EXtensible Markup Language

XML

Main source: W3C School tutorials

Mark-up Languages

- A way of describing information in a document.
- Standard Generalized Mark-Up Language (SGML) - a specification for a mark-up language ratified in 1986.
- Key aspect - using pairs of tags that surround information - a begin tag <tag_name> and a matching end tag </tag_name>.

Example

<title> Grid Computing Home Page </title>

HyperText Markup Language (HTML)

A mark-up language used in web pages.

"Hypertext" refers to the text's ability to link to other documents.

"Markup" refers to providing information to tell browser how to display page and other things.

XML

- Very important standard mark-up language - a "simplified" SGML.
- Developed to represent textual information in a structured manner that could be read and interpreted by a computer.
- **A foundation for web services and grid services.**

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 tags are not predefined. We must **define our own tags**
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**
- XML is a W3C Recommendation

The main difference between XML and HTML

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

XML vs SGML

- SGML:
  - SGML is the Standard Generalized Markup Language
  - Not well suited to serving documents over the web
  - Hard for browsers to handle

- XML specifies neither semantics nor a tag set.
  - meta-language for describing markup languages
  - Restricted form of SGML

XML Application Areas

- Used for two applications areas:
  - Document-centric XML
  - Data-centric XML

Document-Centric XML

- Documents usually meant for humans, although could be processed by computers.
- Semi-structured - some tags can be placed more-or-less anywhere, similar to HTML tags.

Data-Centric XML

- Usually generated and meant to be read by computer programs.
- Structured.
- Nesting useful to create a clearly structured and computer-readable document.

Sample data-centric XML

```
<poid= "53912" submitted= "2/13/2015">
<billTo>
   <name>Department of Computer and Information
   Sciences</name>
   <company>University of Hyderabad</company>
   <street>Gochibowli</street>
   <city>Hyderabad</city>
   <state>AP</state>
   <postalCode>500046</postalCode>
</billTo>
```

XML does not DO anything

- XML was not designed to DO anything.
- XML was created to structure, store and to send information.
- The following example is a note to Murali from Bala, stored as XML:

```
<note>
    <to>Bala</to>
    <from>Murali</from>
    <heading>Reminder</heading>
    <body>Bring chocolates for me</body>
</note>
```

XML is free and extensible

- XML tags are not predefined. We must "invent" our own tags.
- The tags used in HTML documents are predefined. We can only use tags that are defined in the HTML standard.
- XML allows the author to define his own tags and his own document structure.
- The tags in the last example (like <to> and <from>) are not defined in any XML standard. These tags are "invented" by the author of the XML document.

XML is a complement to HTML

- **XML is not a replacement for HTML.**
- In Web development XML is used to describe the data, while HTML is used to format and display the same data.
- XML is a cross-platform, software and hardware independent tool for transmitting information.
- It is important to remember that XML was designed to store, carry, and exchange data. XML was not designed to display data.

XML can Separate Data from HTML

- With XML, our data is stored outside the HTML.
- When HTML is used to display data, the data is stored inside our HTML. With XML, data can be stored in separate XML files.
- XML data can also be stored inside HTML pages as "Data Islands". We can still concentrate on using HTML only for formatting and displaying the data.

XML is used to Exchange Data

- **With XML, data can be exchanged between incompatible systems.**
- In the real world, data lie in incompatible formats. One of the most time-consuming challenges for developers has been to exchange data between such systems over the Internet.
- Converting the data to XML can greatly reduce this complexity and create data that can be read by many different types of applications.

XML can be used to Share Data

- Since XML data is stored in plain text format, XML provides a software- and hardware-independent way of sharing data.
- This makes it much easier to create data that different applications can work with. It also makes it easier to expand or upgrade a system to new operating systems, servers, applications, and new browsers.

XML can be used to Store Data

- **With XML, plain text files can be used to store data.**
- XML can also be used to store data in databases. Applications can be written to store and retrieve information from the store, and generic applications can be used to display the data.

XML-Databases

- Two major classes of XML database exist:
- **XML-enabled**: these may either map XML to traditional database structures (such as a relational database), accepting XML as input and rendering XML as output, or more recently support native XML types within the traditional database. This term implies that the database processes the XML itself (as opposed to relying on middleware).
- **Native XML** (NXD): the internal model of such databases depends on XML and uses XML documents as the fundamental unit of storage, which are, however, not necessarily stored in the form of text files.

