYPE="text/css" HREF="rss-html.css">
   </HEAD>
   <BODY>
      <H1>UML Headlines</H1>
      <P>Recent news about the Unified Modeling Language (UML).
      </P>
      <UL>
         <LI><A HREF="http://www.omg.org">UML version 1.3 adopted by the
                OMG</A></LI>
         <LI><A HREF="http://www.rational.com">Rational Rose 2000e
                released</A></LI>
         <LI><A HREF="http://www.togethersoft.com">TogetherJ 4.0
                released</A></LI>
      </UL>
   </BODY>
</HTML>
```

XPath

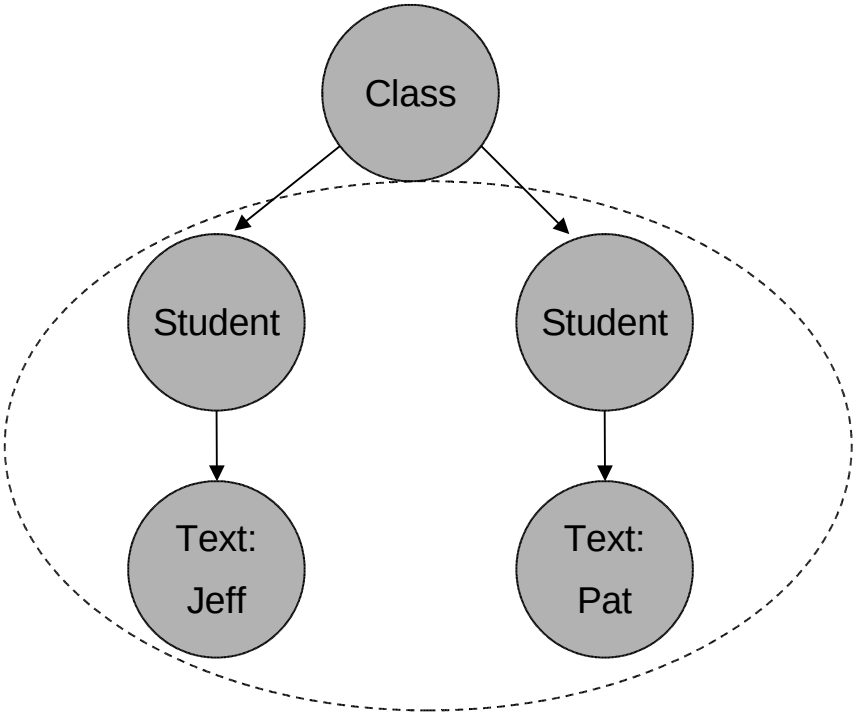
XPath: XML browsing

(XML tree can be seen as a directory tree)

XPath permits to "*select*" any node of such tree:

//Class/Student

