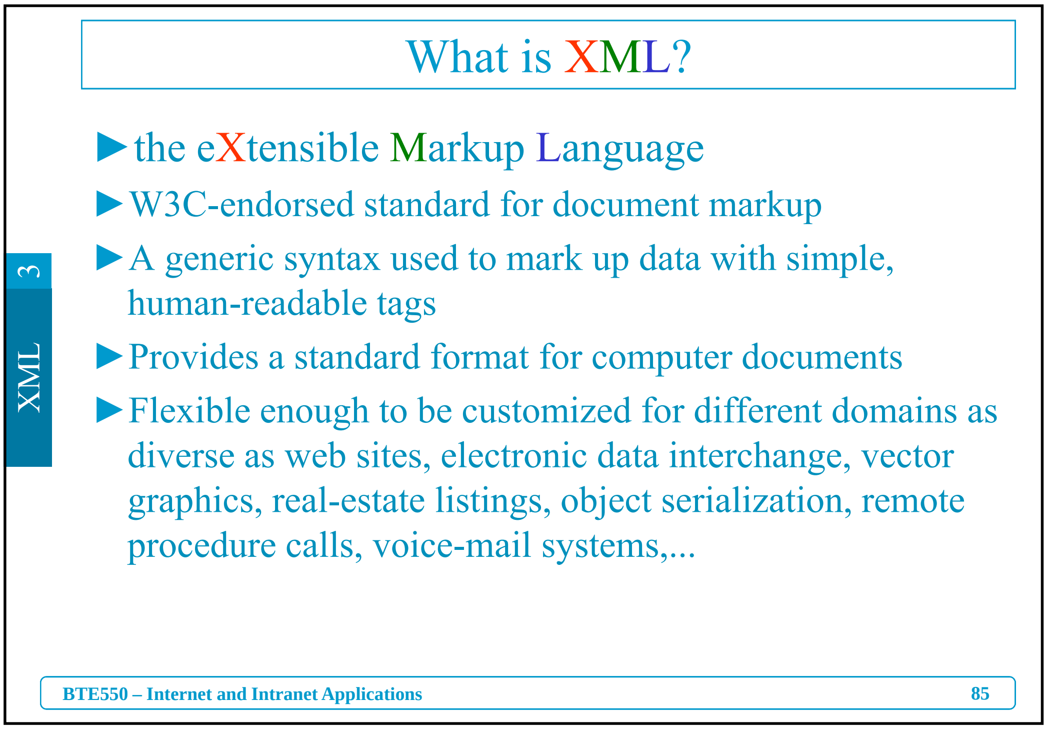
3 Overview of XML

Content

- ▶Define XML
- ▶Compare and contrast HTML and XML
- ▶Identify characteristics of XML documents

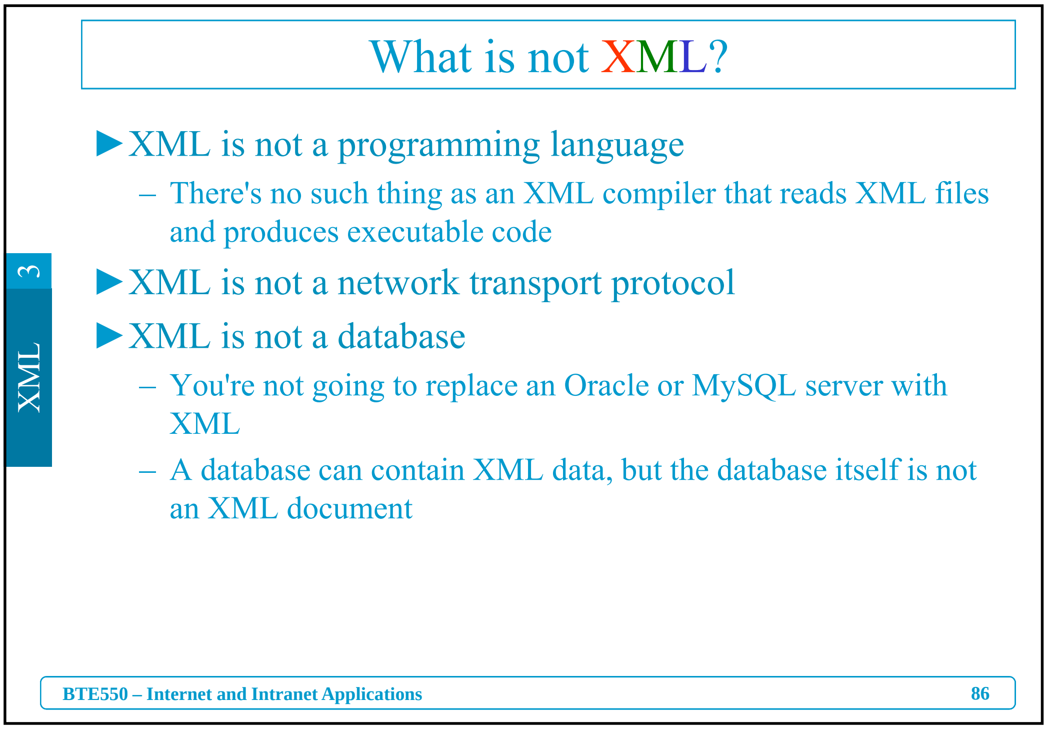


3

XML

What is XML?

- ▶ the eXtensible Markup Language
- ▶ W3C-endorsed standard for document markup
- ▶ A generic syntax used to mark up data with simple, human-readable tags
- ▶ Provides a standard format for computer documents
- ▶ Flexible enough to be customized for different domains as diverse as web sites, electronic data interchange, vector graphics, real-estate listings, object serialization, remote procedure calls, voice-mail systems,...

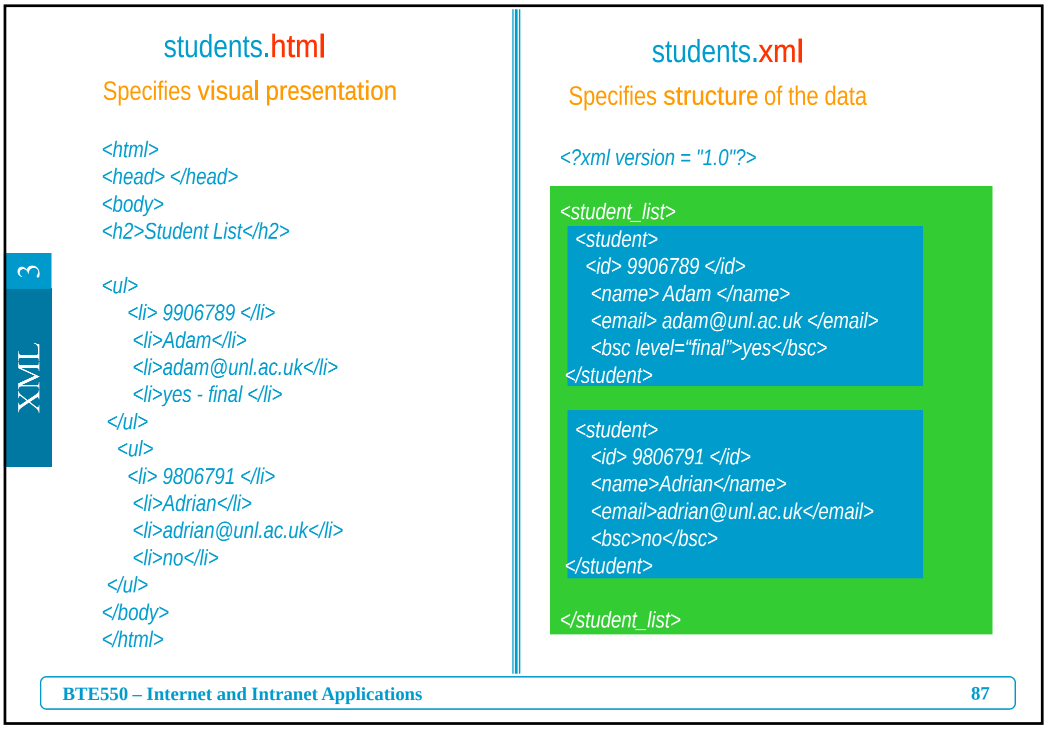


3

XML

What is not XML?

- ▶ XML is not a programming language
  - – There's no such thing as an XML compiler that reads XML files and produces executable code
- ▶ XML is not a network transport protocol
- ▶ XML is not a database
  - – You're not going to replace an Oracle or MySQL server with XML
  - – A database can contain XML data, but the database itself is not an XML document



students.html

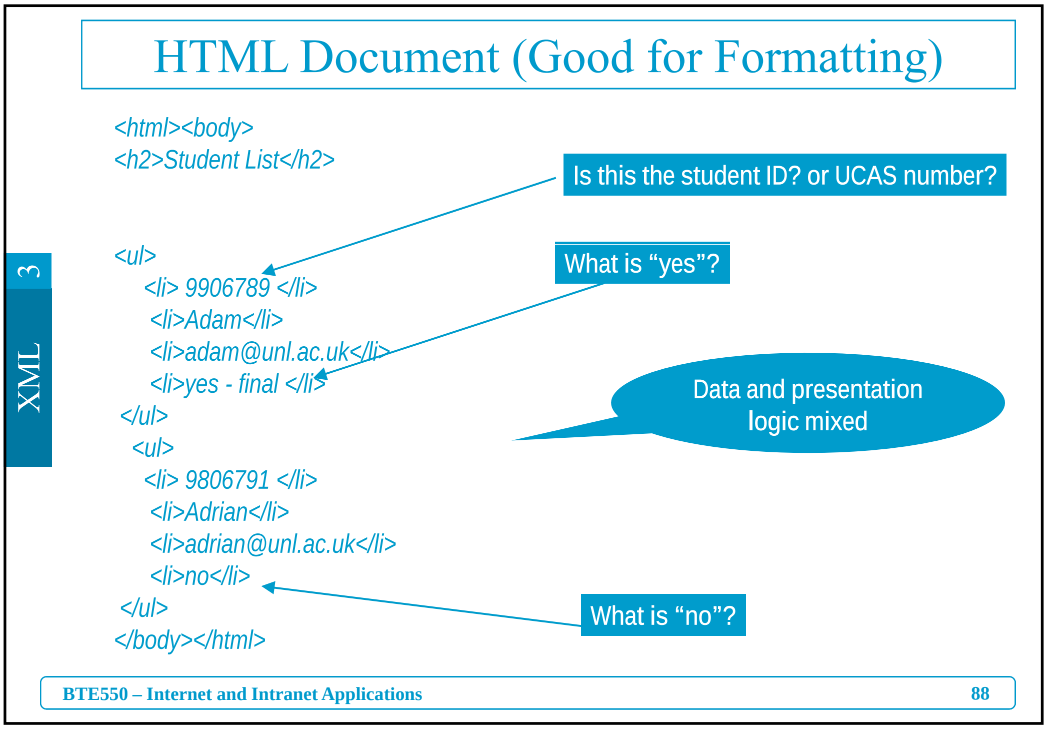
Specifies visual presentation

```
<html>
<head> </head>
<body>
<h2>Student List</h2>

<ul>
    <li> 9906789 </li>
     <li>Adam</li>
     <li>adam@unl.ac.uk</li>
     <li>yes - final </li>
 </ul>
  <ul>
    <li> 9806791 </li>
     <li>Adrian</li>
     <li>adrian@unl.ac.uk</li>
     <li>no</li>
 </ul>
</body>
</html>
```

students.xml

Specifies structure of the data

```
<?xml version = "1.0"?>

<student_list>
  <student>
    <id> 9906789 </id>
     <name> Adam </name>
     <email> adam@unl.ac.uk </email>
     <bsc level=“final”>yes</bsc>
 </student>

  <student>
     <id> 9806791 </id>
     <name>Adrian</name>
     <email>adrian@unl.ac.uk</email>
     <bsc>no</bsc>
 </student>

</student_list>
```



HTML Document (Good for Formatting)

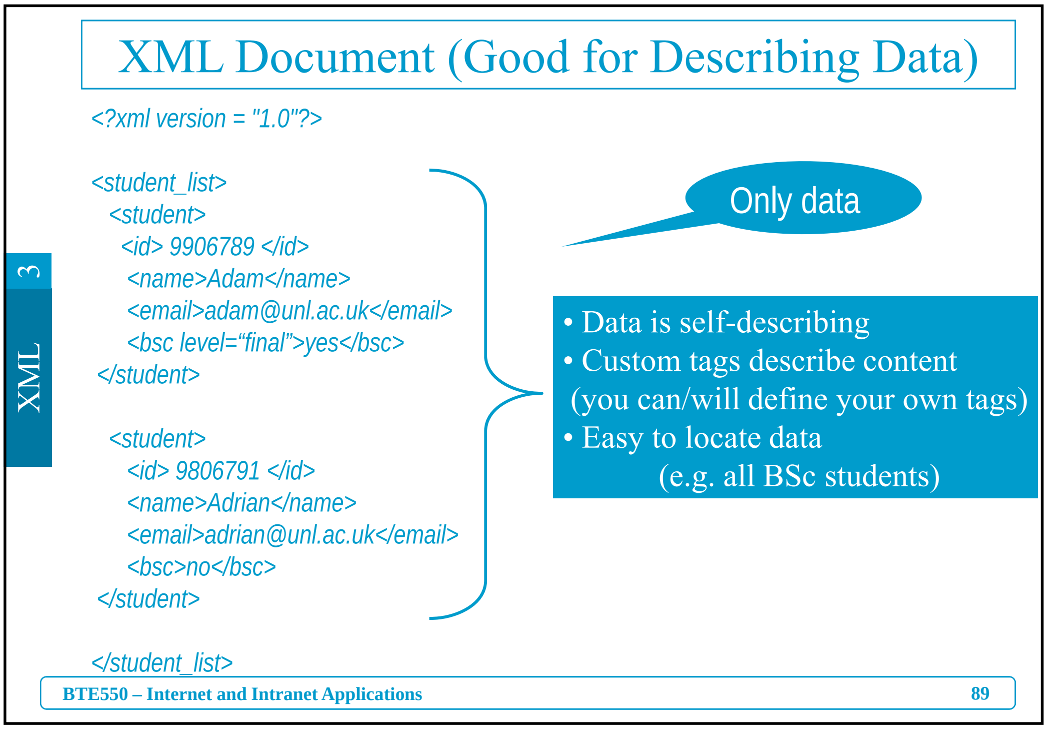
```
<html><body>
<h2>Student List</h2>


<ul>
    <li> 9906789 </li>
     <li>Adam</li>
     <li>adam@unl.ac.uk</li>
     <li>yes - final </li>
 </ul>
  <ul>
    <li> 9806791 </li>
     <li>Adrian</li>
     <li>adrian@unl.ac.uk</li>
     <li>no</li>
 </ul>
</body></html>
```

Is this the student ID? or UCAS number?

What is “yes”?

Data and presentation logic mixed

What is “no”?