XML enabled

- XML enabled databases typically offer one or more of the following approaches to storing XML within the traditional relational structure:
  - XML is stored into a CLOB (Character large object)
  - XML is `shredded` into a series of Tables based on a Schema
  - XML is stored into a native XML Type as defined by the ISO
- RDBMS that support the ISO XMLType are:
  - IBM DB2 (pureXML)
  - Microsoft SQL Server
  - Oracle Database
  - PostgreSQL

Native XML Databases

- BaseX
- eXist
- MarkLogic
- Sedna

XML can be used to Create new Languages

- **XML is the mother of WAP and WML.**
- The Wireless Markup Language (WML), used to markup Internet applications for devices like mobile phones, is written in XML.

Element Naming

XML elements must follow these naming rules:

- Names can contain letters, numbers, and other characters
- Names must not start with a number or punctuation character
- Names must not start with the letters xml (or XML, or Xml, etc)
- Names cannot contain spaces
- The "**:**" should not be used in element names because it is reserved to be used for something called namespaces (more on it in later slides)

An example XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML and uses the ISO-8859-1 (Latin-1/West European) character set.

- The next line describes the root element of the document (indicating: "this document is a note"):

An example XML document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

- The next 4 lines describe 4 child elements of the root (to, from, heading, and body):

- And finally the last line defines the end of the root element:

All XML elements must have a closing 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
```

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

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

- *IMP: XML declaration doesn't have a closing tag. This is not an error. The declaration is not a part of the XML document itself. It is not an XML element, and it should not have a closing tag.*

XML tags are case sensitive

- **Unlike HTML, XML tags are case sensitive.**
- With XML, the tag <Letter> is different from the tag <letter>.
- Opening and closing tags must therefore be written with the same case
- **All XML elements must be properly nested**
- Improper nesting of tags makes no sense to XML.

```
<b><i>This text is bold and italic</b></i>
```

- Valid in HTML but not in XML

All XML documents must have a root element

- **All XML documents must contain a single tag pair to define a root element (need not named as root)**
- All other elements must be within this root element.
- All elements can have sub elements (child elements). Sub elements must be correctly nested within their parent element

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

Attribute values must always be quoted

- **With XML, it is illegal to omit quotation marks around attribute values.**

- XML elements can have attributes in name/value pairs just like in HTML.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<note date=12/11/2002>

<to>Tove</to>

<from>Jani</from>

</note>
```

Some other facts

- The syntax for writing comments in XML is similar to that of HTML.

```
<!-- This is a comment -->
```

- Unlike HTML, the white space in XML document is not truncated.
- With XML, CR / LF is converted to LF.
- With XML, a new line is always stored as LF.

XML Elements are Extensible

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <body>Don't forget me this weekend!</body>
</note>
```

```
MESSAGE
To: Tove
From: Jani
Don't forget me this weekend!
```

```
<note>
    <date>2002-08-01</date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Elements are related as parents and children.

My First XML

Introduction to XML

- What is HTML
- What is XML

XML Syntax

- Elements must have a closing tag
- Elements must be properly nested

Elements are related as parents and children.

```
<book>
   <title>My First XML</title>
   <prod id="33-657" media="paper"></prod>
   <chapter>Introduction to XML
      <para>What is HTML</para>
      <para>What is XML</para>
   </chapter>

   <chapter>XML Syntax
       <para>Elements must have a closing tag</para>
       <para>Elements must be properly nested</para>
   </chapter>

</book>
```

Root element

Child element of book

Elements have Content

- An **XML element** is everything from (including) the element's start tag to (including) the element's end tag.
- An element can have **element** content, **mixed** content, **simple** content, or **empty** content. An element can also have **attributes**.
- In the example presented, book has **element content**, because it contains other elements. Chapter has **mixed content** because it contains both text and other elements.

Elements have Content

- para has **simple content** (or **text content**) because it contains only text. prod has **empty content**, because it carries no information.

- In the example above only the prod element has **attributes**. The **attribute** named id has the **value** "33-657". The **attribute** named media has the **value** "paper".