```
<Class>
    <Student>Jeff</Student>
    <Student>Pat</Student>
</Class>
```



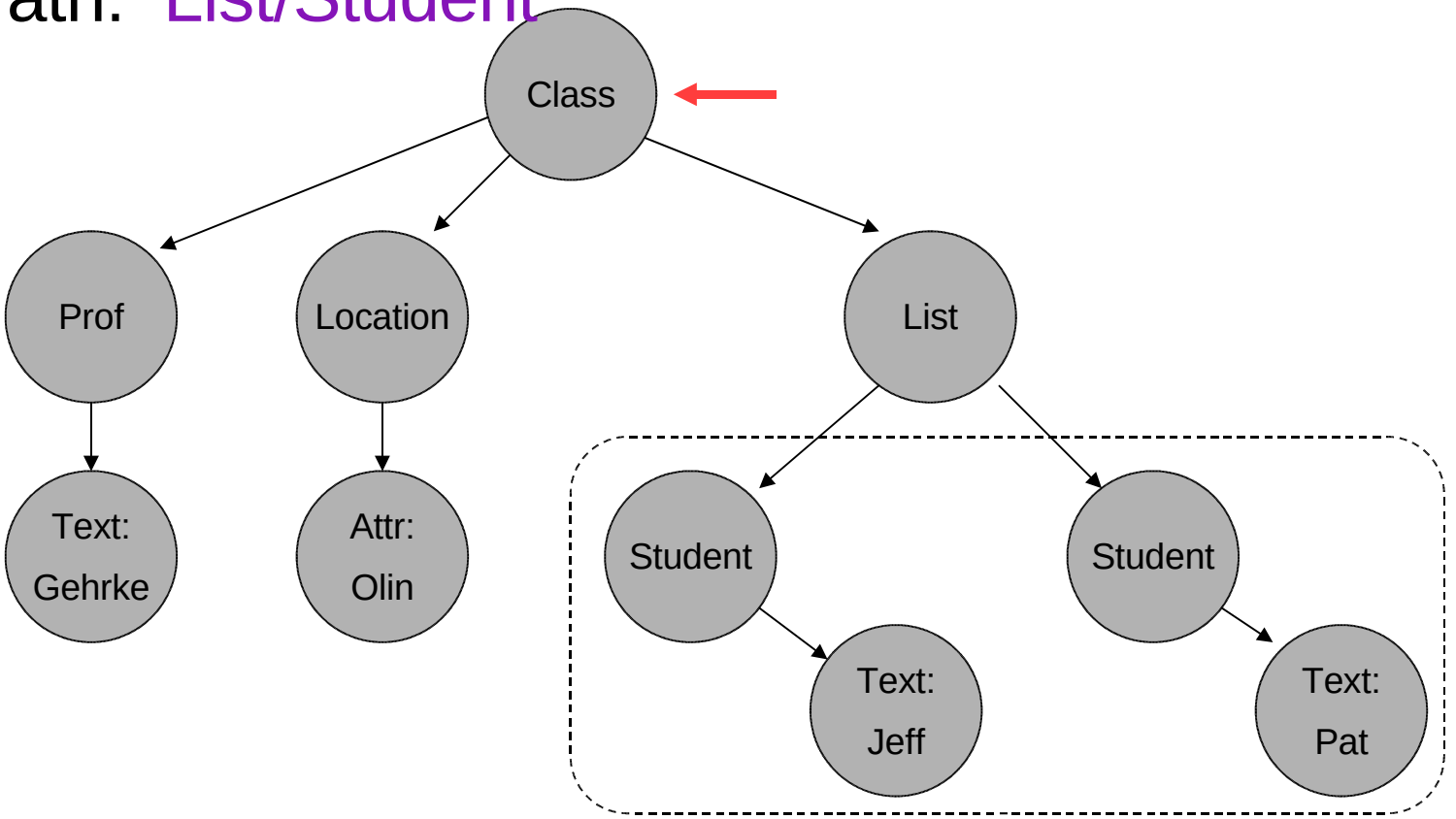
(c) slides of XPath: Jeff Derstadt

http://www.cs.cornell.edu/courses/cs433

XPath - Context

- *Context:* current working point in the XML tree.