XML Document (Good for Describing Data)

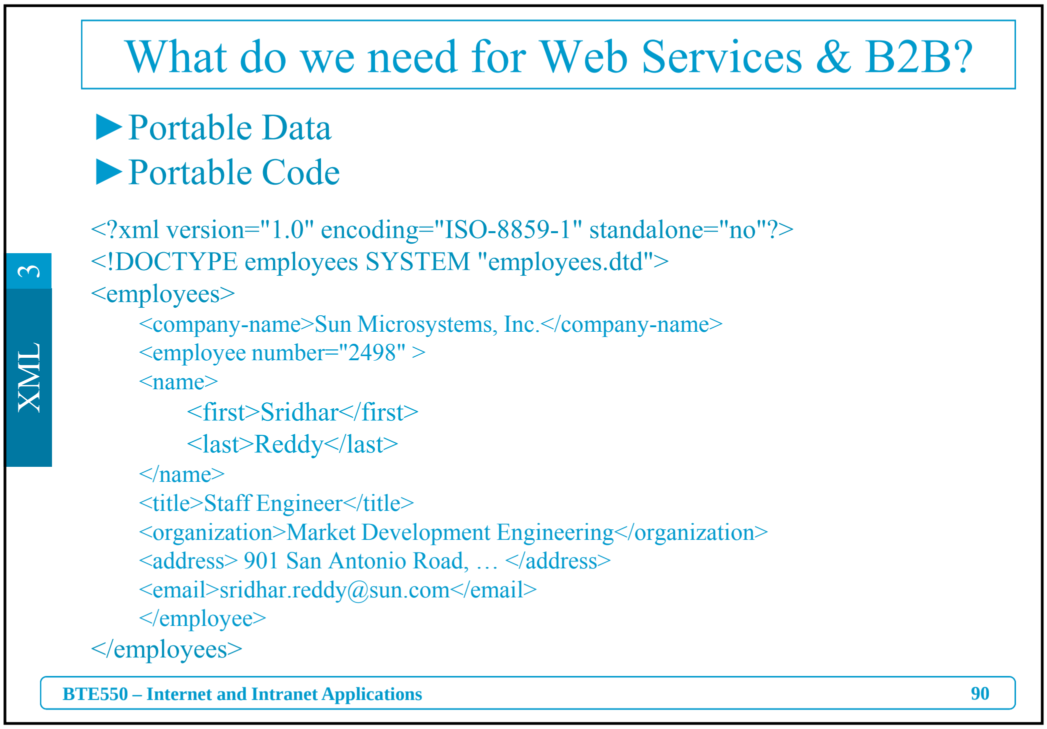
3

XML

```
<?xml version = "1.0"?>

<student_list>
  <student>
    <id> 9906789 </id>
     <name>Adam</name>
     <email>adam@unl.ac.uk</email>
     <bsc level="final">yes</bsc>
 </student>

  <student>
     <id> 9806791 </id>
     <name>Adrian</name>
     <email>adrian@unl.ac.uk</email>
     <bsc>no</bsc>
 </student>

</student_list>
```

Only data

- Data is self-describing
- Custom tags describe content (you can/will define your own tags)
- Easy to locate data (e.g. all BSc students)



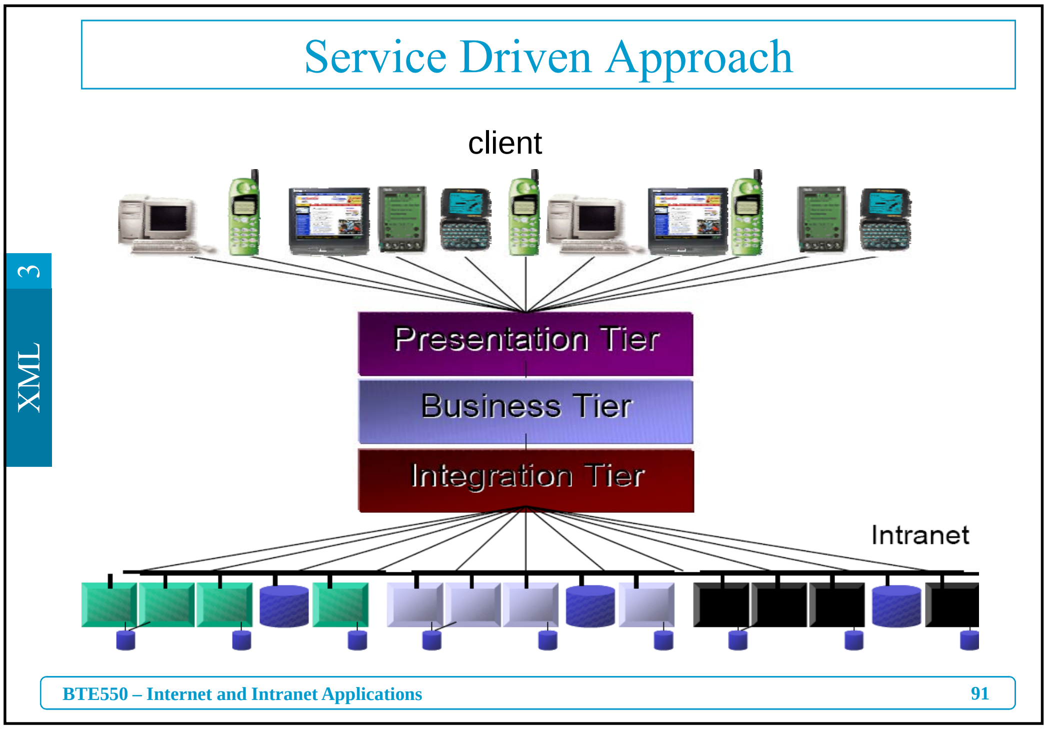
What do we need for Web Services & B2B?

3

XML

- ▶ Portable Data
- ▶ Portable Code

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE employees SYSTEM "employees.dtd">
<employees>
      <company-name>Sun Microsystems, Inc.</company-name>
      <employee number="2498" >
      <name>
            <first>Sridhar</first>
            <last>Reddy</last>
      </name>
      <title>Staff Engineer</title>
      <organization>Market Development Engineering</organization>
      <address> 901 San Antonio Road, … </address>
      <email>sridhar.reddy@sun.com</email>
      </employee>
</employees>
```



Service Driven Approach

3 XML

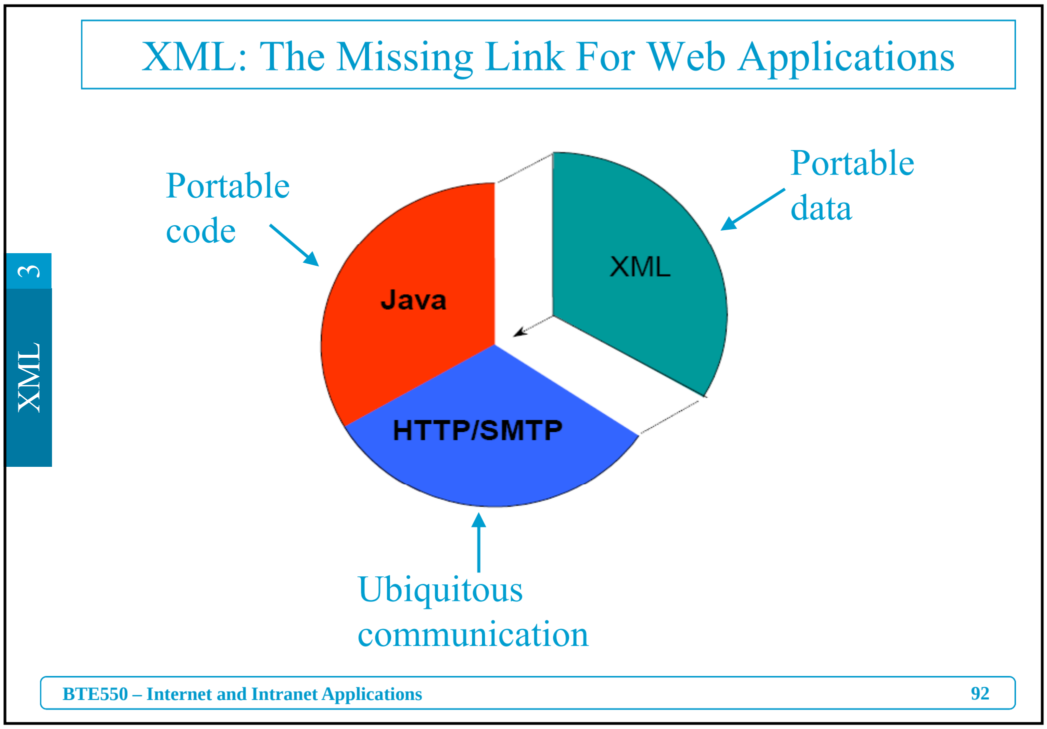
client

Presentation Tier

Business Tier

Integration Tier

Intranet



XML: The Missing Link For Web Applications

3 XML

Portable code

Portable data

Java

XML

HTTP/SMTP

Ubiquitous communication

XML

- Portable data
- Works anywhere
- Lingua franca of the Internet
- Multiple vendors
- Open development process:
  - World Wide Web Consortium (W3C)

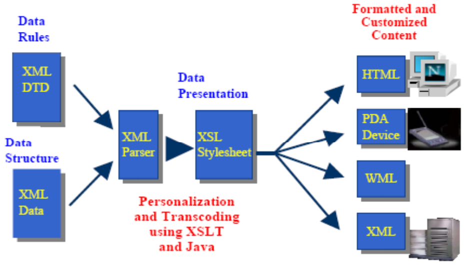
Java and XML: Symbiotic Relationship

- It's a "Match made in Heaven"
  - Java enables Portable Code
  - XML enables Portable Data
- XML tools and programs are mostly written in the Java programming language
- Better API support for Java platform than any other language
- Two great tastes that taste great together

Two Viewpoints of XML

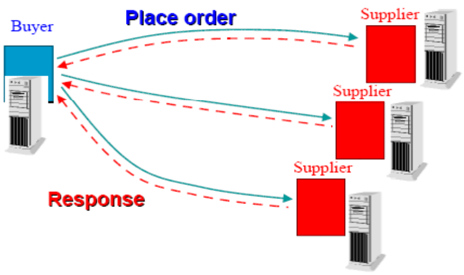
- Presentation Oriented Publishing (POP)
  - Useful for Browsers and Editors
  - Usually used for data that will be consumed by Humans
- Message Oriented Middleware (MOM)
  - Useful for Machine-to-Machine data exchange
  - Business-to-Business communication an excellent example

POP - MOM



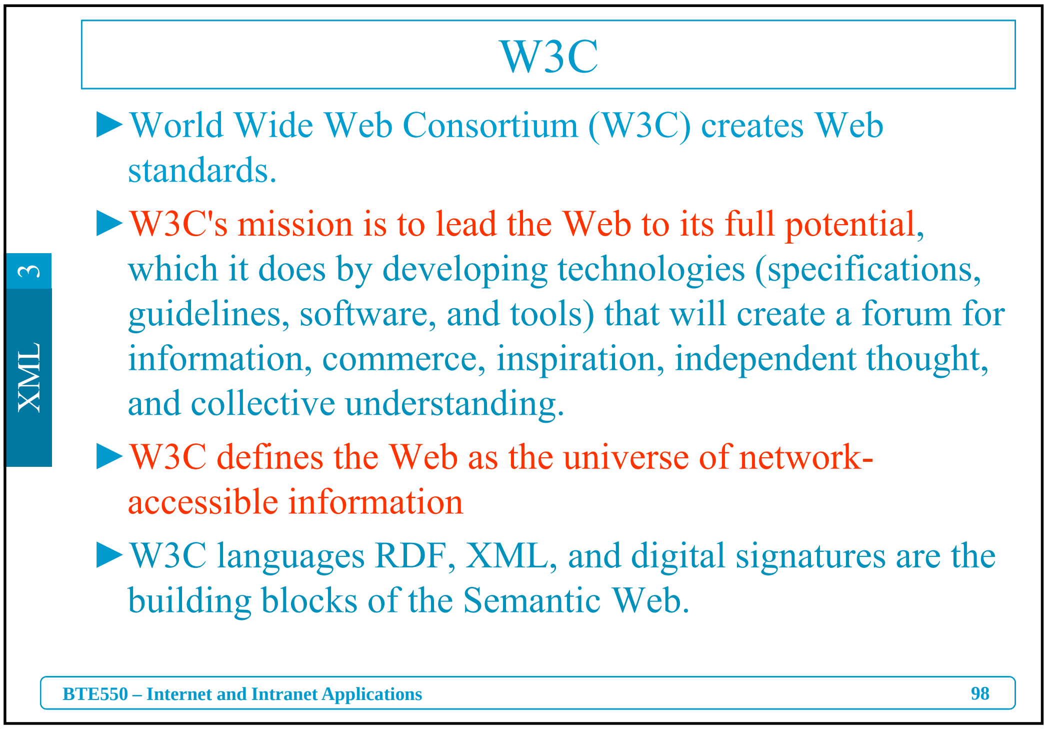
Presentation Oriented Publishing (POP)

Message Oriented Middleware (MOM)



Standardization Activities

- XML Standards
  - Through Standard organizations
  - W3C, IETF, OASIS, UN/CEFACT



W3C

- World Wide Web Consortium (W3C) creates Web standards.
- W3C's mission is to lead the Web to its full potential, which it does by developing technologies (specifications, guidelines, software, and tools) that will create a forum for information, commerce, inspiration, independent thought, and collective understanding.
- W3C defines the Web as the universe of network-accessible information
- W3C languages RDF, XML, and digital signatures are the building blocks of the Semantic Web.

XML

3

XML

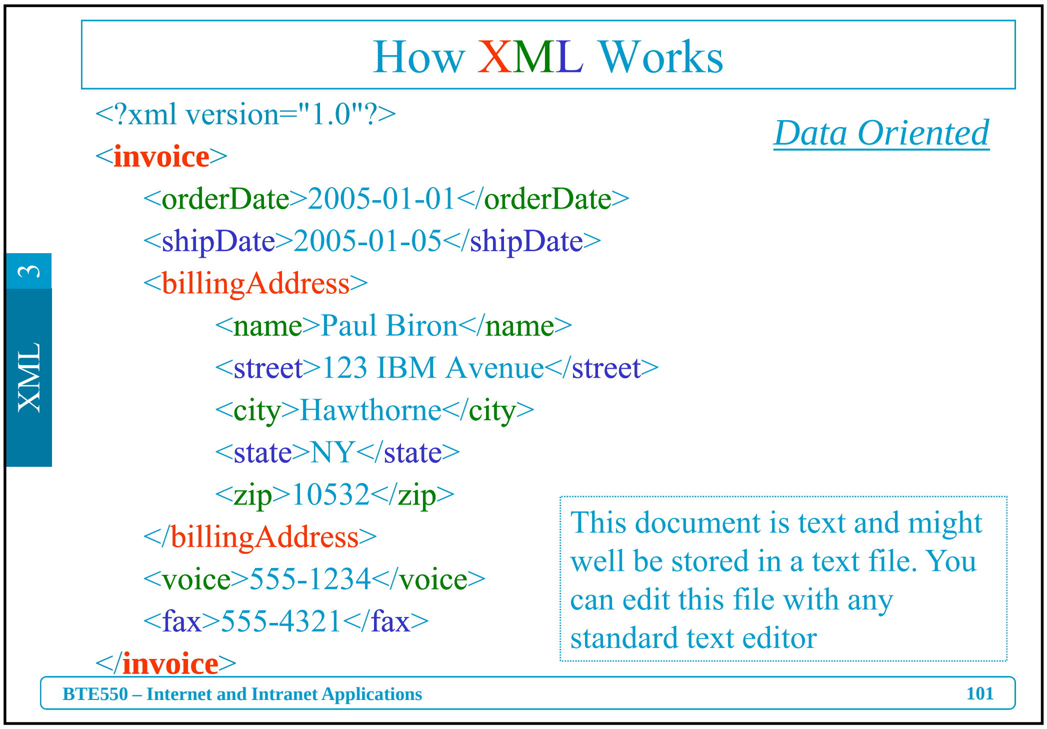
- XML is an extremely flexible format for data
- In theory, any data that can be stored in a computer can be stored in XML format.
- In practice, XML is suitable for storing and exchanging any data that can plausibly be encoded as text.
- Unsuitable for multimedia data such as photographs, recorded sound, video, and other very large bit sequences

XML

3

XML

- The eXtensible Markup Language (XML) is the universal format for structured documents and data on the Web
- XML is a text-based markup language.
- As with HTML, you identify data using tags (identifiers enclosed in angle brackets, like this: <...>).
- Collectively, the tags are known as "markup".
- But unlike HTML, XML tags tell you what the data means, rather than how to display it.

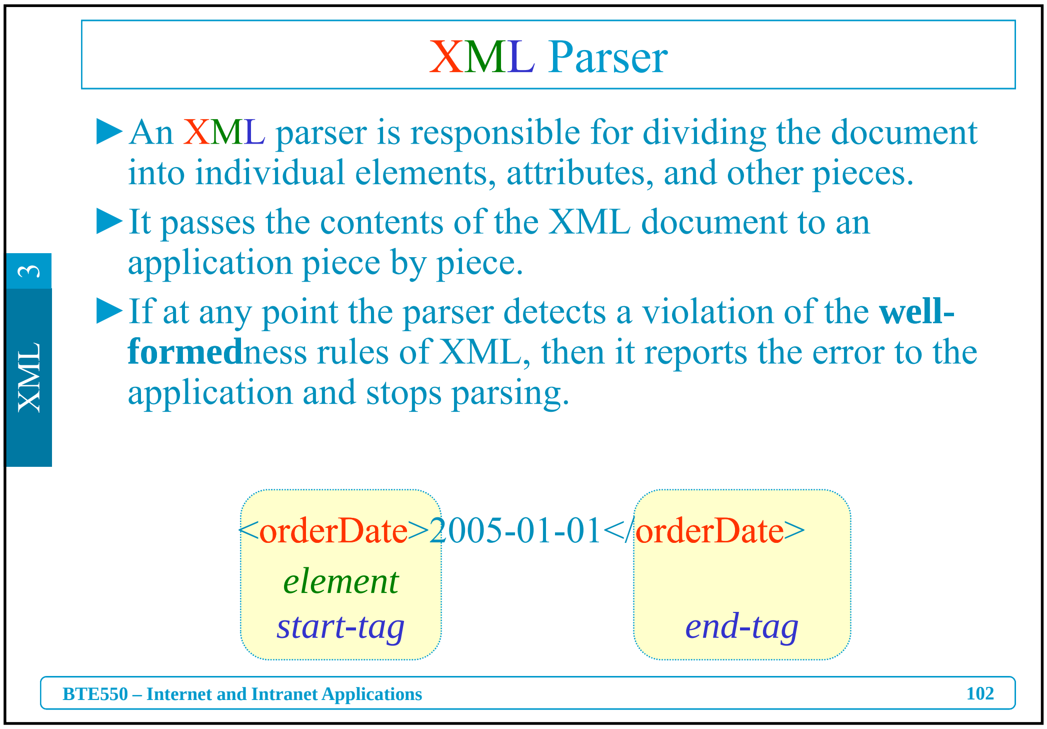


How XML Works

Data Oriented

