Web Technology & Programming

XML

LECTURER: KRUNAL PATEL
ICCT ENGG. COLLEGE
NEW VALLABH VIDYANAGAR

Introduction

What is XML?

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML.
- XML was designed to **describe data**.
- XML is used to store data or information.
- This data might be intended to be by read by people or by machines.
- It can be highly structured data such as data typically stored in databases or spreadsheets, or loosely structured data, such as data stored in letters or manuals.

Introduction

What is XML?

- XML tags are not predefined in XML.
- You must **define your own tags**.
- XML is **self describing**.
- XML uses a DTD (**Document Type Definition**) to formally describe the data.

Difference between XML and HTML

The main difference between XML and HTML

- XML is **not a replacement** for HTML.
  XML and HTML were designed with **different goals**:
- XML was designed to **describe data** and to focus on **what data is**.
  HTML was designed to **display data** and to focus on **how data looks**.
- HTML is about **displaying** information, XML is about **describing** information.

Use of XML

- XML can keep data separated from your HTML
- XML can be used to store data inside HTML documents
- XML can be used as a format to exchange information
- XML can be used to store data in files or in databases
- XML is a styles based markup language
- XML is hierarchical in nature.
- XML is extremely readable and easy to understand.

Working with XML

Well Formed

- The definition of "well formed" is:

- **All XML elements must have a starting and ending tag**
- **XML tags are case sensitive**
- **All XML elements must be properly nested**
- **All XML documents must have a root tag.**
- There must be one and only one top level element.

```
<root>
    <child>
      <subchild>.....</subchild>
    </child>
</root>
```

- **Attribute values must always be quoted**
- **Tags in XML are case sensitive**

Working with XML

Valid

Valid documents must:

- **Be well formed.**
  - XML documents must be well formed and are additionally error checked against a Document Type Definition (DTD).
- **Include a DTD.**
  - A DTD is a set of rules outlining which tags are allowed, what values those tags may contain and how the tags relate to each other.
- **Follow the rules set by the DTD.**
  - Typically a valid document is used when documents require error checking, and enforced structure, or are working within a company/ industry wide environment in which many documents need to follow the same guidelines

Working with XML

Incorrect

```
<note date=12/11/2007>
 <to>Tove</to>
 <from>Jani</from>
</note>
```

correct

```
<note date="12/11/2007">
 <to>Tove</to>
 <from>Jani</from>
</note>
```

Working with XML

- **Entity References**

Some characters have a special meaning in XML.

- If you place a character like "<" inside an XML element, it will generate an error because the parser interprets it as the start of a new element.

This will generate an XML error:

- <message>if salary < 1000 then</message>

To avoid that error :

- <message> if salary < 1000 then</message>

Working with XML

&lt;	<	less than
&gt;	>	greater than
&amp;	&	ampersand
&apos;	'	apostrophe
&quot;	"	quotation mark

Working with XML

- **White space is preserved**
  - HTML truncates multiple white-space characters to one single white-space but With XML, the white-space in a document is not truncated
    - HTML: Hello User
    - Output: Hello User
- **Comments in XML**
  - <!-- comment -->

XML Document

Document Parts

- Prolog
- Document Element (root element)

XML Document

- Documents are made up of a prolog and a body.
- The document prolog contains the XML Declaration and the document body contains the actual marked up document.
- According to the official XML specifications, every well formed XML document must contain a prolog

XML Document

The prolog must follow two simple rules:

- It must come before the opening root tag (first element in the document).
- It can contain an **version declaration, a document type declaration, comments, and processing instructions**

XML Document

The Prolog

- The prologue, equivalent to the header in HTML, may include the following:
- An XML declaration (optional) such as:
  - <?xml version="1.0"?>
- A DTD or reference to one (optional).
  - An example reference to an external DTD file:
  - <!DOCTYPE LANGLIST SYSTEM "langlist.dtd">
- Processing instructions - An example processing instruction that causes style to be determined by a style sheet:
  - <?xml-stylesheet type="text/css" href="xmlstyle.css"?>