Elements are related as parents and children.

```
<book>
   <title>My First XML</title>
   <prod id="33-657" media="paper"></prod>
   <chapter>Introduction to XML
      <para>What is HTML</para>
      <para>What is XML</para>
   </chapter>

   <chapter>XML Syntax
       <para>Elements must have a closing tag</para>
       <para>Elements must be properly nested</para>
   </chapter>

</book>
```

Element content

Empty content

Mixed content

Simple or text content

Attribute id has value

XML Attributes

- **XML elements can have attributes.**

- In XML attributes provide additional information about elements:

```
<img src="computer.gif">
<a href="demo.asp">
```

- Attribute values must always be enclosed in quotes, but either single or double quotes can be used. For a person's sex, the person tag can be written like this:

```
<person sex="female">      or like this:
<person sex='female'>
```

Use of Elements vs. Attributes

- Data can be stored in child elements or in attributes.

```
<person sex="female">                                    Attribute
        <firstname>Sonia</firstname>
        <lastname>Gandhi</lastname>
</person>
```

Example

```
<note date="12/11/2002">
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
</note>

<note>
       <date>12/11/2002</date>
       <to>Tove</to>
       <from>Jani</from>
       <heading>Reminder</heading>
       <body>Don't forget me this weekend!</body>
</note>
```

Example

```
<note>
    <date>
        <day>12</day>
        <month>11</month>
        <year>2002</year>
    </date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Should we avoid using attributes?

Some of the problems with using attributes are:

- attributes cannot contain multiple values (child elements can)
- attributes are not easily expandable (for future changes)
- attributes cannot describe structures (child elements can)
- attributes are more difficult to manipulate by program code
- attribute values are not easy to test against a XML Schema or Document Type Definition (DTD) - which is used to define the legal elements of an XML document
- Exception: an unique identifier that is not a part of the note data in the XML file may be an attribute.

"Valid" & "Well Formed" XML documents

- A "Well Formed" XML document has correct XML syntax.
- A "Valid" XML document also conforms to a DTD.
- A "Valid" XML document is a "Well Formed" XML document, which also conforms to the rules of a **DTD or Schema**:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "InternalNote.dtd">
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Well Formed XML Document

- Contains exactly one root element (the document element), and all the child elements are nested properly within each other.