```
<?xml version="1.0"?>
<invoice>
    <orderDate>2005-01-01</orderDate>
    <shipDate>2005-01-05</shipDate>
    <billingAddress>
        <name>Paul Biron</name>
        <street>123 IBM Avenue</street>
        <city>Hawthorne</city>
        <state>NY</state>
        <zip>10532</zip>
    </billingAddress>
    <voice>555-1234</voice>
    <fax>555-4321</fax>
</invoice>
```

This document is text and might well be stored in a text file. You can edit this file with any standard text editor



XML Parser

- ▶ An XML parser is responsible for dividing the document into individual elements, attributes, and other pieces.
- ▶ It passes the contents of the XML document to an application piece by piece.
- ▶ If at any point the parser detects a violation of the **well-formed**ness rules of XML, then it reports the error to the application and stops parsing.

<orderDate>2005-01-01</orderDate>

element

start-tag

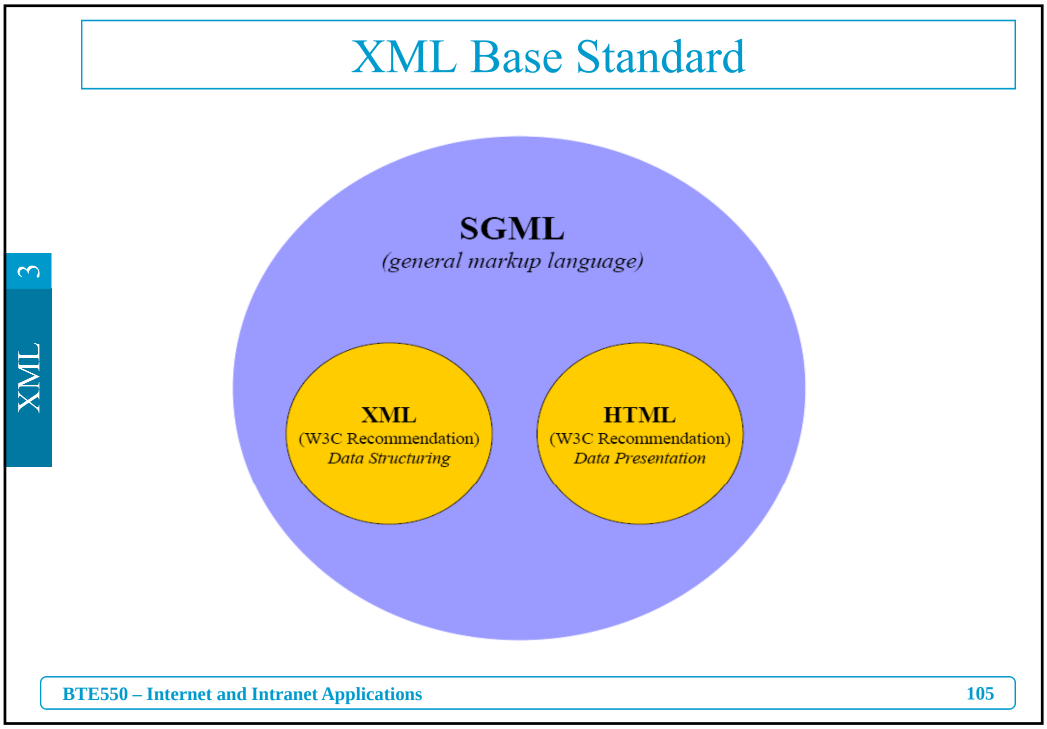
end-tag

XML Parser (Con’t)

- Individual XML applications normally dictate more precise rules about exactly which elements and attributes are allowed where
  - DTD, XML Schema
- Some XML parsers compare the document to its schema as they read it to see if the document satisfies the constraints specified there
- Such a parser is called a validating parser
- Checking a document against a schema is called **validation**
- Not all parsers are validating parsers. Some merely check for well-formedness

The Evolution of XML

- XML is a descendant of SGML, the Standard Generalized Markup Language
- SGML was invented by Charles F. Goldfarb, Ed Mosher, and Ray Lorie at IBM in the 1970s
- Became ISO standard 8879 in 1986
- It is a semantic and structural markup language for text documents
- Achieved some success in the U.S. military and government, in the aerospace sector
- SGML's biggest success was HTML, which is an SGML application

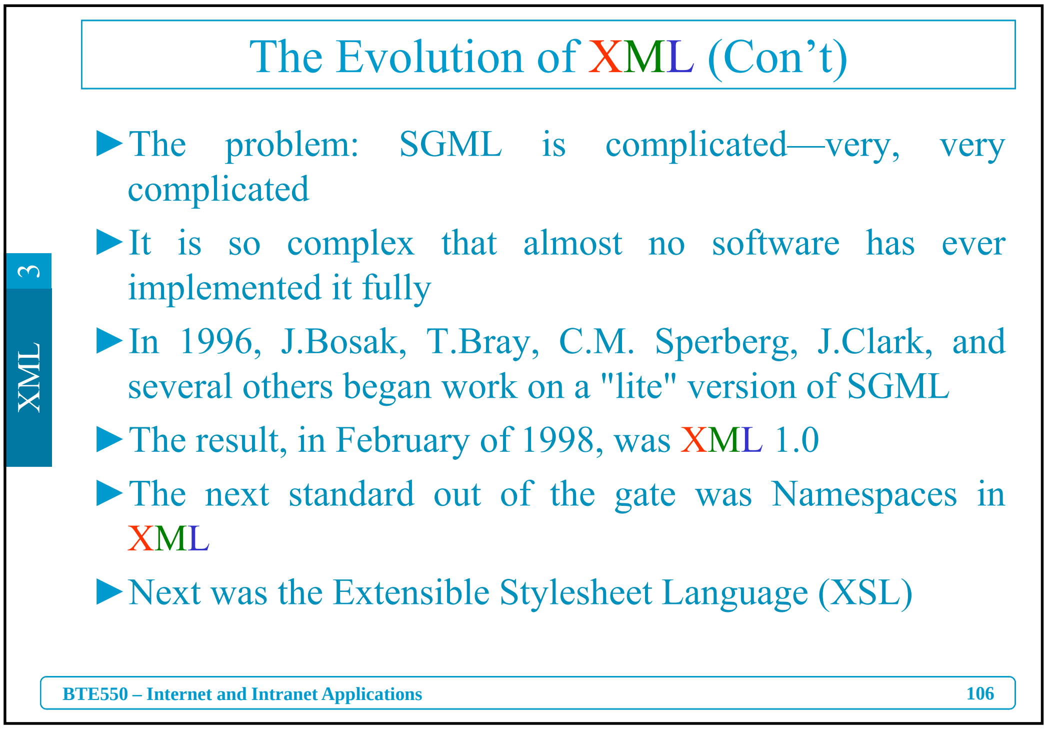


XML Base Standard

SGML
(general markup language)

XML
(W3C Recommendation)
Data Structuring

HTML
(W3C Recommendation)
Data Presentation



The Evolution of XML (Con't)

- ►The problem: SGML is complicated—very, very complicated
- ►It is so complex that almost no software has ever implemented it fully
- ►In 1996, J.Bosak, T.Bray, C.M. Sperberg, J.Clark, and several others began work on a "lite" version of SGML
- ►The result, in February of 1998, was XML 1.0
- ►The next standard out of the gate was Namespaces in XML
- ►Next was the Extensible Stylesheet Language (XSL)

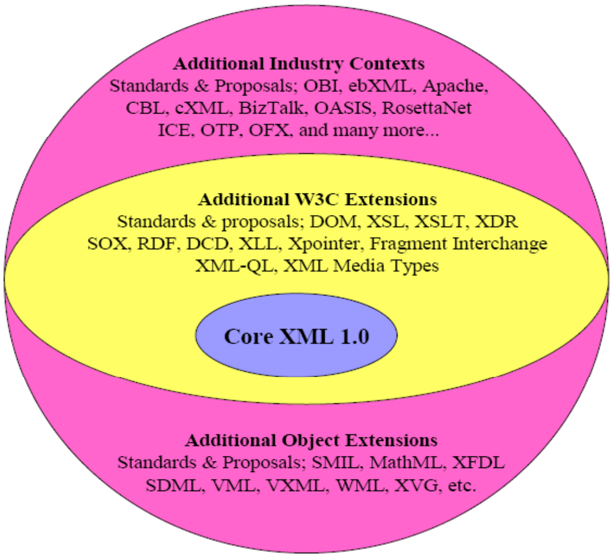
The Evolution of XML (Con’t)

XML 3

- Development of extensions to the core XML specification continues
  - XML Namespaces
  - XML DTDs, XML Schema
  - XSL (Extensible Style Sheet Language)
  - XPath (=XSLT∩ XPointer), XLink
  - XQL (XML Query Language)
  - eXcelon

XML Family of Standards

XML 3



Additional Industry Contexts
Standards & Proposals; OBI, ebXML, Apache,
CBL, cXML, BizTalk, OASIS, RosettaNet
ICE, OTP, OFX, and many more...

Additional W3C Extensions
Standards & proposals; DOM, XSL, XSLT, XDR
SOX, RDF, DCD, XLL, Xpointer, Fragment Interchange
XML-QL, XML Media Types

Core XML 1.0

Additional Object Extensions
Standards & Proposals; SMIL, MathML, XFDL
SDML, VML, VXML, WML, XVG, etc.

XML Fundamentals

XML Documents and XML Files

- An XML document contains text, never binary data
- It can be opened with any program that knows how to read a text file

```
<person>
   Alan Turing
</person>
```

person.xml

A very simple yet complete XML document

Your operating system may or may not like these names
But an XML parser won't care

Elements, Tags, and Character Data

```
<person>
   Alan Turing
</person>
```

- Example is composed of a single **element** named person
- The element is delimited by the start-tag <person> and the end-tag </person>.
- Everything between the start-tag and the end-tag of the element is called the element's **content**
- The whitespace is part of the content, though many applications will choose to ignore it
- The string "Alan Turing" and its surrounding whitespace are **character data**

XML Characteristics

- **Elements**