XPath: List/Student

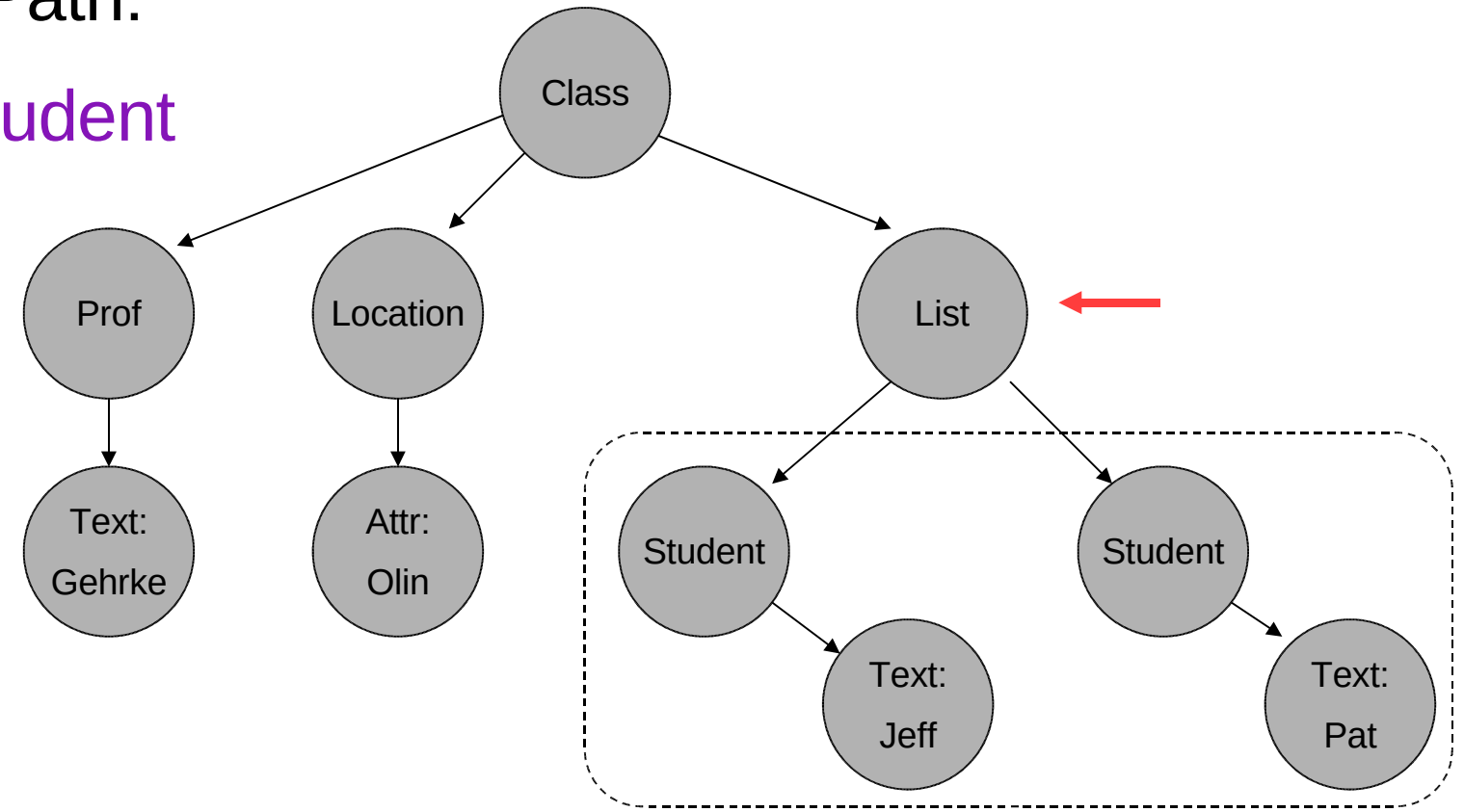


XPath - Context

- *Context:* current working point in the XML tree.

XPath:

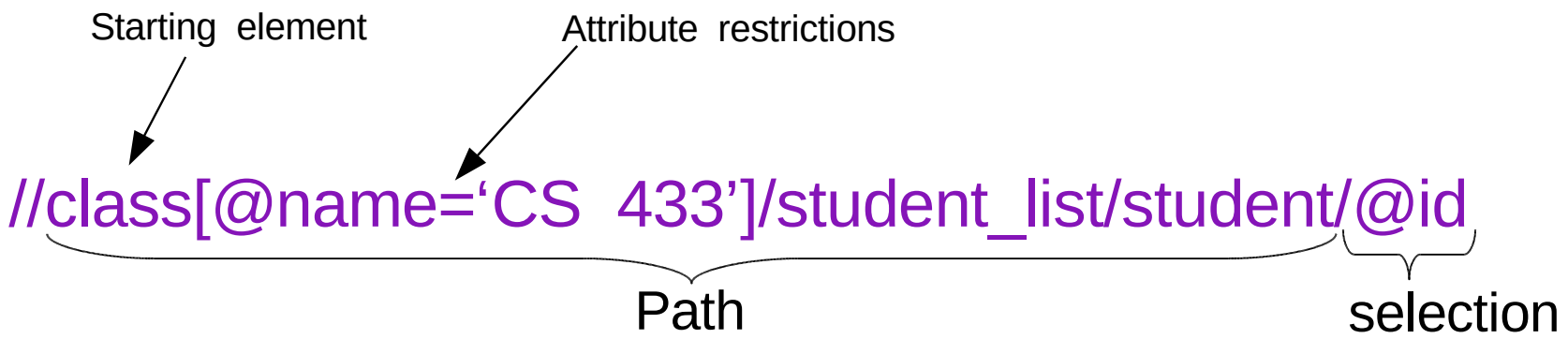
Student



XPath

- Example: Select the nodes containing the id attribute