```
<?xml version="1.0"?>
<pizzas>
  <pizza highfat="dream on">
    <name>DoubleCheeze</name>
    <description>Greasy and good.</description>
    <price>150</price>
  </pizza>
</pizzas>
```

XML Document Schema

- Database schemas constrain what information can be stored, and the data types of stored values
- XML documents are not required to have an associated schema
- However, schemas are very important for XML data exchange
  - Otherwise, a site cannot automatically interpret data received from another site
- Two mechanisms for specifying XML schema
  - **Document Type Definition (DTD)**
    - Widely used earlier
  - **XML Schema**
    - Increasing use

Document Type Definition (DTD)

- The type of an XML document can be specified using a DTD
- DTD constraints structure of XML data
  - What elements can occur
  - What attributes can/must an element have
  - What subelements can/must occur inside each element, and how many times.
- DTD does not constrain data types
  - All values represented as strings in XML
- DTD syntax
  - <!ELEMENT element (subelements-specification) >
  - <!ATTLIST element (attributes) >

Element Specification in DTD

- Sub-elements can be specified as
  - names of elements, or
  - #PCDATA (parsed character data), i.e., character strings
  - EMPTY (no subelements) or ANY (anything can be a subelement)
- Example

  <! ELEMENT depositor (customer_name account_number)>
  <! ELEMENT customer_name (#PCDATA)>
  <! ELEMENT account_number (#PCDATA)>
- Subelement specification may have regular expressions

  <!ELEMENT bank ((account | customer | depositor)+)>
  - Notation:
    - **“|” - alternatives**
    - **“+” - 1 or more occurrences**
    - **“*” - 0 or more occurrences**

Source: Silberschatz, Korth and Sudarshan

XML Schema

- A DTD defines the legal elements of an XML document.
- The purpose of a DTD is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.
- XML Schema is an XML based alternative to DTD.

XML Primitive Data Types

Binary	Base 2 number
Boolean	True or false value
decimal	Base 10 number
double	64-bit float
float	32 bit, base 10 number
recurringDuration	A period of time that is recurring
string	Finite sequence of chars
timeDuration	Some duration of time
uriReference	Network or system resource reference

A general XML Viewer

- Raw XML files can be viewed in Mozilla, Firefox, Opera, Internet Explorer, and in Netscape 6+.
- However, to make XML documents to display like nice web pages, you will have to add some display information.
- With CSS (Cascading Style Sheets) you can add display information to an XML document.
- Alternatively, with eXtensible Stylesheet Language (XSL) we can add display information to our XML document (W3C's XSL standard)

Querying and Transforming XML Data

- Translation of information from one XML schema to another and Querying on XML data are closely related, and handled by the same tools

- Followings are the XML querying/translation languages (based on the tree model of XML)
  - XPath
    - Simple language consisting of path expressions
  - XQuery
    - An XML query language with a rich set of features

XPath

- XPath is used to select parts of documents using **path expressions**
- A path expression is a sequence of steps separated by "/"
  - Similar to the file name path
- Result of path expression is the set of values that along with their containing elements/attributes matching the specified path
- E.g. /bank-2/account/balance evaluated on the bank-2 data returns
  <balance> 500 </balance>

- E.g. /bank-2/account/balance/text( )
  returns the same names, but without the enclosing tags

XPath (Cont.)

- The initial "/" denotes root of the document (above the top-level tag)
- Path expressions are evaluated left to right
  - Each step operates on the set of instances produced by the previous step
- Selection predicates may follow any step in a path, in [ ]
  - E.g. /bank-2/account[balance > 400]
    - returns account elements with a balance value greater than 400
- Attributes are accessed using "@"
  - E.g. /bank-2/account[balance > 400]/@account_number
    - returns the account numbers of accounts with balance > 400

Functions in XPath

- XPath provides several functions
  - The function count() at the end of a path counts the number of elements in the set generated by the path
    - E.g. /bank-2/account[count(./customer) > 2]
      - Returns accounts with > 2 customers
  - Also function for testing position (1, 2, ..) of node w.r.t. siblings
- Boolean connectives and and or and function not() can be used in predicates

More XPath Features

- Operator "|" used to implement union
  - E.g. /bank-2/account/id(@owner) | /bank-2/loan/id(@borrower)
    - Gives customers with either accounts or loans
    - However, "|" cannot be nested inside other operators.
- "//" can be used to skip multiple levels of nodes
  - E.g. /bank-2//customer_name
    - finds any customer_name element *anywhere* under the /bank-2 element, regardless of the element in which it is contained.
  - "//", described above, is a short from for specifying "all descendants"
  - ".." specifies the parent.

Xquery (W3C Standard)

- XQuery is a general purpose query language for XML data
- It is derived from the Quilt query language, which itself borrows from SQL, XQL and XML-QL
- XQuery uses a

  for … let … where … order by …return …

- Syntax

for	⇔ SQL **from** [Series of variable]
where	⇔ SQL **where**
order by	⇔ SQL **order by**
return	⇔ SQL **select**
let	allows temporary variables := complicated expression

FLWOR Syntax in XQuery

- Simple FLWOR expression in XQuery
  - find all accounts with balance > 400, with each result enclosed in an <account_number> .. </account_number> tag

```
for     $x in /bank-2/account
let     $acctno := $x/@account_number [let gives variable]
where $x/balance > 400
return <account_number> { $acctno } </account_number>
```

  - Items in the **return** clause are XML text unless enclosed in {}, in which case they are evaluated

- If done In XPath. Query can be written as:

```
for $x in /bank-2/account[balance>400]
return <account_number> { $x/@account_number }
</account_number>
```

Viewing XML Files

In Firefox and Internet Explorer

- Open the XML file - The XML document will be displayed with color-coded root and child elements. A plus (+) or minus sign (-) to the left of the elements can be clicked to expand or collapse the element structure. To view the raw XML source (without the + and - signs), select "View Page Source" or "View Source" from the browser menu.

In Netscape 6

- Open the XML file, then right-click in XML file and select "View Page Source". The XML document will then be displayed with color-coded root and child elements.

XML parser

- To read and update - create and manipulate - an XML document, you will need an XML parser
- Microsoft's XML parser (MSXML Parser 3.x) is a COM component (Explorer 6.0 and higher & XP). Once Internet Explorer is installed, the parser is available to scripts.
- Microsoft's XML parser supports all the necessary functions to traverse the node tree, access the nodes and their attribute values, insert and delete nodes, and convert the node tree back to XML.

XML parser

- Most browsers have a build-in XML parser to read and manipulate XML.
- The parser converts XML into a JavaScript accessible object.
- The parser reads XML into memory and converts it into an XML DOM object that can be accesses with JavaScript.
- http://www.w3schools.com/xml/xml_parser.asp

XML Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- Since element names in XML are not predefined, a name conflict will occur when two different documents use the same element names

XML Namespaces

- This XML document carries information in a table:

```
<table>
    <tr>
      <td>Apples</td>
      <td>Bananas</td>
    </tr>
</table>
```

- This XML document carries information about a table (a piece of furniture):

```
<table>
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

If these two XML documents are added together, there would be an element name conflict

Solving Name Conflicts using a Prefix

```
<h:table>
     <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
     </h:tr>
</h:table>

<f:table>
     <f:name>African Coffee Table</f:name>
     <f:width>80</f:width>
     <f:length>120</f:length>
</f:table>
```

Using Namespaces

```
<h:table xmlns:h="http://www.w3.org/TR/html4/">
    <h:tr>
        <h:td>Apples</h:td>
        <h:td>Bananas</h:td>
    </h:tr>
</h:table>

<f:table xmlns:f="http://www.w3s.com/furniture" >
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
</f:table>
```

Instead of using only prefixes, we have added an xmlns attribute to the <table> tag to give the prefix a qualified name associated with a namespace.

The XML Namespace (xmlns) Attribute

- The XML namespace attribute is placed in the start tag of an element and has the following syntax:

  xmlns:*namespace-prefix*="*namespaceURI*"

- All child elements with the same prefix are associated with the same namespace defined in the start tag.

- *"Address used to identify the namespace is not used by the parser to look up information. The only purpose is to give the namespace a unique name. However, very often companies use the namespace as a pointer to a real web page containing information about the namespace"*

Uniform Resource Identifier (URI)

- A **Uniform Resource Identifier** (URI) is a string of characters which identifies an Internet Resource. The most common URI is the **Uniform Resource Locator** (URL) which identifies an Internet domain address.

Default Namespaces

- Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:

xmlns="*namespaceURI*"

Uniform Resource Identifier (URI)

```
<table xmlns="http://www.w3.org/TR/html4/">
    <tr>
        <td>Apples</td>
        <td>Bananas</td>
    </tr>
</table>

<table xmlns="http://www.w3s.com/furniture" >
    <name>African Coffee Table</name>
    <width>80</width>
    <length>120</length>
</table>
```

Storing XML on the Server

- XML files can be stored on an Internet server exactly the same way as HTML files.
- Start Windows Notepad and write the following lines:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<from>Jani</from>
<to>Tove</to>
    <message>Remember me this weekend </message>
</note>
```

- Save the file on your web server with a proper name like "note.xml".

Some facts

- XML can be generated from a database without any installed XML software.
- One can use ASP and Microsoft's XMLDOM object to create and save the XML file.
- One can be benefited from using a professional XML Editor.

Why an XML Editor?

- XML Schema to define XML structures and data types
- XSLT to transform XML data
- SOAP to exchange XML data between applications
- WSDL to describe web services
- RDF to describe web resources
- XPath and XQuery to access XML data
- SMIL (Synchronized Multimedia Integration Language) to define graphics

Altova's XMLSPY

- A Professional XML editors that will help you to write error-free XML documents, validate your XML against a DTD or a schema, and force you to stick to a valid XML structure.

An XML editor helps you to:

- Add closing tags to your opening tags automatically
- Force you to write valid XML
- Verify your XML against a DTD
- Verify your XML against a Schema
- Color code your XML syntax

Serna Free, oXygen are other editors to look for

XML DOM

- The XML DOM defines a standard way for accessing and manipulating XML documents.

XSLT

- XSLT is the style sheet language for XML files.
- With XSLT you can transform XML documents into other formats, like XHTML.

DTD and XML Schema

- The purpose of a DTD/Schema is to define what elements, attributes and entities is legal in an XML document.

XML Schema has almost replaced DTD.