```
<PurchaseOrder>
</PurchaseOrder>
```

XML Characteristics

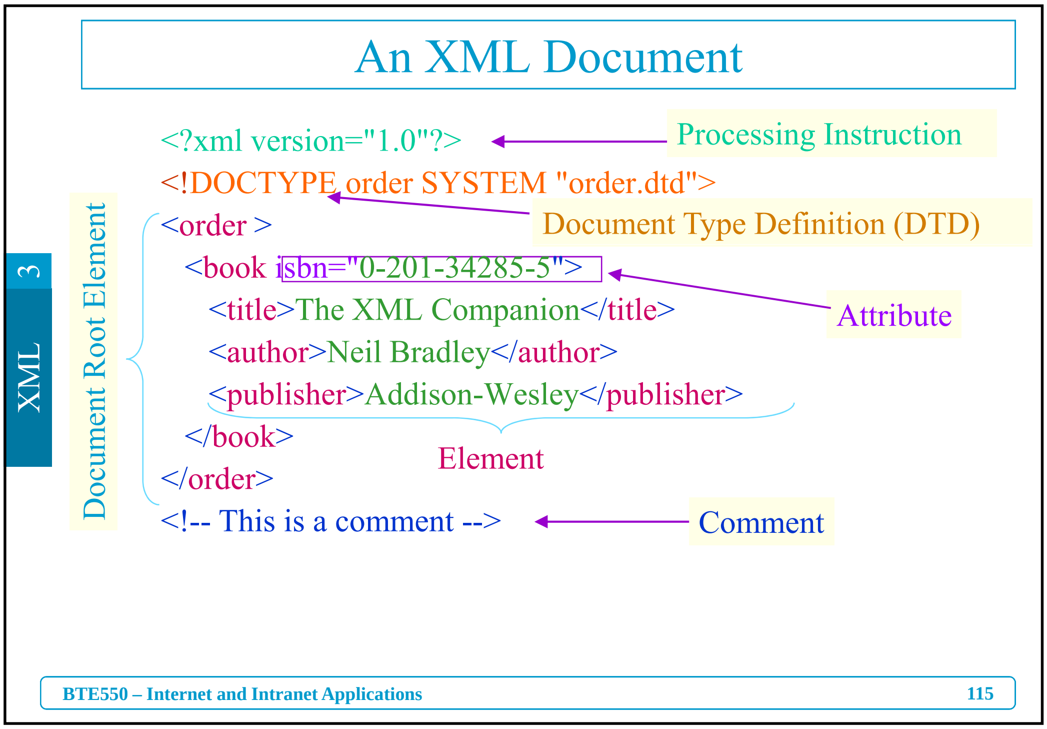
- Elements
- **Text**

```
<PurchaseOrder>
<description>Battery</description>
<quantity>9</quantity>
<price>60</price>
```

XML Characteristics

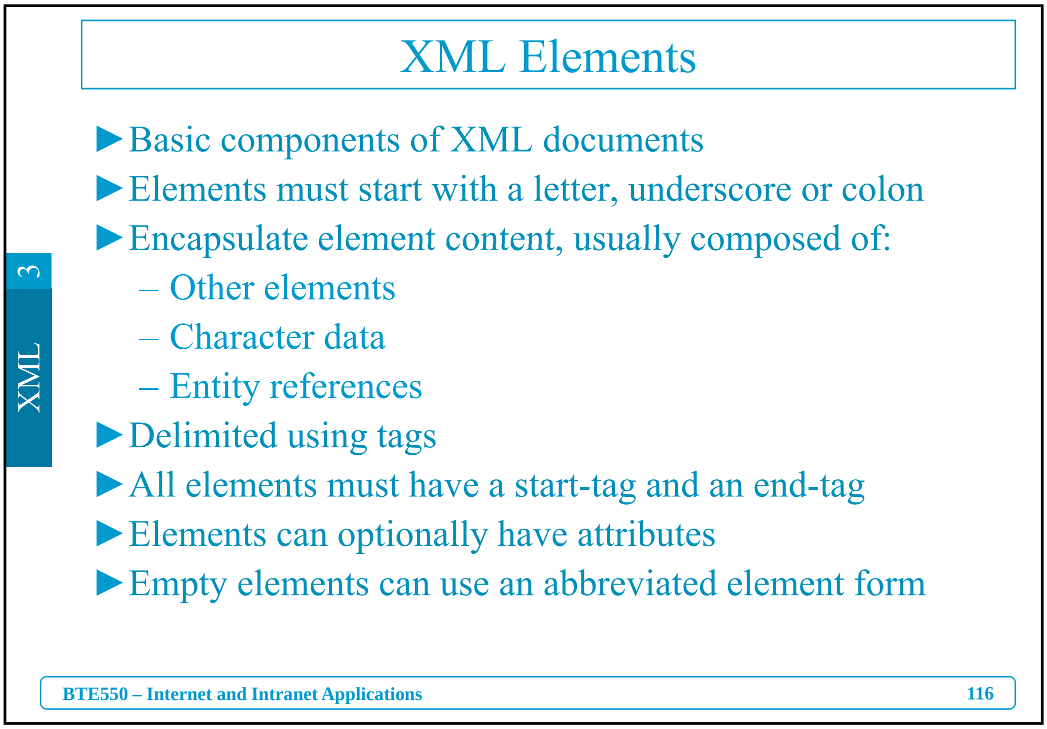
- Elements
- Text
- **Attributes**

```
<ShoeOrder id="4040458">
<color>Brown</color>
<size>9</size>
<width>AA</width>
</ShoeOrder>
```



An XML Document

```
<?xml version="1.0"?>
<!DOCTYPE order SYSTEM "order.dtd">
<order >
  <book isbn="0-201-34285-5">
    <title>The XML Companion</title>
    <author>Neil Bradley</author>
    <publisher>Addison-Wesley</publisher>
  </book>
</order>
<!-- This is a comment -->
```



XML Elements

- Basic components of XML documents
- Elements must start with a letter, underscore or colon
- Encapsulate element content, usually composed of:
  - Other elements
  - Character data
  - Entity references
- Delimited using tags
- All elements must have a start-tag and an end-tag
- Elements can optionally have attributes
- Empty elements can use an abbreviated element form

XML Namespaces

XML 3

- ► XML Namespaces allow a prefix to be associated with an element to avoid name collisions
- ► XML Namespaces are a W3C specification
- ► A unique URI must be used with a prefix to denote elements in this namespace from other namespaces
- ► The URI is only for distinguishing prefixes, it is not actually resolved
- ► Namespaces use the reserve word xmlns

```
<CC:LunchMenu xmlns:Camp="http://catering.com/CC">
. . .
<CC:MainCourse>. . .</CC:MainCourse>
```

Case Sensitivity

XML 3

- ► XML, unlike HTML, is case sensitive
- ► <Person> is not the same as <PERSON> is not the same as <person>.
- ► If you open an element with a <person> tag, you can't close it with a </PERSON> tag

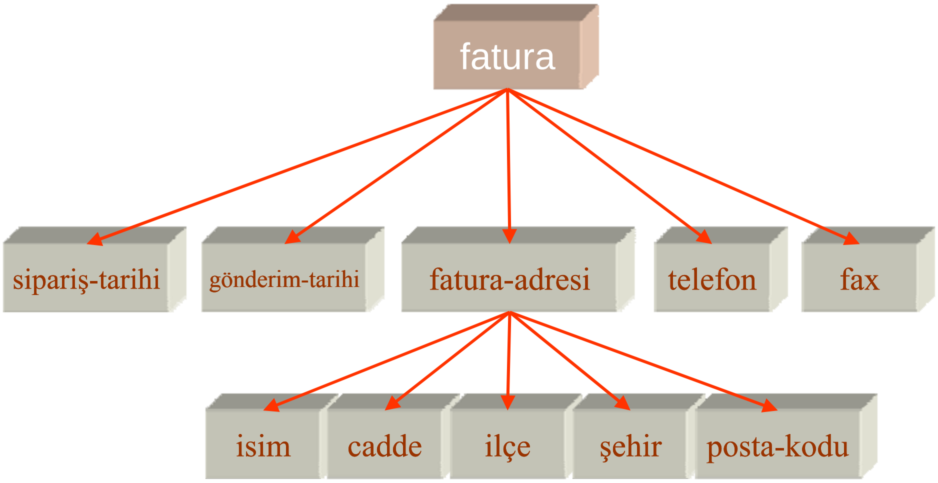
XML Trees

- Every XML document has one element that does not have a parent: root element

```
<invoice>
    <orderDate>2005-01-01</orderDate>
    <shipDate>2005-01-05</shipDate>
    <billingAddress>
        <name>Paul Biron</name>
        <street>123 IBM Avenue</street>
        <city>Hawthorne</city>
        <state>NY</state>
        <zip>10532</zip>
    </billingAddress>
    <voice>555-1234</voice>
    <fax>555-4321</fax>
</invoice>
```

Root Element is invoice

XML Trees (Con't)



Attributes

- ►Elements can contain attributes to provide information about the element
- ►Attributes are not considered part of an element’s content
- ►Attributes are not part of the presentation to an end user, though they may be used to affect the presentation
- ►An attribute is a name-value pair attached to the element's start-tag
- ►Names are separated from values by an equals sign and optional whitespace
- ►Values are enclosed in single or double quotation marks

Attributes (Con’t)

```
<person born="1912-06-23" died="1954-06-07">
    Alan Turing
</person>
```

or

```
<person born= `1912-06-23` died= `1954-06-07`>
    Alan Turing
</person>
```

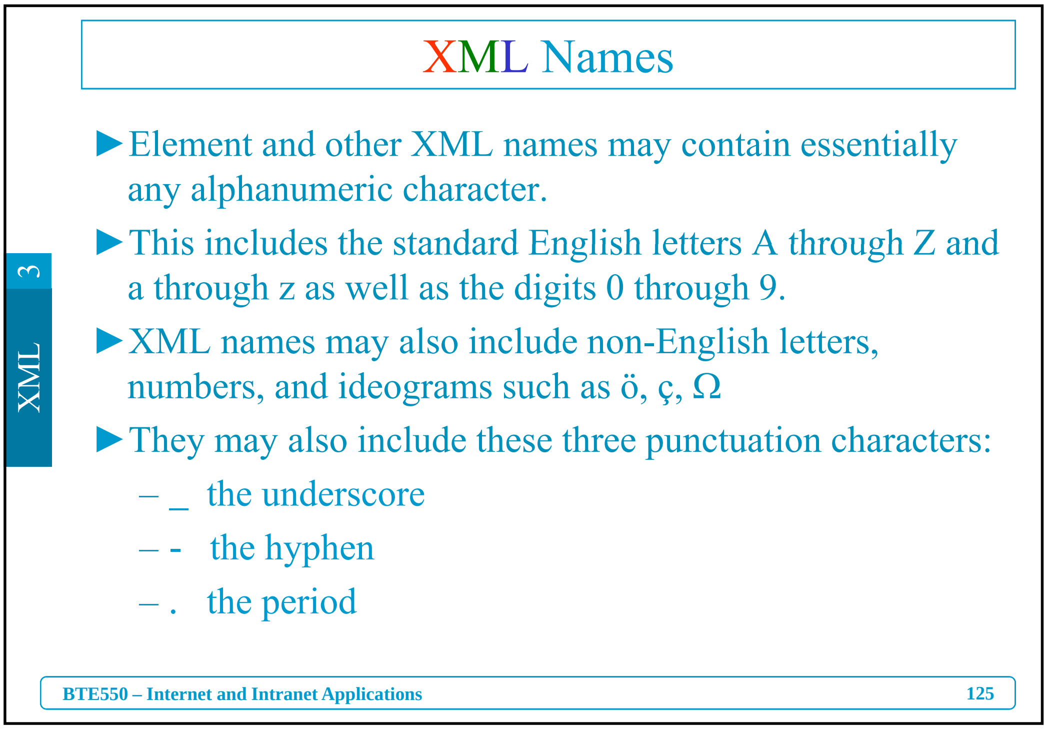
Attributes (Con’t)