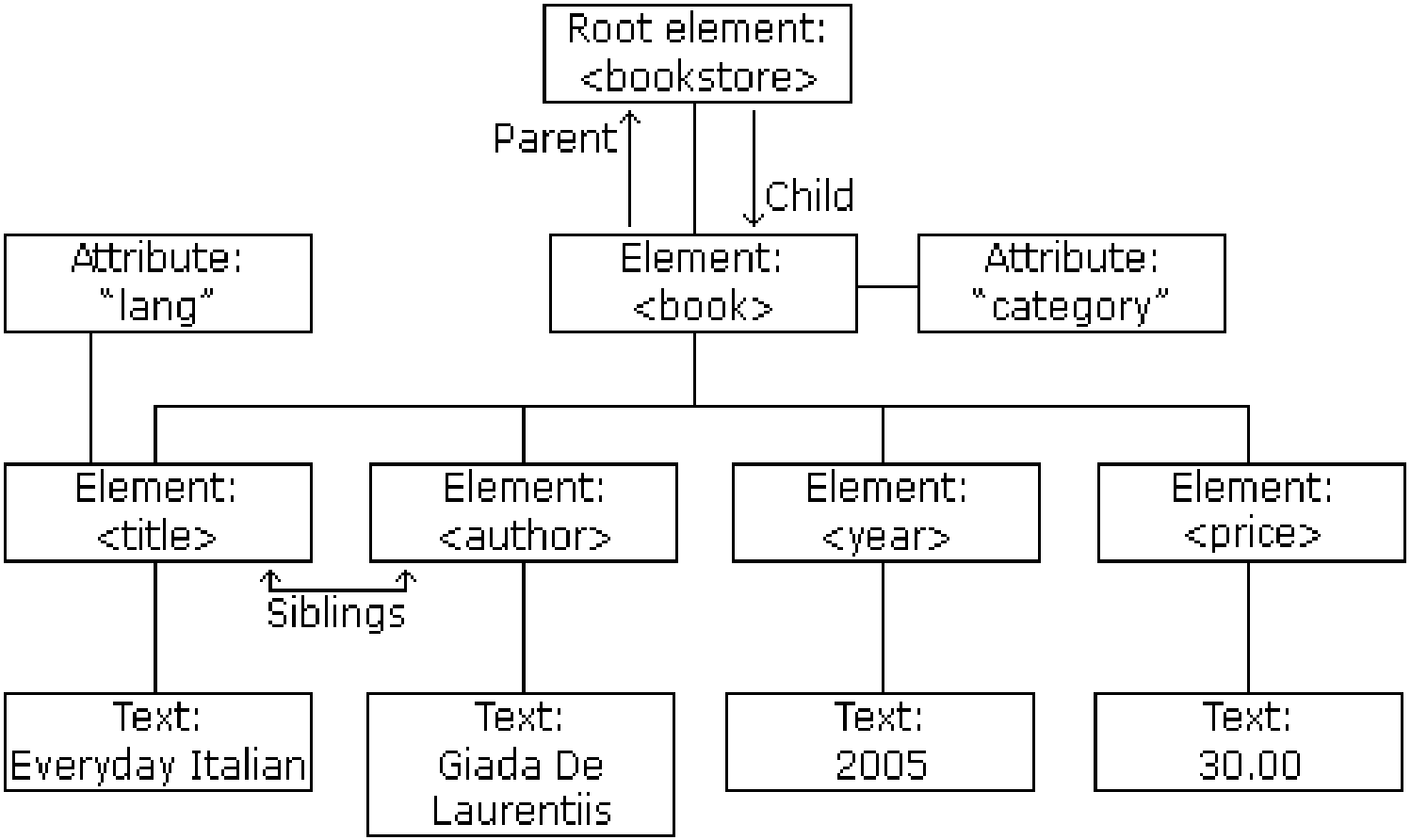
Simple XML Document

```
<?xml version="1.0" ?>
<home>
<Title> Example </Title>
<Text>
<message>
        This is simple XML document
 </message>
</Text>
</home>
```