```
<class name='CS 433'>
     <location building='Olin' room='255'/>
     <professor>Johannes Gehrke</professor>
     <ta>Dan Kifer </ta>
     <student_list>
          <student id='999-991'>John Smith</student>
          <student id='999-992'>Jane Doe</student>
     </student_list>
</class>
```



XSL Engines

- XSL in the Web:
  - Some web browsers Mozilla, I.E.
  - Server side Xalan
    - Supports preprocessing and on-the-fly
    - Java and C++ implemented by Apache XML team
- Generic XSL Transformations
  - DocBook
    - WWW
    - PDF ...

DOM and SAX

- DOM and SAX are XML parser
- An XML parser is a special software that analyzes the syntax of an XML document.
- There are two types of parsers:
  - Well-formed → Syntax
  - Valid → Given a DTD or a Schema
- DOM and SAX check either that the document is well-formed and valid.

DOM and SAX: Example

```
<?xml version="1.0"?>
<!DOCTYPE rss PUBLIC "-//Netscape Communications//DTD RSS 0.91//EN"
        "rss-0.91.dtd">
<rss version="0.91">
  <channel>
    <title>UML Headlines</title>
    <description>Recent news about the Unified Modeling Languag      L).
    </description>
    <language>en-us</language>
    <link>http://xmlmodeling.com</link>
    <managingEditor>editor@xmlmodeling.com</managingEditor>
    <skipDays>
      <day>Saturday</day><day>Sunday</day>
    </skipDays>
    <pubDate>July 1, 2000</pubDate>
    <image>
      <title>UML Headlines</title>
      <url>http://xmlmodeling.com/images/xmlmodeling.jpg</url>
      <link>http://xmlmodeling.com</link>
      <width>88</width>
      <height>31</height>
    </image>
   </image>
</channel>
</rss>
```

Check the document against this DTD to check if it is valid

The document is **not** *well-formed*

DOM and SAX

- A parser is not used only to check if a XML document is either well-formed or valid.
- The parser will need to read the entire XML document, it is also used to process and filter it.
- Using DOM and SAX you can process an XML document

DOM

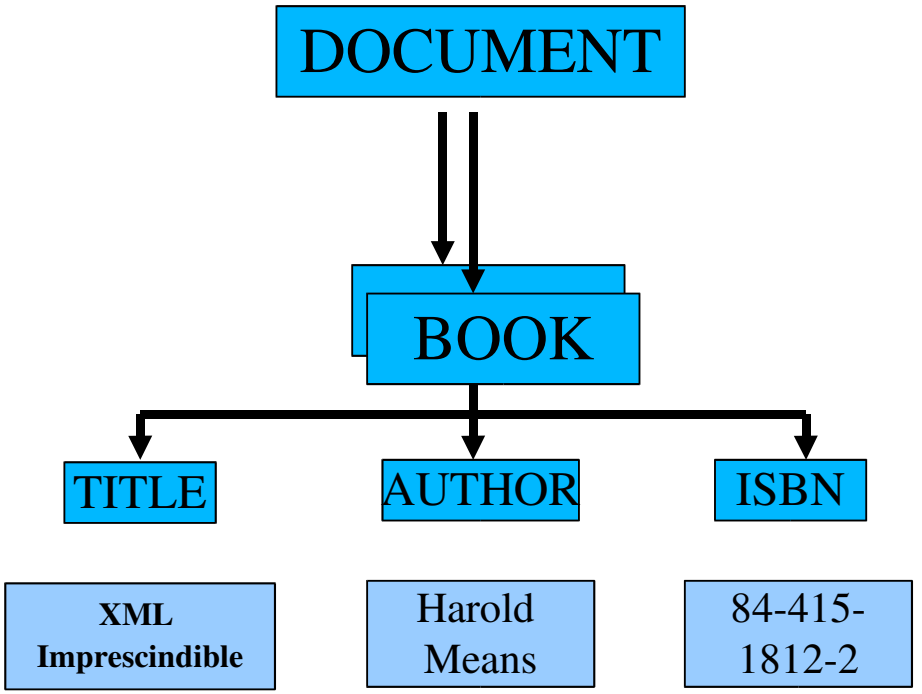
- DOM stands for **D**ocument **O**bject **M**odel
- DOM Provides a standard interface to process XML documents.
- DOM represents the XML document as a **tree**
- DOM is multi-platform
  - In java