```
<person>
      <name first="Alan" last="Turing"/>
      <profession value="computer scientist"/>
      <profession value="mathematician"/>
     <profession value="cryptographer"/>
</person>
```

When and whether one should use child elements or attributes to hold information?
This is a subject of heated debate

White Space

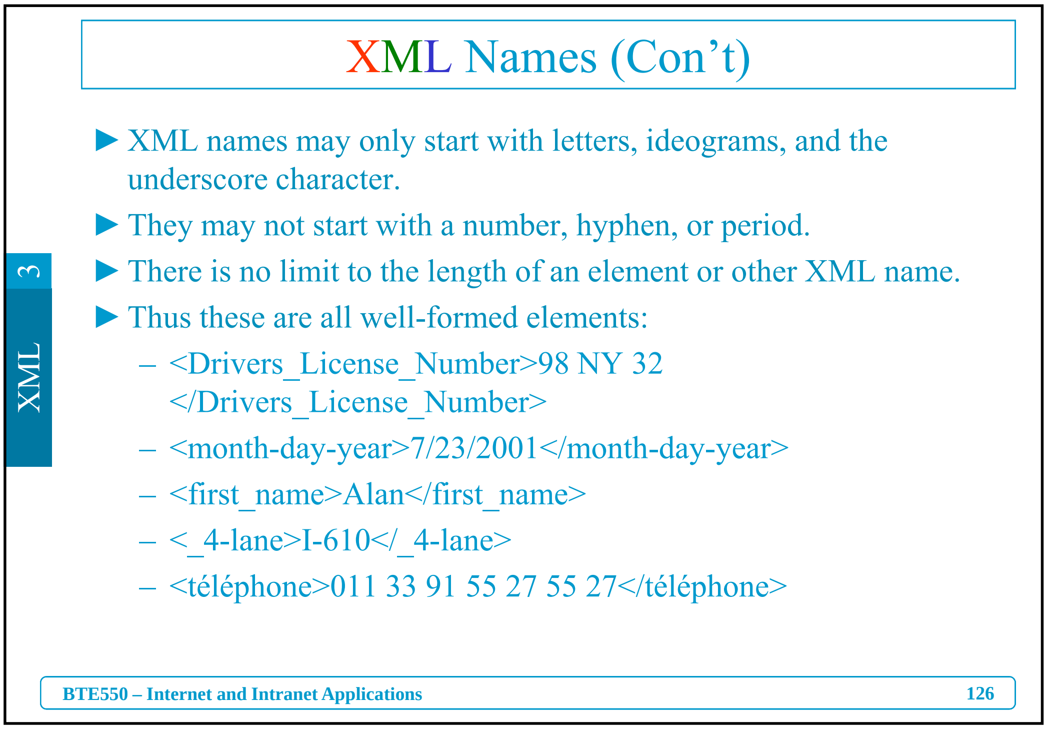
- ► XML defines white space as any of these 4 characters
  - – Horizontal tab
  - – Line feed
  - – Carriage return
  - – Space
- ► An XML parser must pass all white space contained within content to the application
- ► An XML parser may remove white space in element tags and attribute values
- ► All end of line characters are converted to line feed characters by parsers



XML Names

XML 3

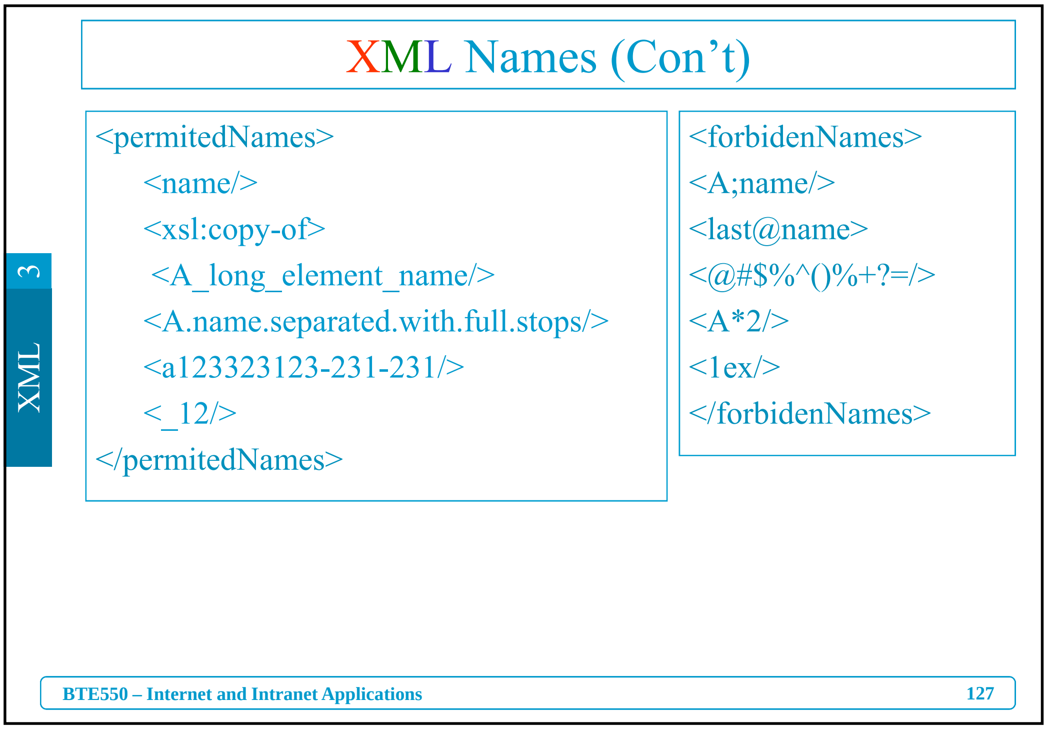
- ▶ Element and other XML names may contain essentially any alphanumeric character.
- ▶ This includes the standard English letters A through Z and a through z as well as the digits 0 through 9.
- ▶ XML names may also include non-English letters, numbers, and ideograms such as ö, ç, Ω
- ▶ They may also include these three punctuation characters:
  - – _ the underscore
  - – - the hyphen
  - – . the period



XML Names (Con’t)

XML 3

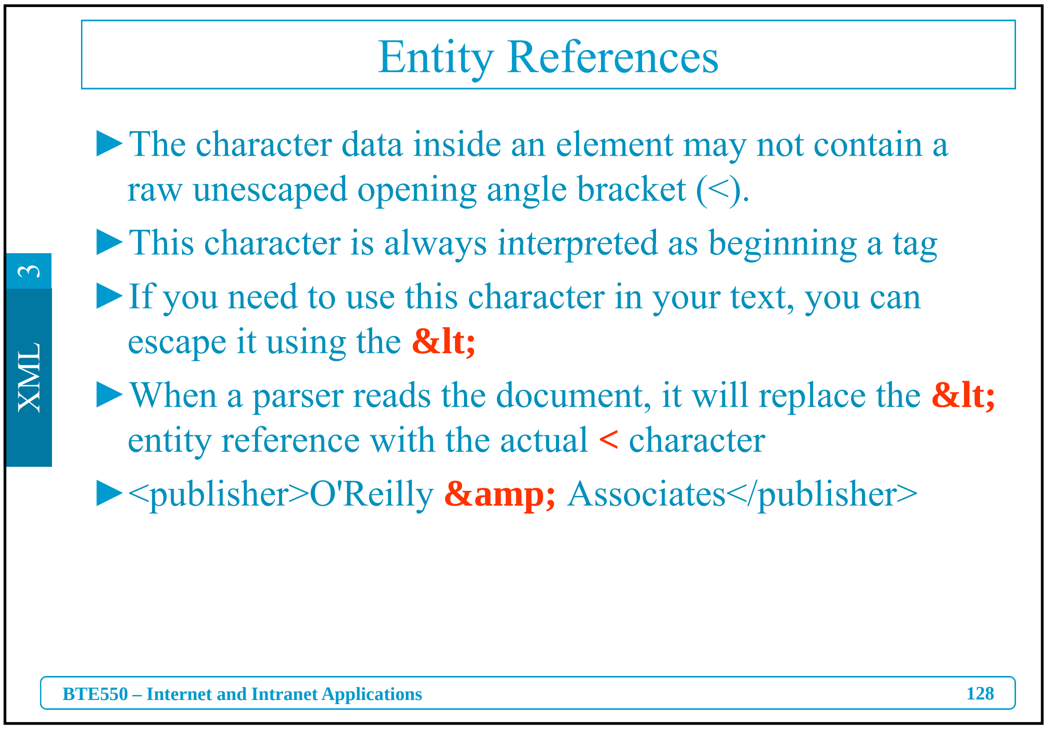
- ▶ XML names may only start with letters, ideograms, and the underscore character.
- ▶ They may not start with a number, hyphen, or period.
- ▶ There is no limit to the length of an element or other XML name.
- ▶ Thus these are all well-formed elements:
  - – <Drivers_License_Number>98 NY 32 </Drivers_License_Number>
  - – <month-day-year>7/23/2001</month-day-year>
  - – <first_name>Alan</first_name>
  - – <_4-lane>I-610</_4-lane>
  - – <téléphone>011 33 91 55 27 55 27</téléphone>



XML Names (Con’t)

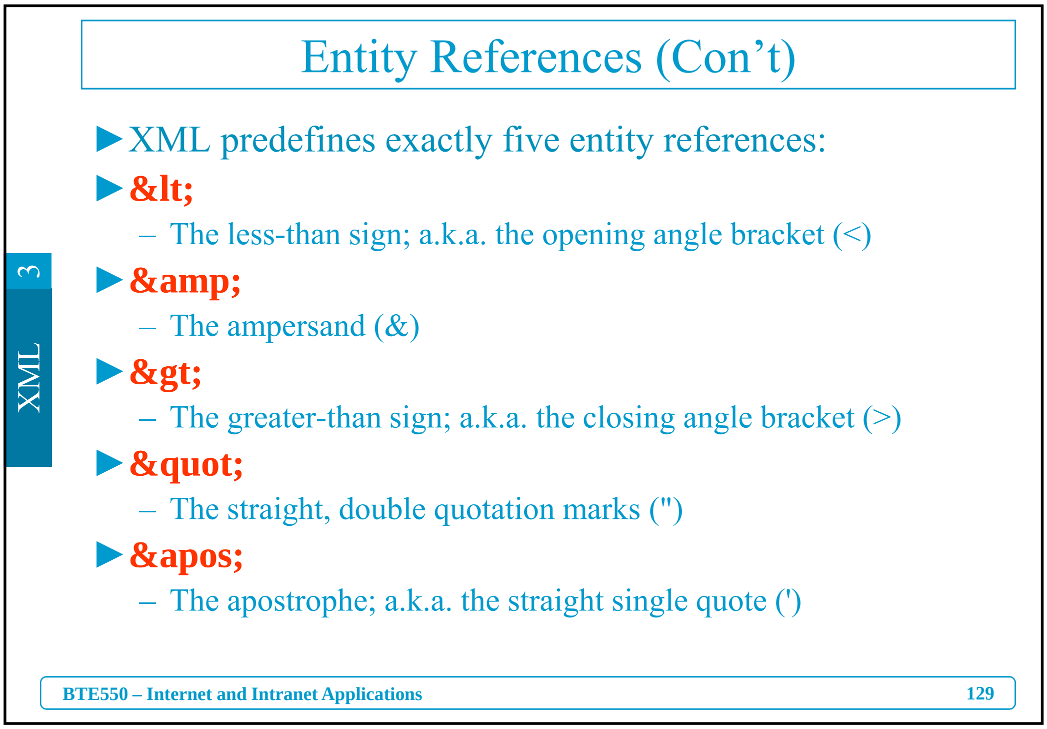
```
<permitedNames>
    <name/>
    <xsl:copy-of>
     <A_long_element_name/>
    <A.name.separated.with.full.stops/>
    <a123323123-231-231/>
    <_12/>
</permitedNames>
```

```
<forbidenNames>
<A;name/>
<last@name>
<@#$%^()%+?=/>
<A*2/>
<1ex/>
</forbidenNames>
```



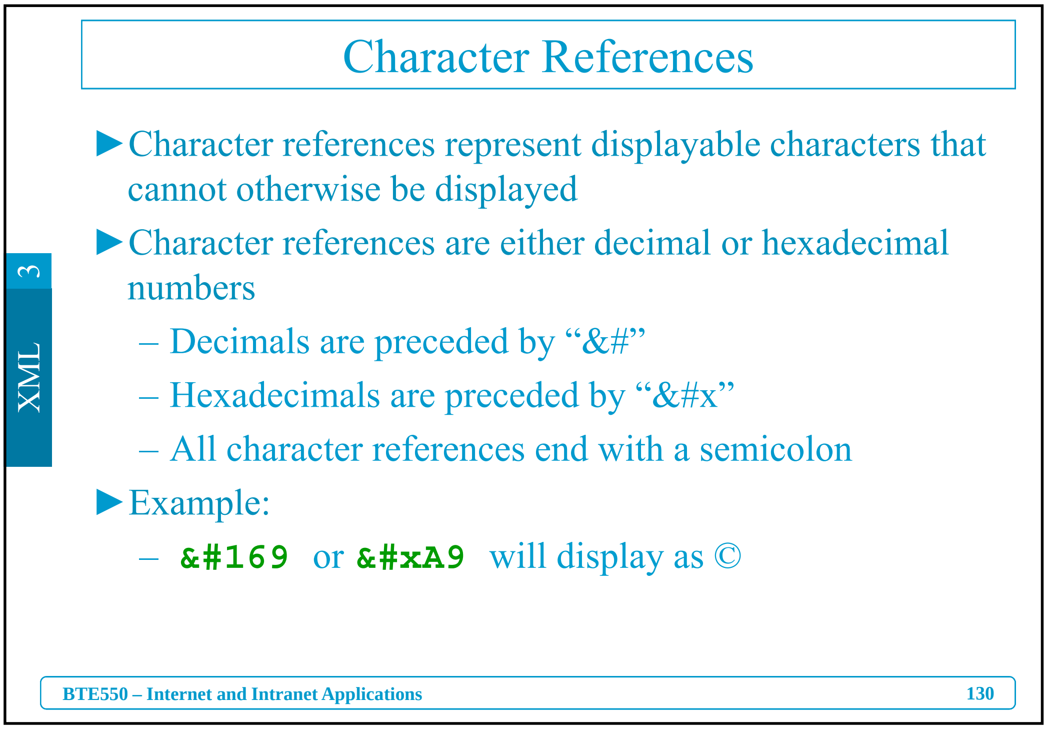
Entity References

- The character data inside an element may not contain a raw unescaped opening angle bracket (<).
- This character is always interpreted as beginning a tag
- If you need to use this character in your text, you can escape it using the **&lt;**
- When a parser reads the document, it will replace the **&lt;** entity reference with the actual < character
- <publisher>O'Reilly **&amp;** Associates</publisher>



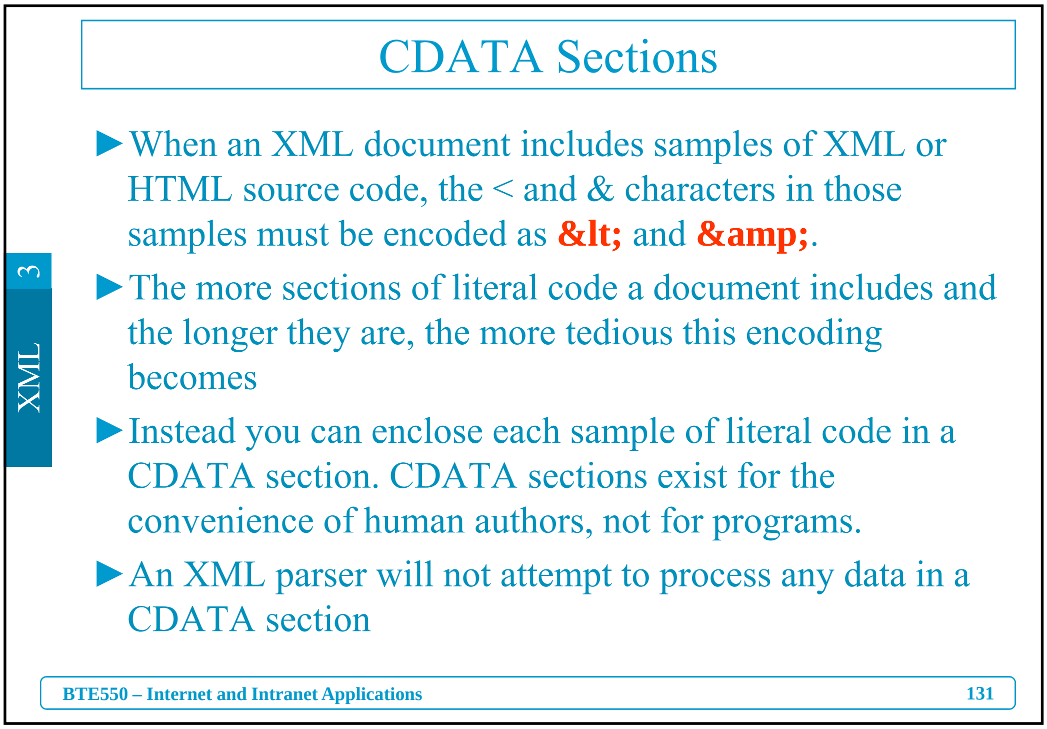
Entity References (Con’t)

- XML predefines exactly five entity references:
- **&lt;**
  - The less-than sign; a.k.a. the opening angle bracket (<)
- **&amp;**
  - The ampersand (&)
- **&gt;**
  - The greater-than sign; a.k.a. the closing angle bracket (>)
- **&quot;**
  - The straight, double quotation marks (")
- **&apos;**
  - The apostrophe; a.k.a. the straight single quote (')



Character References

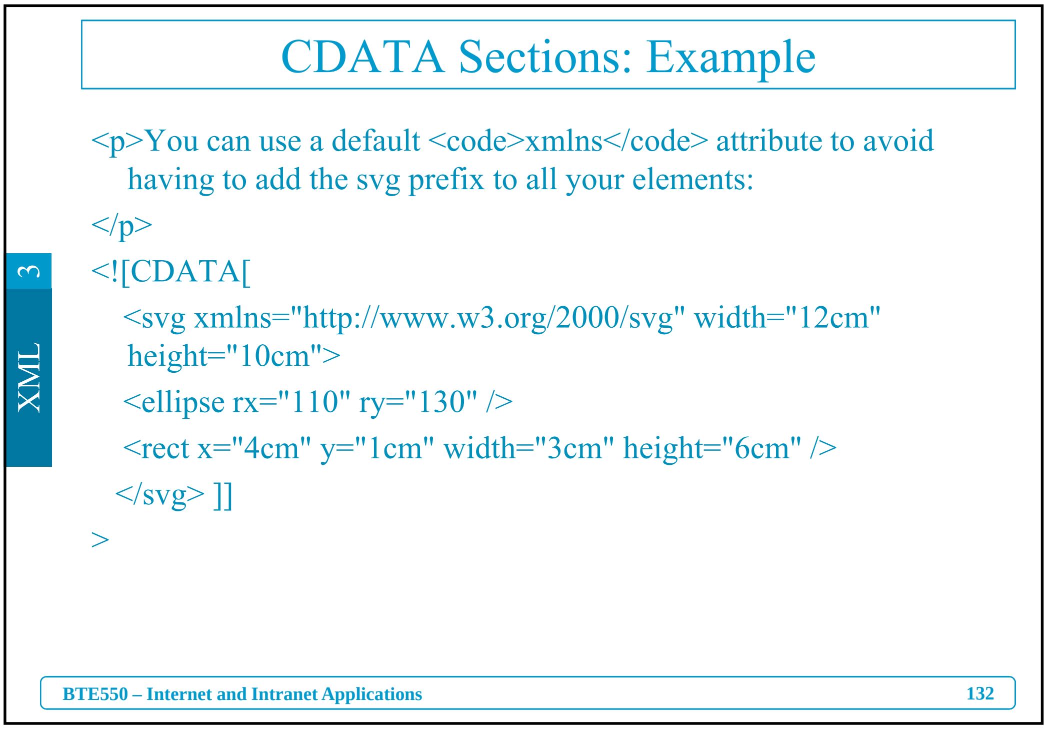
- Character references represent displayable characters that cannot otherwise be displayed
- Character references are either decimal or hexadecimal numbers
  - Decimals are preceded by “&#”
  - Hexadecimals are preceded by “&#x”
  - All character references end with a semicolon
- Example:
  - `&#169` or `&#xA9` will display as ©



CDATA Sections

3 XML

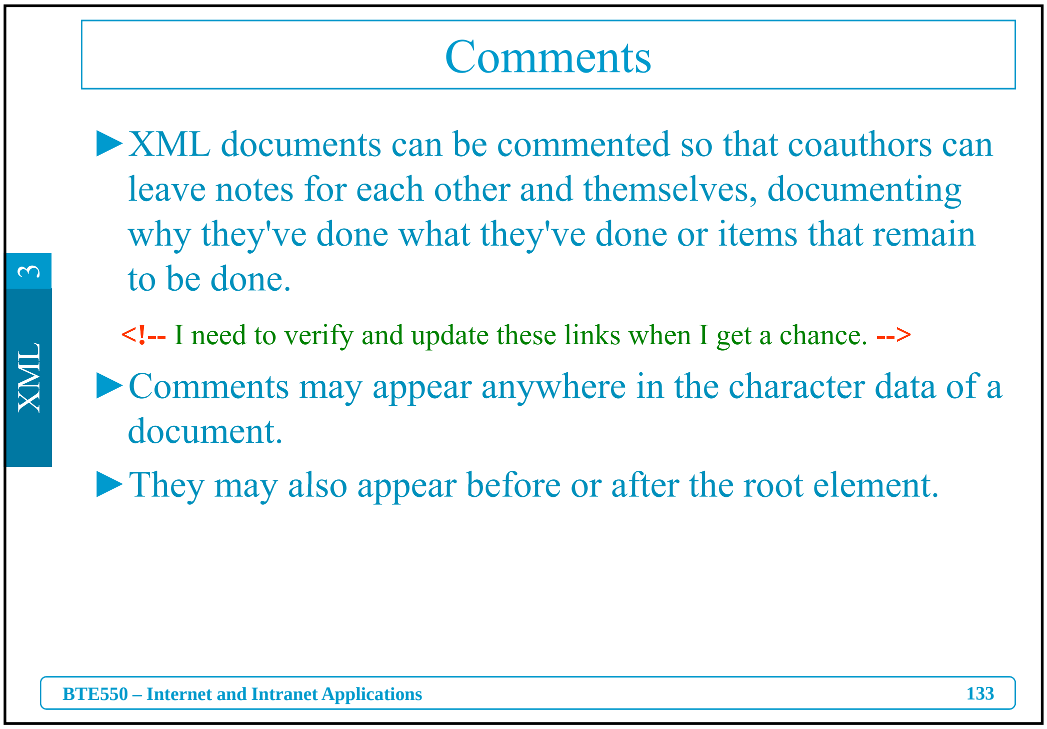
- When an XML document includes samples of XML or HTML source code, the < and & characters in those samples must be encoded as **&lt;** and **&amp;**.
- The more sections of literal code a document includes and the longer they are, the more tedious this encoding becomes
- Instead you can enclose each sample of literal code in a CDATA section. CDATA sections exist for the convenience of human authors, not for programs.
- An XML parser will not attempt to process any data in a CDATA section



CDATA Sections: Example

3 XML