XML Tree

```
<?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>
</bookstore>
```

XML Tree (Cont…)



XML Element

- An XML document contains XML Elements.
- **What is an XML Element?**
  - An XML element is everything from (including) the element's start tag to (including) the element's end tag.
  - An element can contain:
  - other elements
  - text
  - attributes
  - or a mix of all of the above…
  - An XML file must contain exactly one element in the top level which is know as root element

XML Element

- Elements with no content may be expressed as:
  - <NOTHING></NOTHING>
- In shorthand it may be expressed as:
  - <NOTHING/>
- Elements with no content may be used to display graphics and other material in the document.

XML Element

```
<bookstore>
  <book category="WEB">
   <title>Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price>39.95</price>
  </book>
</bookstore>
```

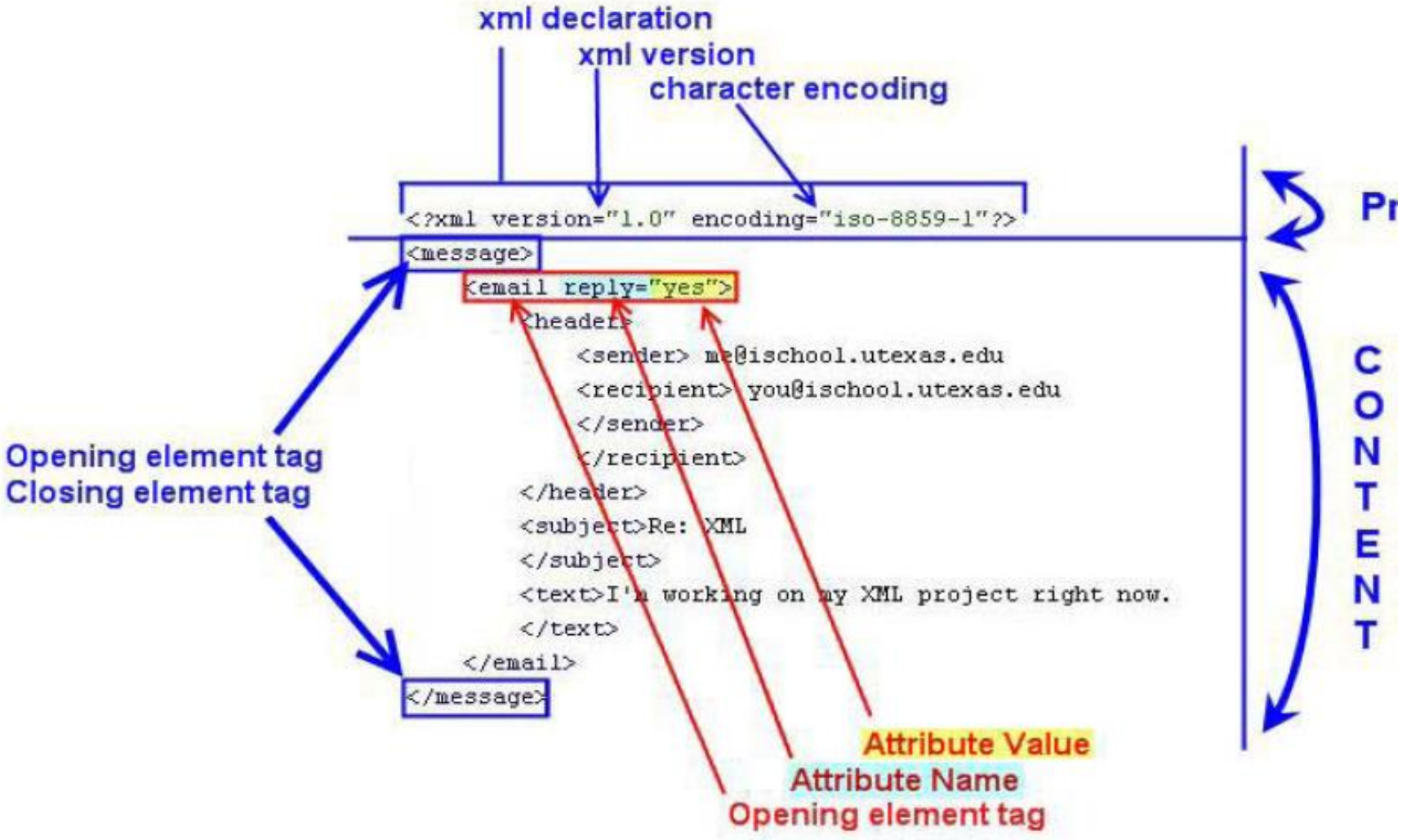
XML Naming Rules

- **XML Naming Rules**

  XML elements must follow these naming rules:

  - Names can contain letters, numbers, and other characters
  - Names cannot start with a number or punctuation character
  - Names cannot start with the letters xml (or XML, or Xml, etc)
  - Names cannot contain spaces
  - Any name can be used, no words are reserved.

XML Naming Rules



XML Attribute

```
<person name="personname">
        <firstname>abc</firstname>
        <lastname>xyz</lastname>
</person>
```

- attributes cannot contain multiple values (elements can)
- attributes cannot contain tree structures (elements can)
- attributes are not easily expandable

Namespace

- In XML, element names are defined by the developer, which may results into conflict when trying to mix XML documents from different XML applications having same element name

- HTML table information

```
<table>
        <tr>
                <td> data </td>
        </tr>
<table>
```

Namespace (cont…)

- Information about table

  <table>

  <name> student </name>

  <width> 100 </width>

  </table>

- When two fragments are added with same name, parser will not know how to handle this difference