```
import org.w3c.dom.*
import org.apache.werces.parsers.DOMParser;
```

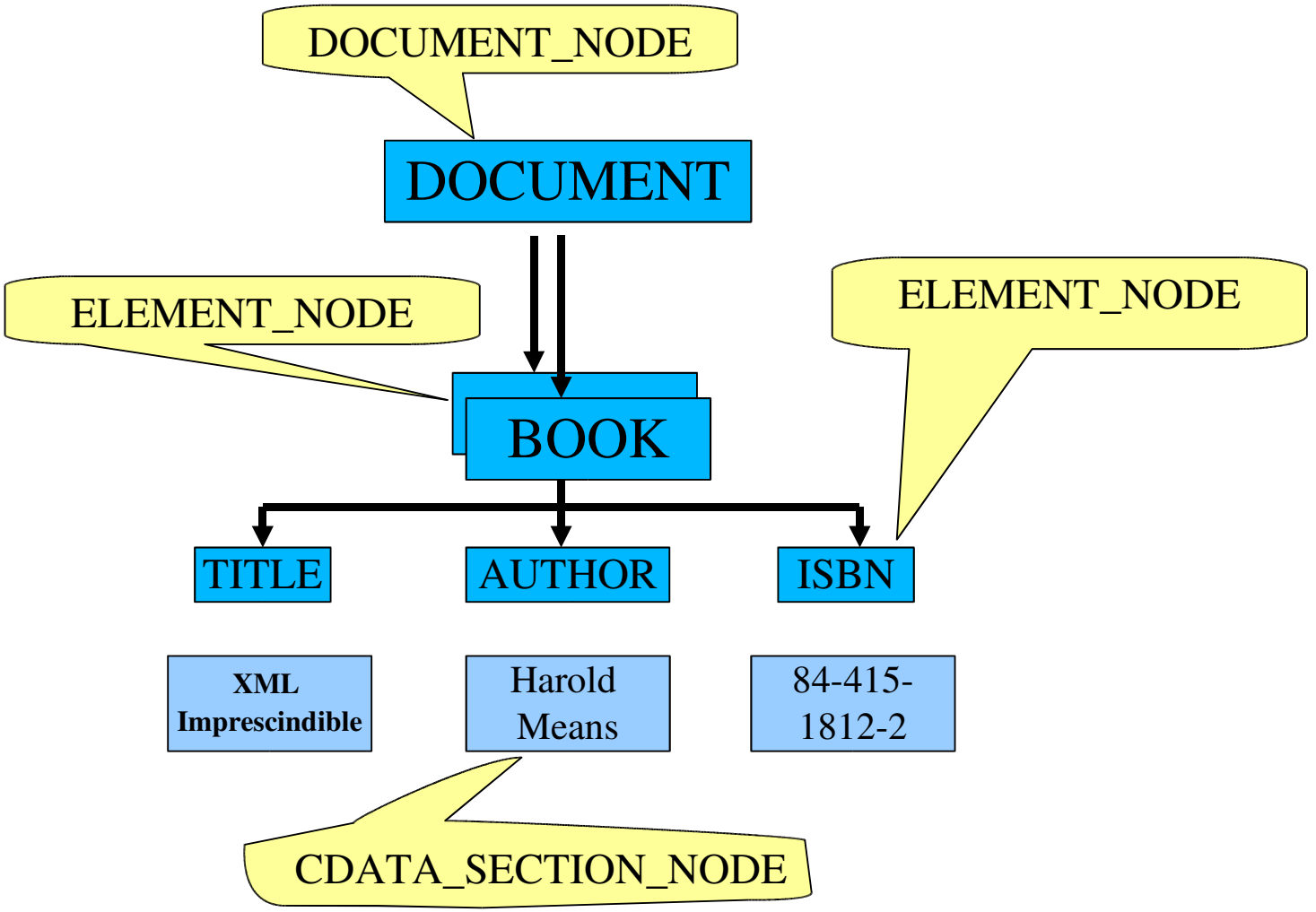
- DOM is a W3C recomendation (October 1998)

DOM