```
<p>You can use a default <code>xmlns</code> attribute to avoid
   having to add the svg prefix to all your elements:
</p>
<![CDATA[
   <svg xmlns="http://www.w3.org/2000/svg" width="12cm"
   height="10cm">
   <ellipse rx="110" ry="130" />
   <rect x="4cm" y="1cm" width="3cm" height="6cm" />
  </svg> ]]
>
```



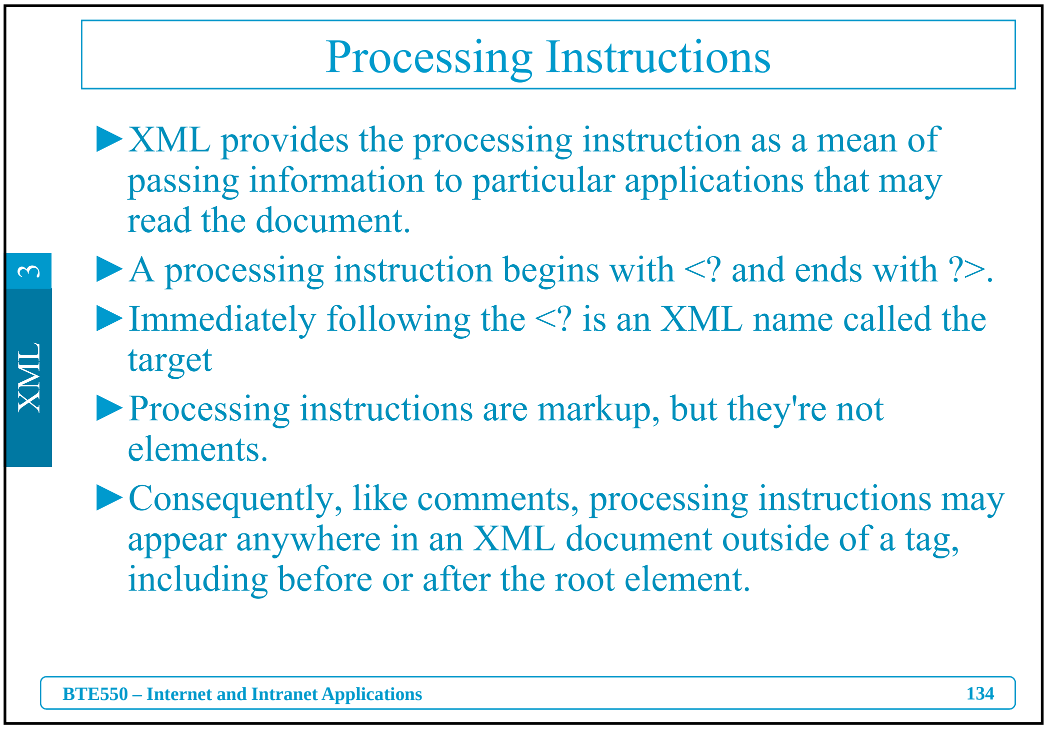
3 XML

Comments

- XML documents can be commented so that coauthors can leave notes for each other and themselves, documenting why they've done what they've done or items that remain to be done.