Namespace (cont…)

- Solving name conflicts using prefix

```
<t1:table>
        <t1:tr>
        <t1:td> data </t1:td>
        </t1:tr>
<t1:table>
```

Namespace (cont…)

```
<t2:table>
        <t2:name> student </t2:name>
        <t2:name> width </t2:name>
</t2:table>
```

- When prefix is used in XML file, namespace for the prefix must be defined
- The namespace is defined by the **xmlns** attribute in the start tag of an element.

Namespace (cont…)

- Syntax: **xmlns:prefix="URI"**
- Uniform Resource Identifier (URI)
- A Uniform Resource Identifier (URI) is a string of characters which identifies an Internet Resource.
- The namespace URI is not used by the parser to look up information. The purpose is to give the namespace a unique name

Namespace (cont…)

```
<mytable>
  <t1:table xmlns:t1="http:// www.w3.org/TR/html4
  /">
    <t1:tr>
      <t1:td>student</t1:td>
      <t1:td>width</t1:td>
    </t1:tr>
  </t1:table>
  </mytable>
```

DTD & Schemas

- XML document must be well formed and correct in syntax
- Two ways to check whether document follows specific order and structure:
  - DTD
  - Schema

XML DTD

- DTD - Set of rules defining relationships within a document
- The purpose of a DTD is to define the structure of an XML document in terms of elements, tags, attributes, entity.
- XML DTD allows to create own DTD for applications, which gives you complete control over the process of checking content and structure of XML documents created for that application – **validation**

XML DTD (cont…)

- DTD can be declared in XML document in two ways:
  - Internal
  - External

XML DTD (cont…)

- **Internal DTD**

```
<?xml version="1.0" ?>
<!DOCTYPE mail [
<!Element mail (to, from, title, message)>
]>
```

XML DTD (cont…)

- **Internal DTD**

```
<mail>
        <to> abc </to>
        <from> xyz </from>
        <title> mail title </title>
        <message> simple message </message>
</mail>
```

DTD Element

- **Empty Elements**

  <!ELEMENT element-name EMPTY>

  Example:

  <!ELEMENT br EMPTY>

  XML example:

DTD Element

- **Elements with Parsed Character Data**

<!ELEMENT element-name (#PCDATA)>

Example:

<!ELEMENT message (#PCDATA)>

DTD Element

- **Declaring Only One Occurrence of an Element**

  <!ELEMENT element-name (child-name)>

  Example:

  <!ELEMENT mail (message)>

DTD Element

- **Declaring Minimum One Occurrence of an Element**

<!ELEMENT element-name (child-name+)>

Example:

<!ELEMENT mail (message+)>

DTD Element

- **Declaring Zero or More Occurrences of an Element**

<!ELEMENT element-name (child-name*)>

Example:

<!ELEMENT mail (message*)>

DTD Element

- **Declaring Zero or One Occurrences of an Element**

<!ELEMENT element-name (child-name?)>

Example:

<!ELEMENT mail (message?)>

DTD Element

- **Declaring either/or Content**

<!ELEMENT mail (to,from,header,(message|body))>

- **Declaring Mixed Content**

<!ELEMENT mail (#PCDATA|to|from|header|message)*>

DTD Element

- **Elements with any Contents**

<!ELEMENT element-name ANY>

Example:

<!ELEMENT mail ANY>

DTD Element

- **External DTD**

<!Element mail (to, from, title, message)>

<!Element to (#PCDATA)>

<!Element from (#PCDATA)>

<!Element title (#PCDATA)>

<!Element message (#PCDATA)>

PCDATA

- PCDATA - Parsed Character Data
- Parsed Character Data (PCDATA) is a term used about text data that will be parsed by the XML parser.
- <name>
  <first>Bill</first>
  <last>Gates</last>
  </name>

CDATA

- CDATA - (Unparsed) Character Data
- Everything inside a CDATA section is ignored by the parse
- Characters like "<" and "&" are illegal in XML elements.
- To avoid errors code can be defined as CDATA
- A CDATA section starts with "**<![CDATA[**" and ends with "]]>".

CDATA

- Example

```
<![CDATA[
        <name>
                <first>Bill</first>
                <last>Gates</last>
        </name>
]]>
```

CDATA

- Rules
  - A CDATA section cannot contain the string "]]>".
  - Nested CDATA sections are not allowed.
  - The "]]>" that marks the end of the CDATA section cannot contain spaces or line breaks.

XML DTD (cont…)

- External DTD

```
<?xml version="1.0"?>
<!DOCTYPE mail SYSTEM "mail.dtd">
<mail>
        <to> abc </to>
        <from> xyz </from>
        <title> mail title </title>
        <message> simple message </message>
</mail>
```

XML DTD (cont…)