```
<?xml version="1.0“ standalone=“yes”?>
<DOCUMENT>
    <BOOK>
          <TITLE>
               XML Imprescindible
          </TITLE>
          <AUTHOR>
               Harold Means
          </AUTHOR>
          <ISBN> 84-415-1812-2 </ISBN>
    </BOOK>
    <BOOK>
          <TITLE>
               Developing Enterprise Web Services
          </TITLE>
          <AUTHOR>
               Sandeep Chatterjee
          </AUTHOR>
          <AUTHOR>
               James Webber
          </AUTHOR>
          <ISBN> 85-435-1411-4 </ISBN>
    </BOOK>
</DOCUMENT>
```



DOM


DOCUMENT_NODE
DOCUMENT
ELEMENT_NODE
ELEMENT_NODE
BOOK
TITLE
AUTHOR
ISBN
XML Imprescindible
Harold Means
84-415-1812-2
CDATA_SECTION_NODE

```
import org.w3c.dom.*;
import org.apache.xerces.parsers.DOMParser;

public class XML_Parser
{
public static void main(String[] args)
{
try {
    DOMParser parser= new DOMParser();
    parser.parse(argv[0]);
    Document doc = parser.getDocument();
    display(document);
    }
catch (Exception e) {e.printStackTrace
    (System.err)}
}

public static void display(Node node)
{
if (node==null) return null;
int type = node.getNodeType();
switch (type) {
 case Node.DOCUMENT_NODE: {
 display(((Document)node).getDocumentElement());
 break;}

 case Node.ELEMENT_NODE:
 NodeList childNodes = node.getChildNodes();
 if (childNodes != null) {
    length=childNodes.getLength();
    for(i=0;i<length;i++)
         display(childNodes.item(i));}
    break;}