```
<!-- I need to verify and update these links when I get a chance. -->
```

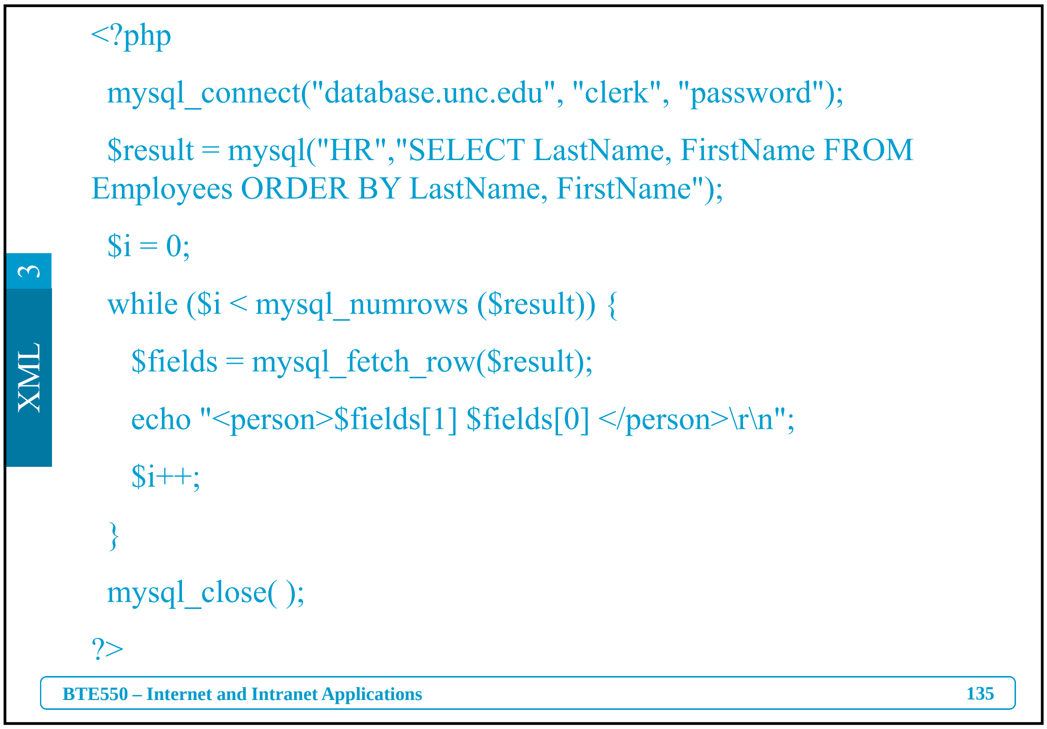
- Comments may appear anywhere in the character data of a document.
- They may also appear before or after the root element.



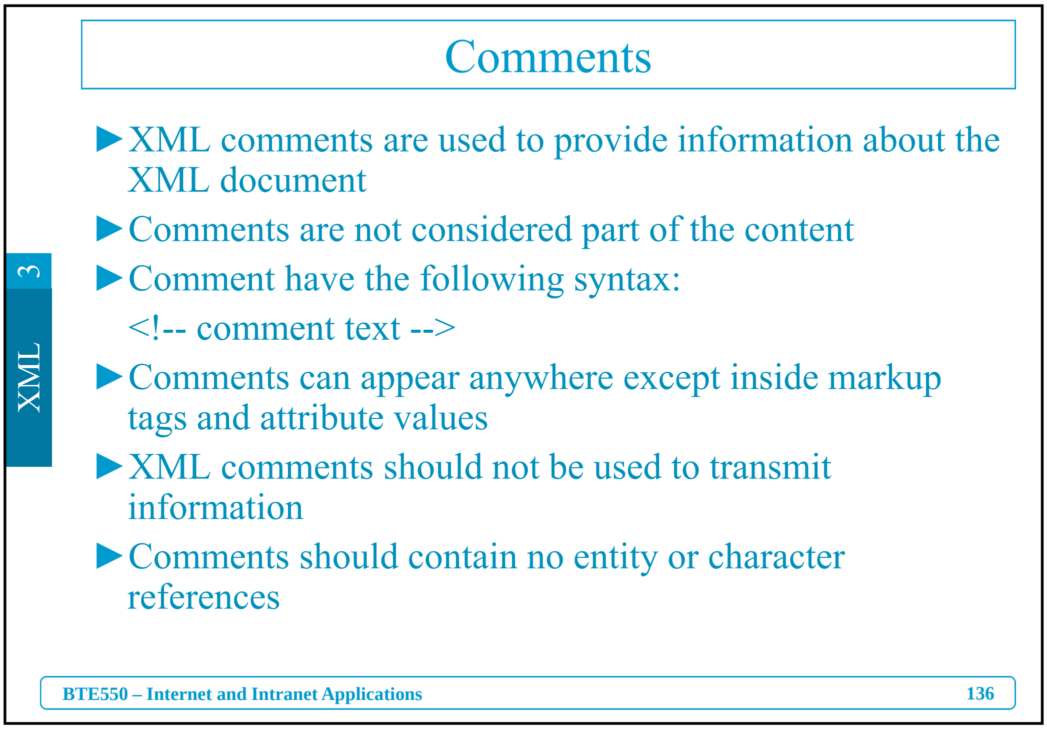
3 XML

Processing Instructions

- XML provides the processing instruction as a mean of passing information to particular applications that may read the document.
- A processing instruction begins with <? and ends with ?>.
- Immediately following the <? is an XML name called the target
- Processing instructions are markup, but they're not elements.
- Consequently, like comments, processing instructions may appear anywhere in an XML document outside of a tag, including before or after the root element.



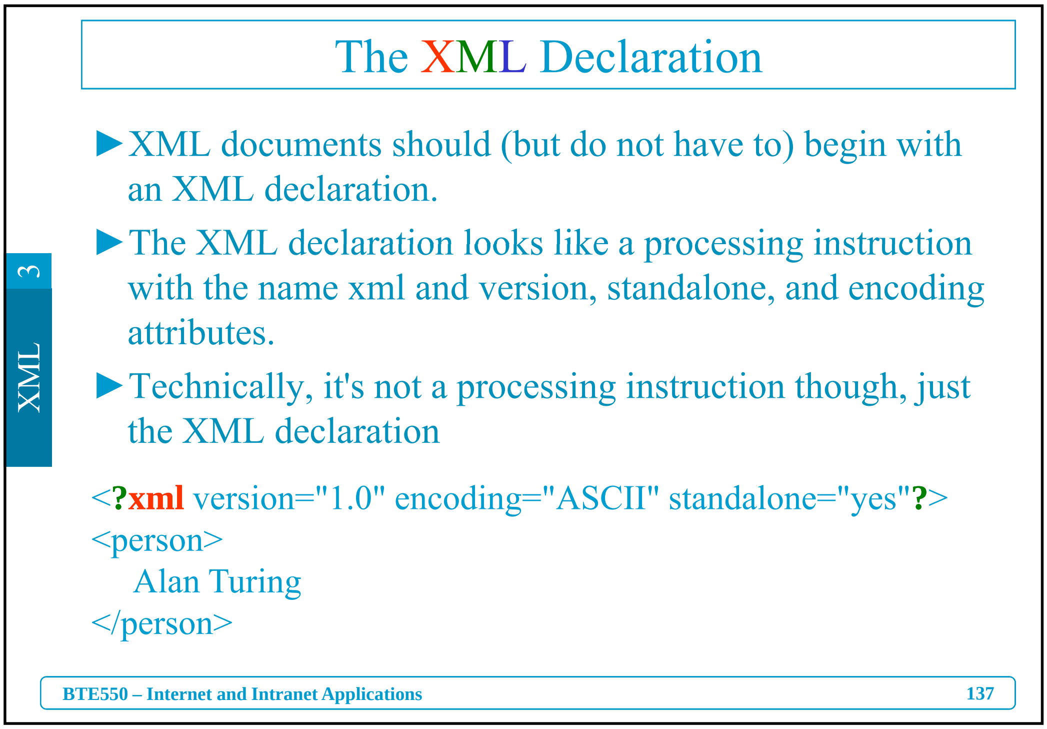
```
<?php
  mysql_connect("database.unc.edu", "clerk", "password");
  $result = mysql("HR","SELECT LastName, FirstName FROM
Employees ORDER BY LastName, FirstName");
  $i = 0;
  while ($i < mysql_numrows ($result)) {
     $fields = mysql_fetch_row($result);
     echo "<person>$fields[1] $fields[0] </person>\r\n";
     $i++;
  }
  mysql_close( );
?>
```



Comments

- ► XML comments are used to provide information about the XML document
- ► Comments are not considered part of the content
- ► Comment have the following syntax:

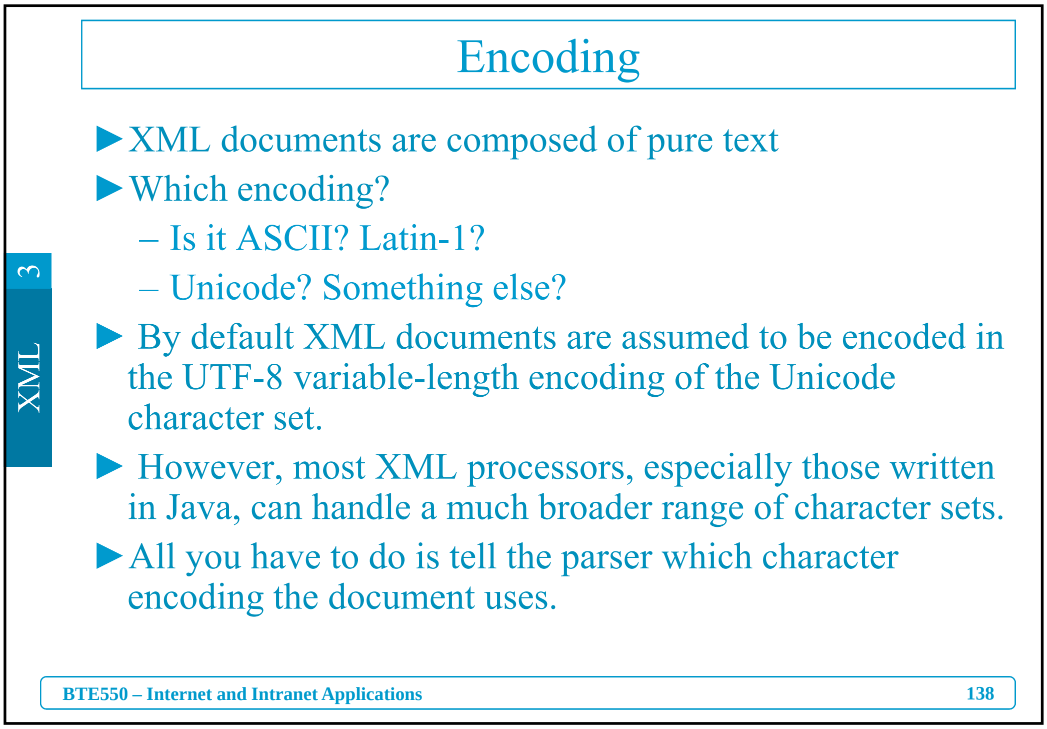
  <!-- comment text -->
- ► Comments can appear anywhere except inside markup tags and attribute values
- ► XML comments should not be used to transmit information
- ► Comments should contain no entity or character references



The XML Declaration

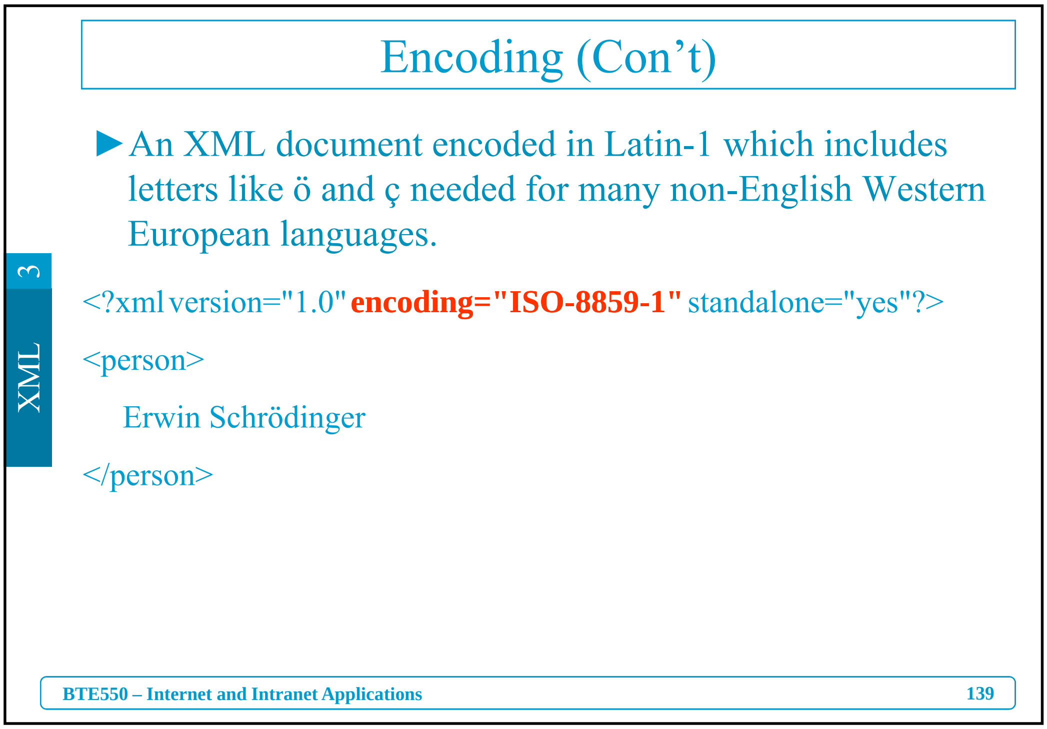
- ▶ XML documents should (but do not have to) begin with an XML declaration.
- ▶ The XML declaration looks like a processing instruction with the name xml and version, standalone, and encoding attributes.
- ▶ Technically, it's not a processing instruction though, just the XML declaration

```
<?xml version="1.0" encoding="ASCII" standalone="yes"?>
<person>
   Alan Turing
</person>
```



Encoding

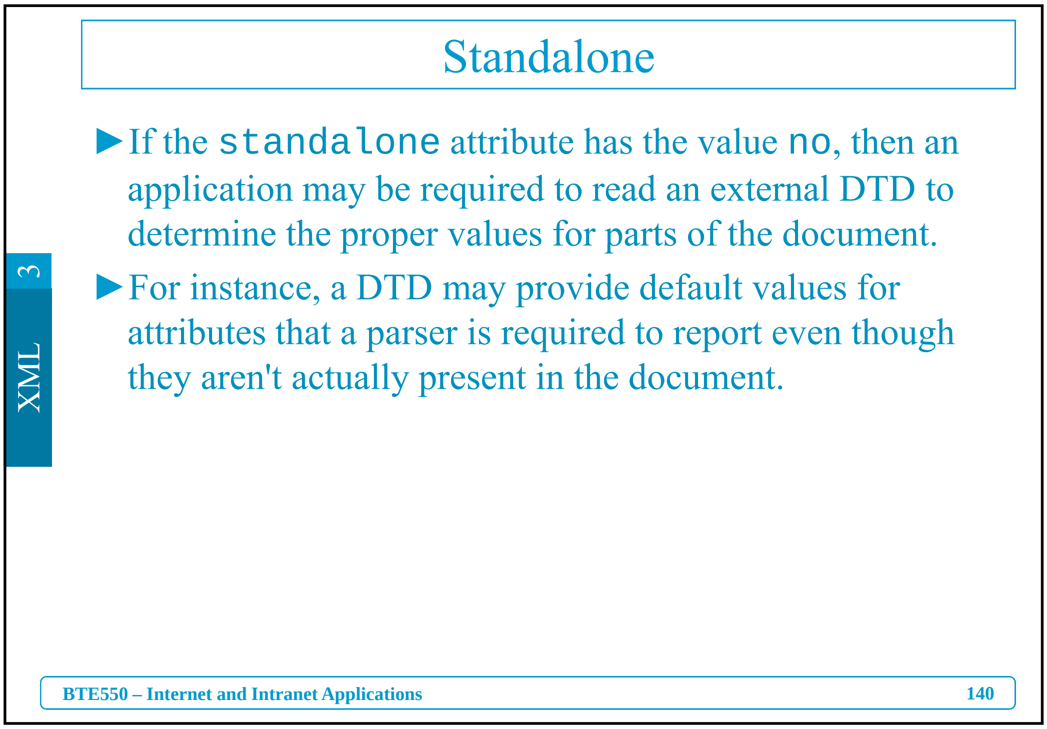
- ▶ XML documents are composed of pure text
- ▶ Which encoding?
  - – Is it ASCII? Latin-1?
  - – Unicode? Something else?
- ▶ By default XML documents are assumed to be encoded in the UTF-8 variable-length encoding of the Unicode character set.
- ▶ However, most XML processors, especially those written in Java, can handle a much broader range of character sets.
- ▶ All you have to do is tell the parser which character encoding the document uses.



Encoding (Con’t)

- An XML document encoded in Latin-1 which includes letters like ö and ç needed for many non-English Western European languages.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<person>
   Erwin Schrödinger
</person>
```



Standalone

- If the `standalone` attribute has the value `no`, then an application may be required to read an external DTD to determine the proper values for parts of the document.
- For instance, a DTD may provide default values for attributes that a parser is required to report even though they aren't actually present in the document.

Well-Formedness

3 XML

- Every XML document must be well-formed. This means it must adhere to a number of rules, including the following:
  1. Every start-tag must have a matching end-tag.
  2. Elements may nest, but may not overlap.
  3. There must be exactly one root element.
  4. Attribute values must be quoted.
  5. An element may not have two attributes with the same name.
  6. Comments and processing instructions may not appear inside tags.
  7. No unescaped < or & signs may occur in the character data of an element or attribute.

Well-formed XML Examples

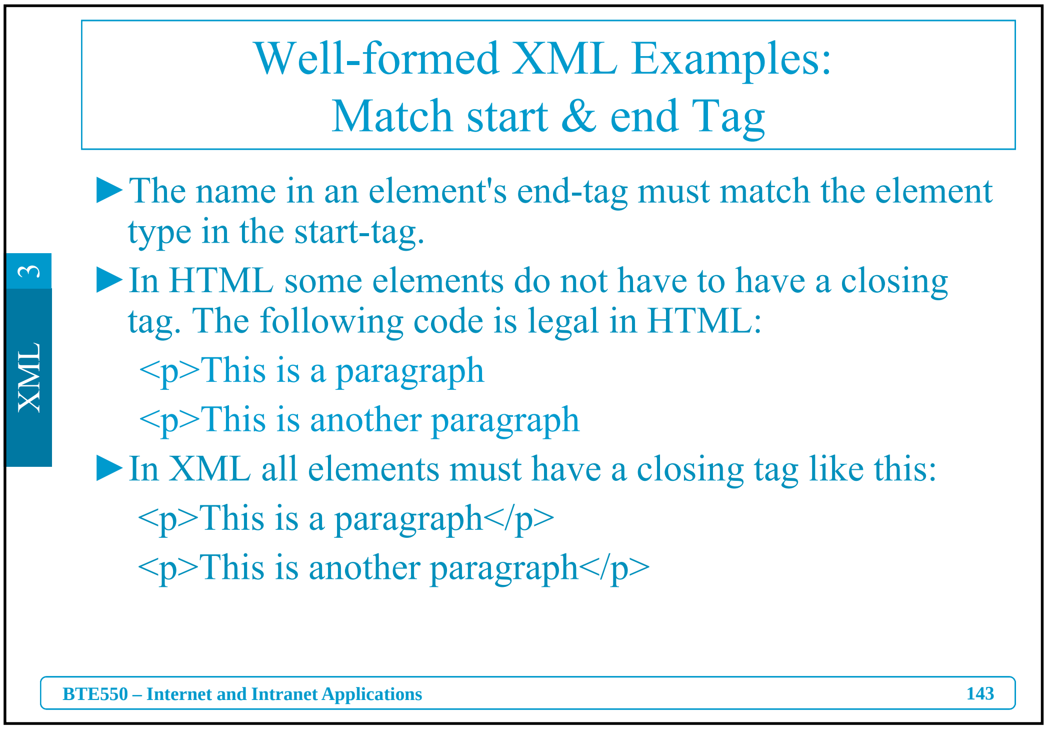
3 XML

- A well formed document with one element:

  <text>This is an XML document</text>

- A well formed document with several elements:

```
<name>
  <first>Binnur</first>
   <last>Kurt</last>
</name>
```