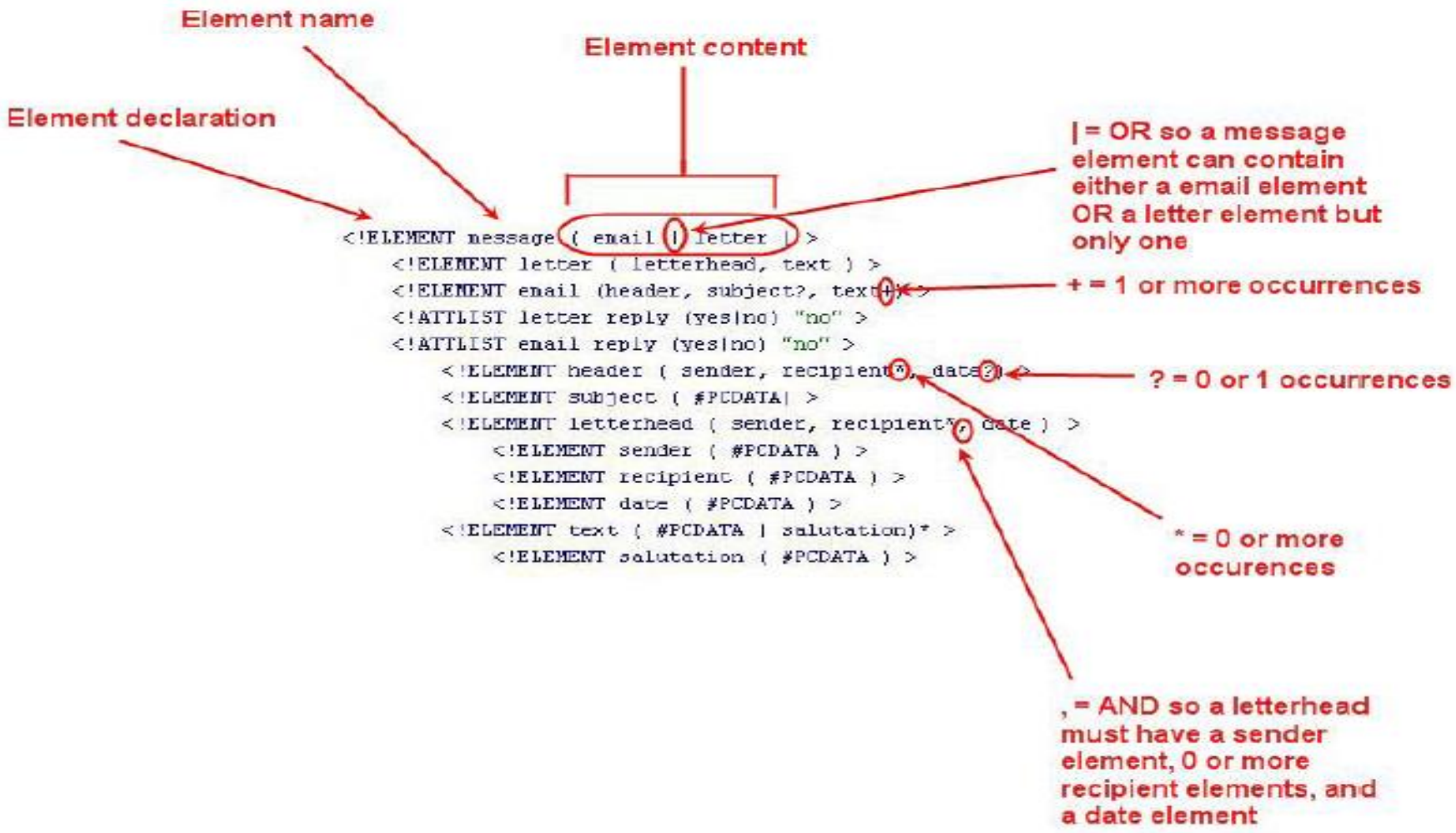
- First is an element followed by list of children elements and second is an element named message followed by parsable data.

<!Element mail (to, from, title, message)>

or

<!Element to (#PCDATA)>

XML DTD (cont…)


Element name
Element content
Element declaration
| = OR so a message element can contain either a email element OR a letter element but only one
+ = 1 or more occurrences
? = 0 or 1 occurrences
* = 0 or more occurences
, = AND so a letterhead must have a sender element, 0 or more recipient elements, and a date element
<!ELEMENT message (email | letter) >
<!ELEMENT letter (letterhead, text) >
<!ELEMENT email (header, subject?, text+) >
<!ATTLIST letter reply (yes|no) "no" >
<!ATTLIST email reply (yes|no) "no" >
<!ELEMENT header (sender, recipient*, date?) >
<!ELEMENT subject (#PCDATA| >
<!ELEMENT letterhead (sender, recipient*, date) >
<!ELEMENT sender (#PCDATA) >
<!ELEMENT recipient (#PCDATA) >
<!ELEMENT date (#PCDATA) >
<!ELEMENT text (#PCDATA | salutation)* >
<!ELEMENT salutation (#PCDATA) >

XML DTD (cont…)

- To declare attribute

  <!ATTLIST element-name attribute-name attribute-type default value>

- Ex:

  <!ATTLIST payment type PCDATA “card”>

- XML: <payment type=“card”>

Limitations of DTD

- DTD itself is not in xml format, so more work to be done by parser.
- Does not express data type (Weak data typing)
- No namespace support
- Document can override external DTD definition.
- No DOM support

XML Schema

- A schema is a description of the structure and rules a document must satisfy for an XML document type.
- It also includes the formal declaration of the elements that make up a document.
- If XML file does not obey the syntax of its associated DTD or Schema, it is not valid.