Case Node.CDATA_SECTION_NODE: {
    // Print values
    break;}
}}
```

Create a DOMParser

Parse the document

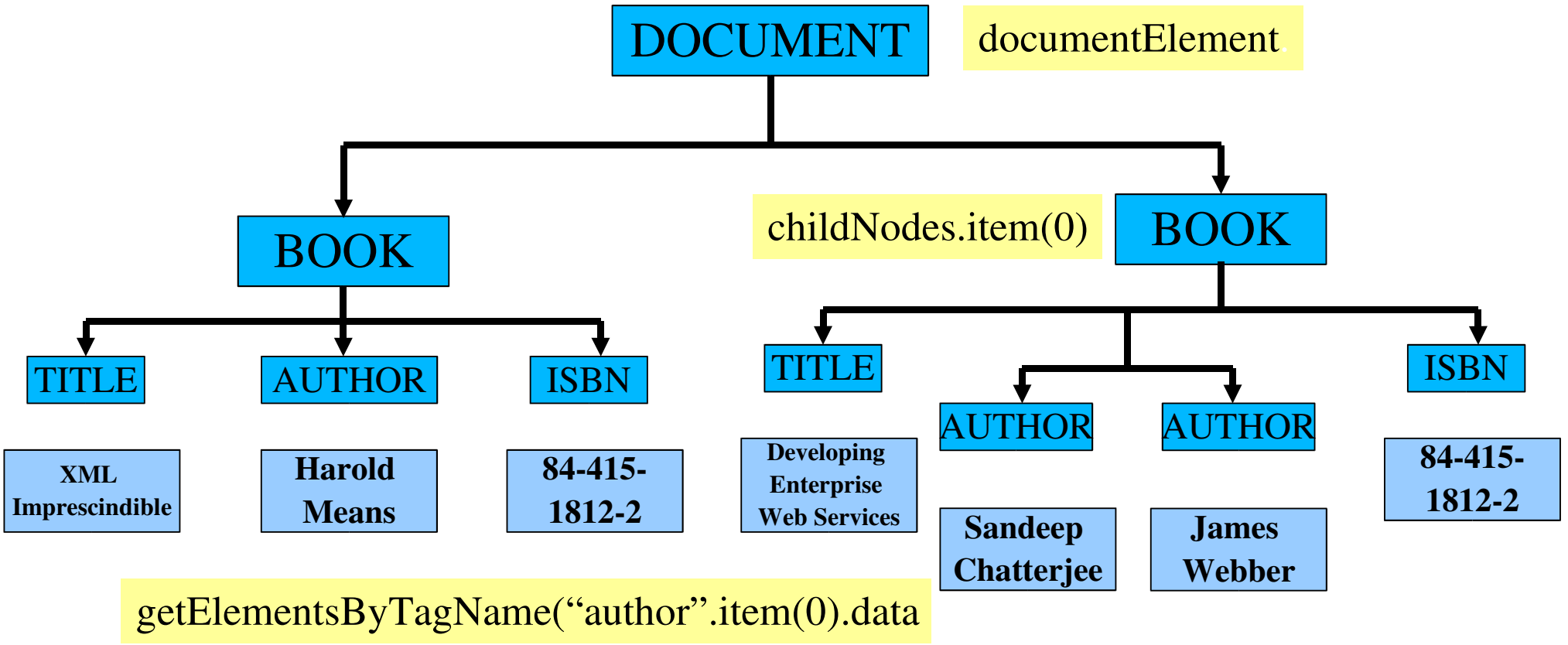
Get a Document object type

If the document is **not** *valid* or *well_formed*

For each child, call the *display* function (recursive)

DOM

doc.



getElementsByTagName("author".item(0).data

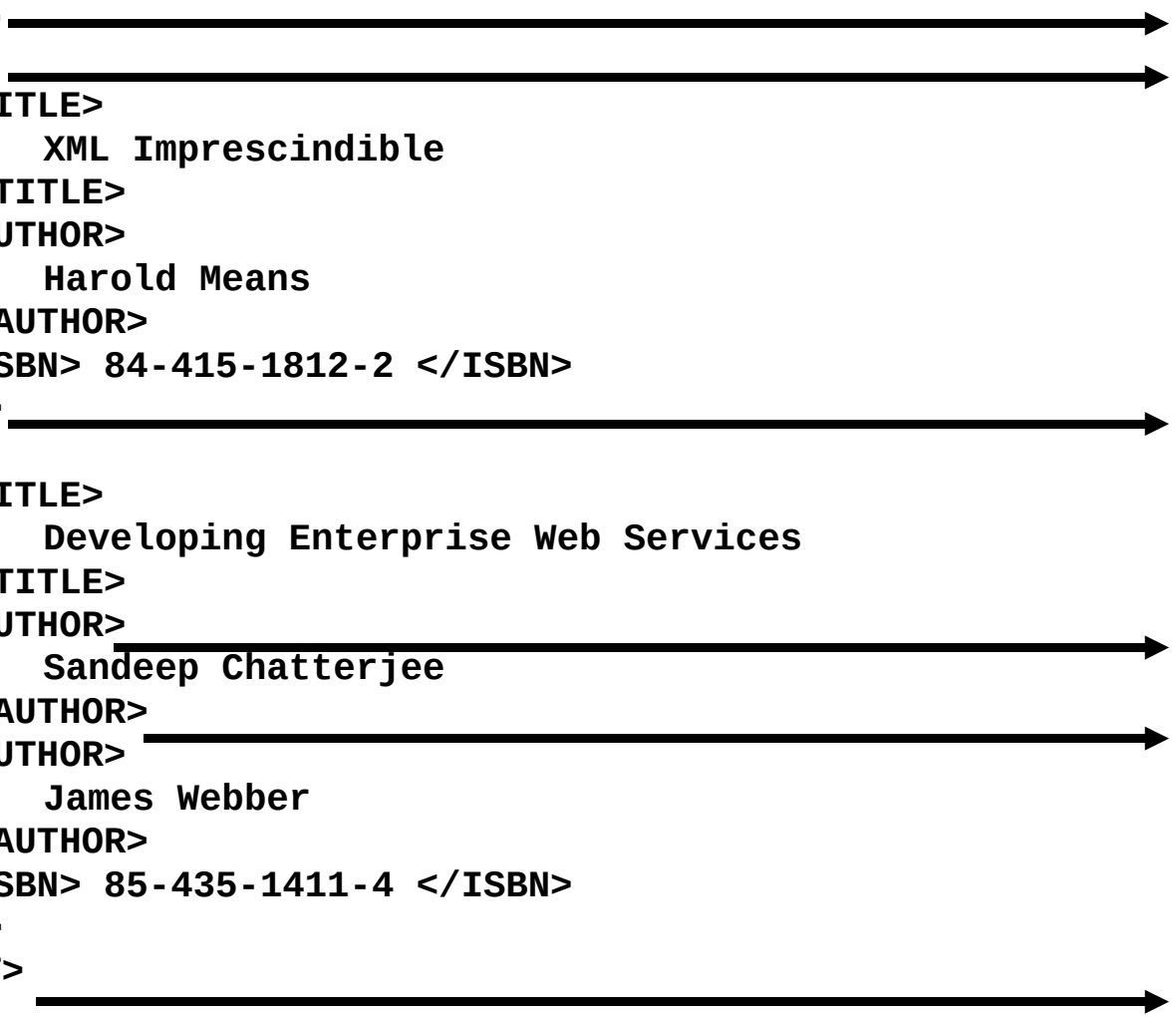
```
doc.documentElement.childNodes.item(0).getElementsByTagName("author").item(0).data
```

SAX

- SAX stands for **S**imple **A**PI for **X**ML
- Rather than having to navigate through the whole document, let the document came to you
  - The document is parsed in a event-based process
- SAX is multi-platform
- Developed by the XML-DEV mailing lists in May 1998

SAX