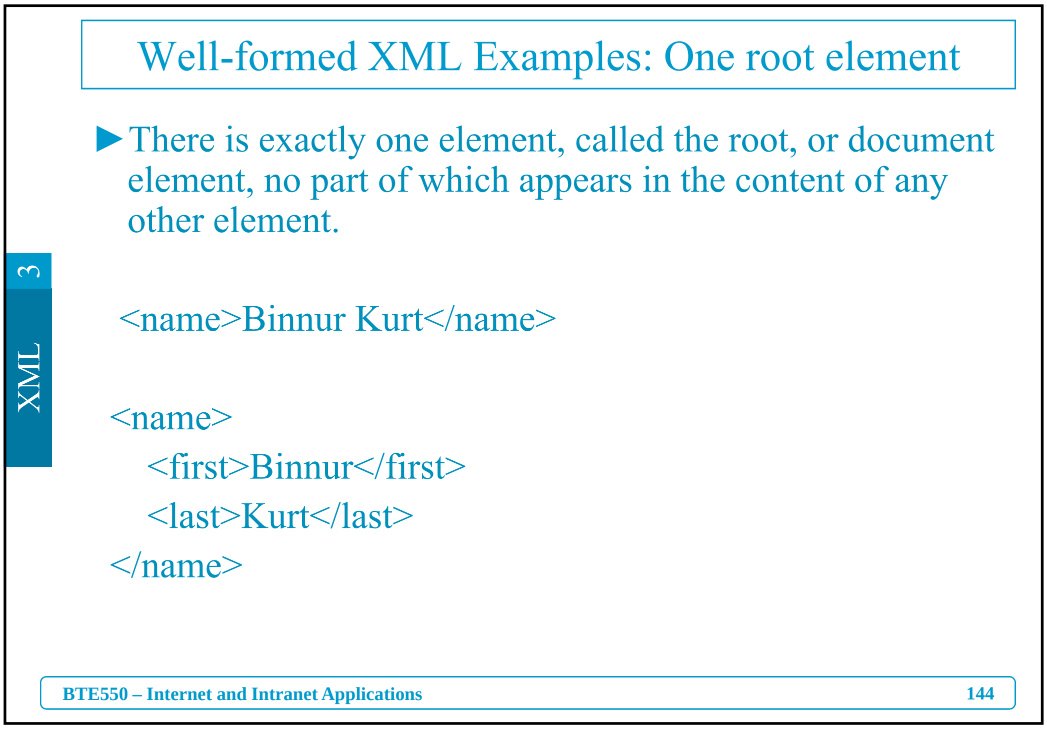
Well-formed XML Examples: Match start & end Tag

- The name in an element's end-tag must match the element type in the start-tag.
- In HTML some elements do not have to have a closing tag. The following code is legal in HTML:

```
<p>This is a paragraph
<p>This is another paragraph
```

- In XML all elements must have a closing tag like this:

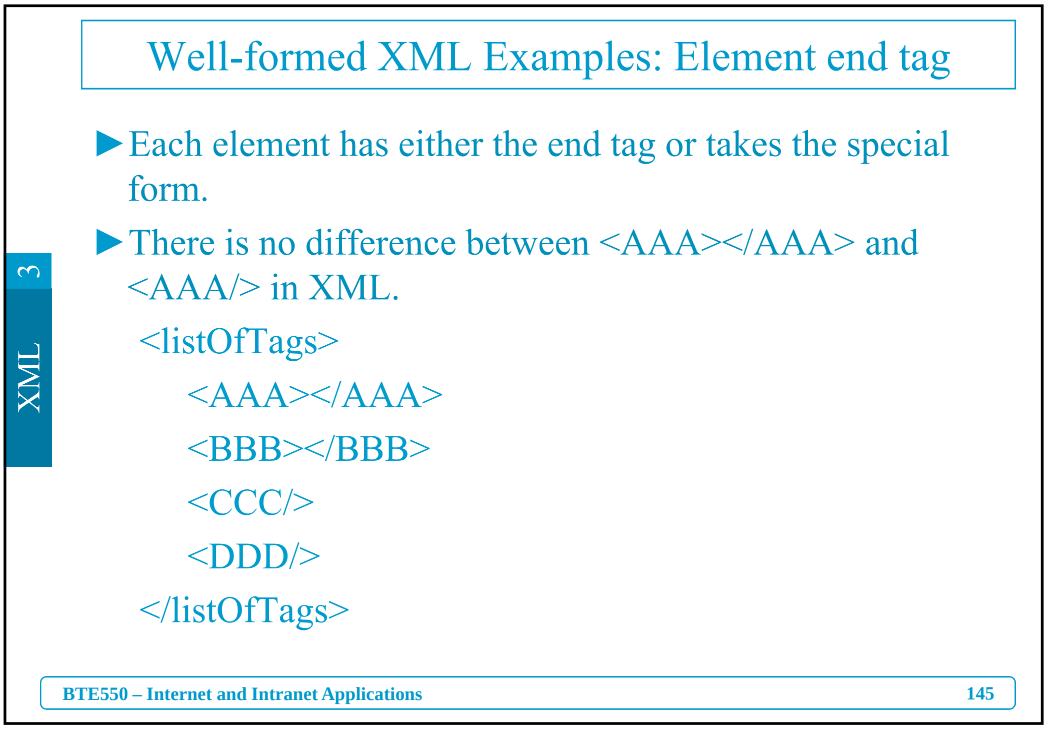
```
<p>This is a paragraph</p>
<p>This is another paragraph</p>
```



Well-formed XML Examples: One root element

- There is exactly one element, called the root, or document element, no part of which appears in the content of any other element.

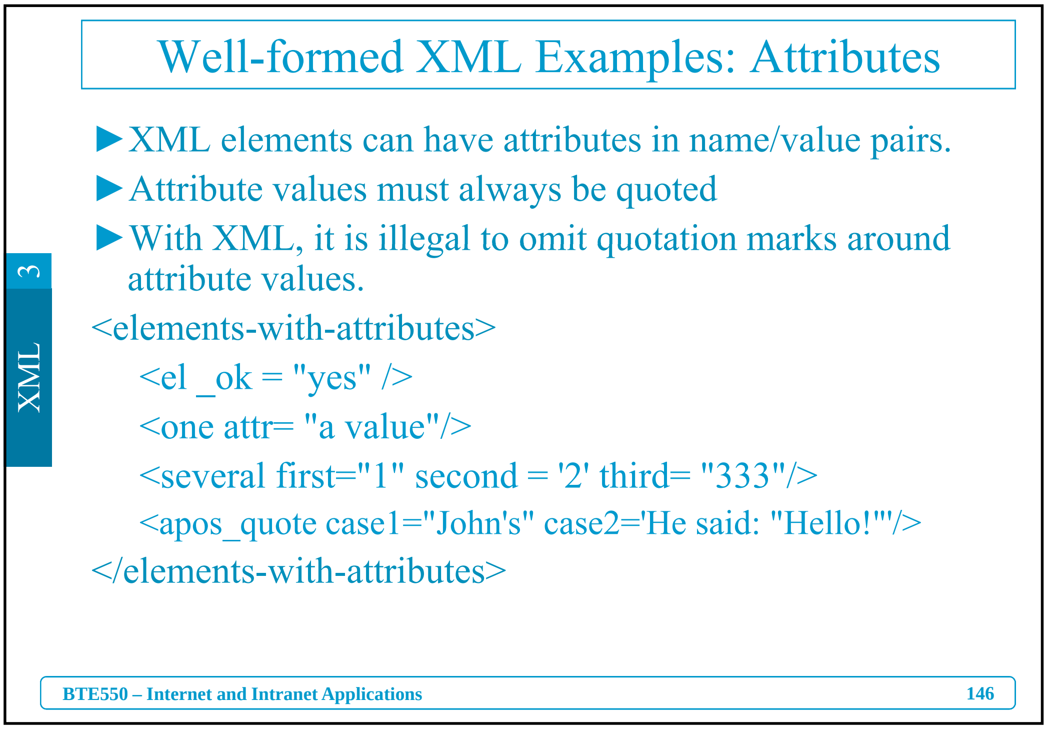
```
<name>Binnur Kurt</name>

<name>
   <first>Binnur</first>
   <last>Kurt</last>
</name>
```



Well-formed XML Examples: Element end tag

- Each element has either the end tag or takes the special form.
- There is no difference between <AAA></AAA> and <AAA/> in XML.

```
<listOfTags>
    <AAA></AAA>
    <BBB></BBB>
    <CCC/>
    <DDD/>
</listOfTags>
```



Well-formed XML Examples: Attributes

- XML elements can have attributes in name/value pairs.
- Attribute values must always be quoted
- With XML, it is illegal to omit quotation marks around attribute values.

```
<elements-with-attributes>
    <el _ok = "yes" />
    <one attr= "a value"/>
    <several first="1" second = '2' third= "333"/>
    <apos_quote case1="John's" case2='He said: "Hello!"'/>
</elements-with-attributes>
```

XML Quiz 1

3 XML

- Find errors:

```
<root>
<e1 a*b = "23432"/>
<e2 value = "12'/>
<e3 value="aa"aa"/>
<e4 value=bbbb/>
<e5 xml-ID = "xml2"/>
</root>
```

XML Quiz 1

3 XML

- Solution:

```
<root>
<e1 a*b = "23432"/>
<e2 value = "12'/>
<e3 value="aa"aa"/>
<e4 value='bbbb'/>
<e5 xml-ID = "xml2"/>
</root>
```

XML Quiz 2

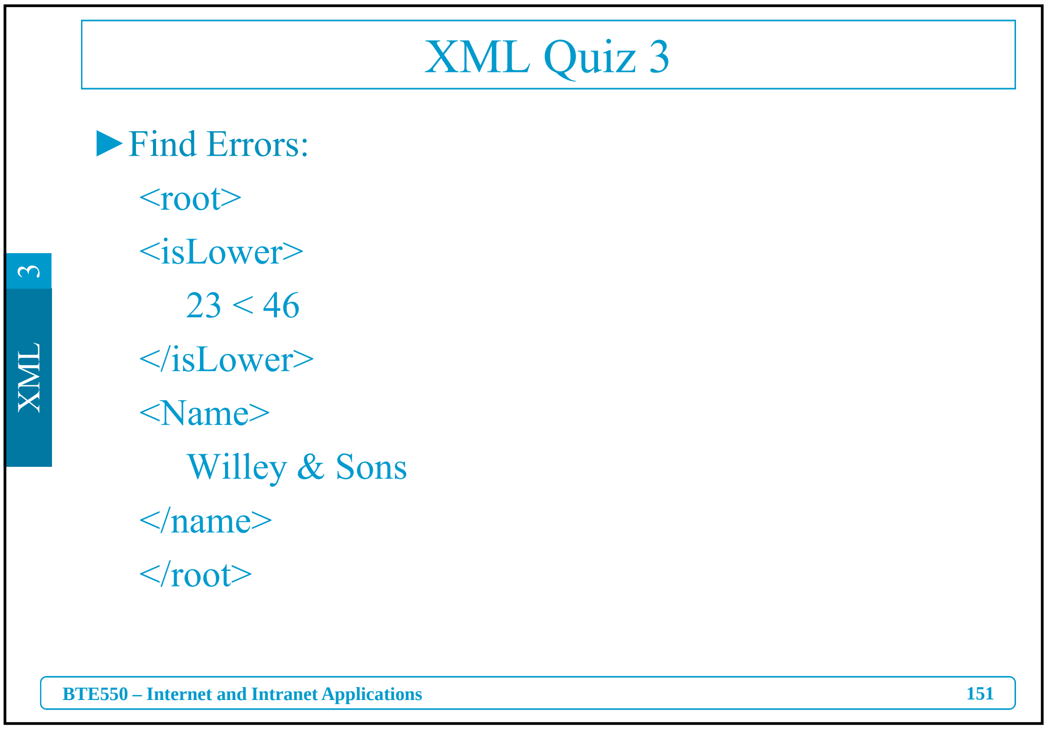
- Find Errors:

```
<root>
<example>
   <![CDATA[ <P>Q&R]]>
</example>
<Name>
   Binnur Kurt
</Name>
<Address/>
</root>
```

XML Quiz 2

- Solution:

  No error

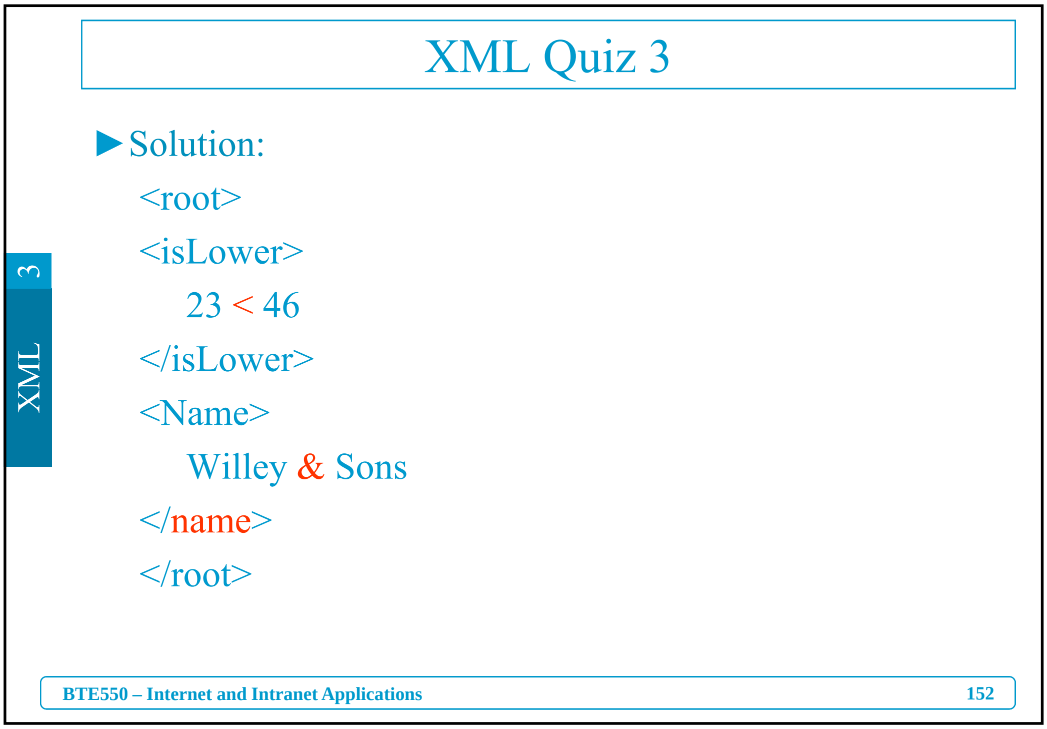


XML Quiz 3

- Find Errors:

```
<root>
<isLower>
    23 < 46
</isLower>
<Name>
    Willey & Sons
</name>
</root>
```

XML 3



XML Quiz 3

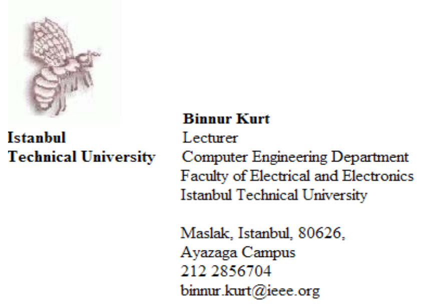
- Solution:

```
<root>
<isLower>
    23 < 46
</isLower>
<Name>
    Willey & Sons
</name>
</root>
```

XML 3

Exercise: Create an XML document

- Create an XML document that captures business card information.
- Give appropriate tag names.
- cd Lab\Mod1
- Review card.txt – make appropriate changes and create card.xml



Istanbul
Technical University

Binnur Kurt
Lecturer
Computer Engineering Department
Faculty of Electrical and Electronics
Istanbul Technical University

Maslak, Istanbul, 80626,
Ayazaga Campus
212 2856704
binnur.kurt@ieee.org