XML Schema

- XML Schema is a more sophisticated schema language which
- Addresses the drawbacks of DTDs. Supports
- Typing of values
  - E.g. integer, string, etc
  - Also, constraints on min/max values
  - User defined,
  - complex types
- Many more features, including
- Uniqueness and foreign key constraints, inheritance
- XML Schema is itself specified in XML syntax, unlike DTDs
- More standard representation, but verbose
- XML Scheme is integrated with namespaces
- BUT: XML Schema is significantly more complicated than DTDs.

XML Schema

The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.

An XML Schema:

- defines elements that can appear in a document
- defines attributes that can appear in a document
- defines which elements are child elements
- defines the order of child elements
- defines the number of child elements
- defines whether an element is empty or can include text
- defines data types for elements and attributes
- defines default and fixed values for elements and attributes

XML Schema

How XML Schema is advantageous over DTDs.

Here are some reasons:

- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML
- XML Schemas support data types
- XML Schemas support namespaces

XML Schema

- <xs:element name="note">

```
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name=“title" type="xs:string"/>
    <xs:element name=“message" type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

The <schema> Element

The <schema> element is the root element of every XML Schema.

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

<xs:schema>
…
…
</xs:schema>
```

XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/XMLSchema">
<xs:element name="note">
<xs:complexType>
  <xs:sequence>
    <xs:element name="to" type="xs:string"/>
    <xs:element name="from" type="xs:string"/>
    <xs:element name="title" type="xs:string"/>
    <xs:element name="message" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
</xs:element>
<xs:schema>
```

XML Schema

<xs:schema>

.....

.....

</xs:schema>

XML Schema

- Schema Element

<name>

<firstname> abc </firstname>

<lastname> abc </lastname>

</name>

<xs:element name="firstname" type=? />

<xs:element name="lastname" type=? />

XML Schema

- Element Type
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

<xs:element name="firstname" type="xs:string" />
<xs:element name="lastname" type="xs:string" />

XML Schema

- Schema Attribute

  <firstname lang="english"> abc </firstname>

- Schema Definition

  <xs:attribute firstname="lang" type="xs:string" />

XML Schema

- ‘fixed’ attribute in schema

  <xs:attribute firstname=“lang” type=“xs:string” fixed=“English” />

- ‘default’ attribute in schema

  <xs:attribute firstname=“lang” type=“xs:string” default=“English” />

XML Schema

- Schema attribute can be optional or required

<xs:attribute firstname="lang" type="xs:string" use="optional" />

<xs:attribute firstname="lang" type="xs:string" use="required" />

Example (.xml file)

<?xml version="1.0" encoding="UTF-8"

<page>

<head> <title> Home page </title> </head>

<body>

<title> Welcome to my page </title>

<message> Simple example </message>

</body>

</page>

DTD for .xml file

<!DOCTYPE page [

<!Element page (head, body) >

<!Element head (title) >

<!Element body (title, message)]

XML Data Islands

- XML data island is XML data embedded into an HTML page
- XML Data Islands only works with Internet Explorer browser

XML Data Islands

```
 <html>
<body>
<xml id="cdcat" src="catalog.xml"></xml>
<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="ARTIST"></span></td>
<td><span datafld="TITLE"></span></td>
</tr>
</table>
</body>
</html>
```

XML Data Islands

- datasrc
  - Refers to the name of data source object.
- datafld
  - Specifies data to which the element bound
- tags allow the datafld attribute to refer to the XML element to be displayed

XML Data Islands

- datasrc
  - Refers to the name of data source object.
- datafld
  - Specifies data to which the element bound
- tags allow the datafld attribute to refer to the XML element to be displayed