```
<?xml version="1.0“ standalone=“yes”?>
<DOCUMENT>
   <BOOK>
        <TITLE>
            XML Imprescindible
        </TITLE>
        <AUTHOR>
            Harold Means
        </AUTHOR>
        <ISBN> 84-415-1812-2 </ISBN>
   </BOOK>
   <BOOK>
        <TITLE>
            Developing Enterprise Web Services
        </TITLE>
        <AUTHOR>
            Sandeep Chatterjee
        </AUTHOR>
        <AUTHOR>
            James Webber
        </AUTHOR>
        <ISBN> 85-435-1411-4 </ISBN>
   </BOOK>
</DOCUMENT>
```



SAX

```
import org.xml.sax.*;
import org.xml.sax.helpers.DeafultHandler;
import org.apache.xerces.parsers.SAXParser;

public class XML_Parser extends DefaultHandler
{
int BookCount=0;

public void startElement(String uri, String localName String rawName, Attributes atr) {
    if rawName.equals(“AUTOR“) BookCount++;
    }

public static void main(String[] args)
{
    try {
        FirstParserSAX SAXHandler = new FirstParserSAX();

        SAXParser parser = new SAXParser();

        parser.setContentHandler(SAXHandler);
        parser.setErrorHandler(SAXHandler);
        parser.parse(argv[0]);

        }
        catch (Exception e) {
              e.printStackTrace(System.err);
    }
}
```