XML

OBJECTIVES

- What is XML
- XML data model
- XML Namespace description
- XSL

WHAT IS XML

EXtensible Markup Language

A framework for defining other markup languages

Defined by W3C http://www.w3.org/TR/xml11
standardized in 1998

Useful as a mechanism to transport and store data

XHTML is an XML variant of HTML

Platform independent, license-free, and internationalized (Unicode)

Basically a meta-language - XML tags are not predefined. You define your own tag names.

XML AND HTML

XML is not a replacement for HTML

XML and HTML have different goals:

- XML describes data and focuses on what the data represent
- HTML structures browser content

XML IS EXTENSIBLE

Tag names which markup HTML documents and the structure of HTML documents are predefined. The author of HTML documents can only use tag names that are defined in the HTML standard (e.g. body, p, a)

XML allows you to define your own tag names and your own XML document structure

XML is a complement to HTML, a descendant of SGML (standard generalized markup language)

XML can be used to structure and describe data used by web technologies

HOW XML IS USED

How can XML be useful?

XML can:

- keep data separate from HTML
- serve as a mechanism to exchange information aka *data serialization – converting complex data objects into bit sequences*
- store data in human readable files or in databases

XML TECHNOLOGIES

Define syntax of languages

- DTD, XML Schema, XHTML, Office Open XML

Display data in browsers

- XPath, XSLT

Store information in databases

- XQuery, RDF

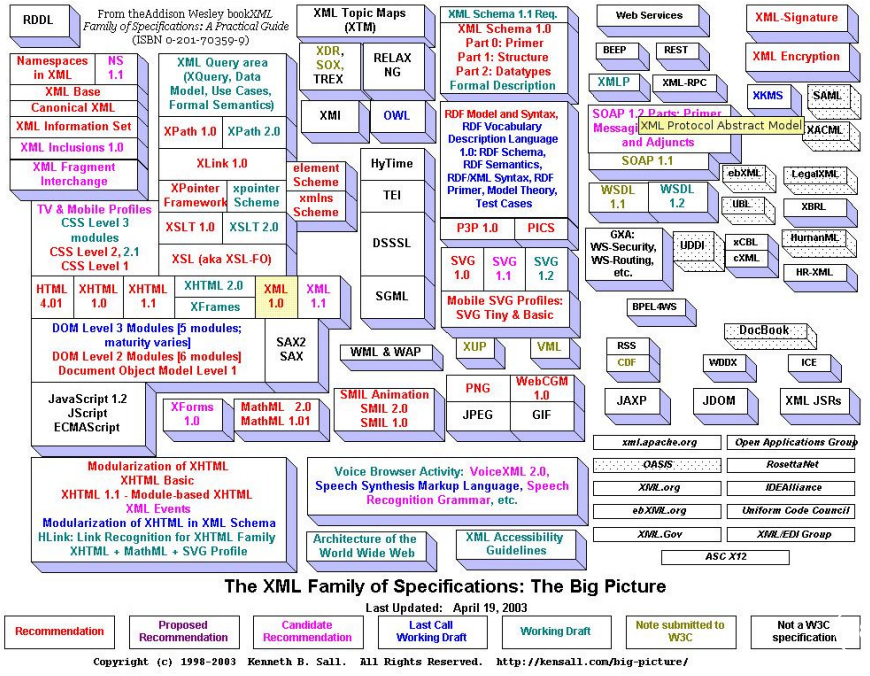
Facilitate web services data, podcasting

- WSDL, SOAP, RSS, Atom, AJAX, mashups

Platform independent program configuration

- **Apache Ant**

- **Image formats**
  - **SVG (Scalable Vector Graphics)**
- **Online forms**
  - **XForms**



```
<?xml version="1.0"?>
<note>
  <to> John </to>
  <from> Sally </from>
  <heading> Reminder </heading>
  <body> Do you have my book?
  </body>
</note>
```

XML documents should begin with an XML declaration.

XML documents use the .xml file extension as in **note.xml**

XML content is **free-format**

WHEN TO USE XML-BASED FORMAT

- Data is structured into a hierarchy form
- Need for a wide range of tools on different platforms
- Need for data that can 'outlive' the applications that process it
- Supports internationalization
- Need for human-readable text content
- Possible use with other XML-encoded formats

SAMPLE XML DOCUMENT #2

```
<?xml version="1.0"?>
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate zombies, an evil sorceress,
      and her own childhood to become queen of the world.</description>
  </book>
</catalog>
```

XML data must start with a **single** "root" or **document entity**, here, it is <catalog>

XML consists of **elements** which are enclosed by its start tag and its end tag. e.g. author, title, genre, price, etc

The characters between the start and end tags, if any, are the element's **content**.

XML TERMINOLOGY

Element

- A logical component of an XML document which either begins with a start-tag and ends with its matching end-tag (e.g. <price>13.50</price>), or consists only of an empty element tag, (e.g. <toppings />)
- The content within the element may contain markup, including other elements, which are then called child elements.

XML TERMINOLOGY

Markup

- XML documents are divided into markup and content
- Markup is any text that either:
  - starts with **<**
  - or, ends with **>**
  - or, begins with **&** and ends with **;**
- Content is anything else (basically the data within the markup)

XML TERMINOLOGY

Attribute

- A markup construct consisting of a name/value pair existing inside the tag (e.g. for the start tag <book bookid= 'bk101' > the name/value pair attribute is bookid/ 'bk101')
- Attributes may be optional
- Attribute value must be in matching single or double quotes
- More than one attribute may be defined
  - order of attribute definitions does not matter

XML TERMINOLOGY

Encoding

- Defines the character set used in the XML document
- Usually it is UTF-8 (8-bit Unicode Transformation Format)

see www.unicode.org

- Backward-compatible with ASCII (one byte per character e.g. 'a' is encoded as U+0061)
- Variable length character encoding
- Can encode any Unicode character, such as those used by other languages e.g. U+8349 is 草 or "grass" in "Mandarin (simplified)"

XML TERMINOLOGY

Processing Instruction (PI)

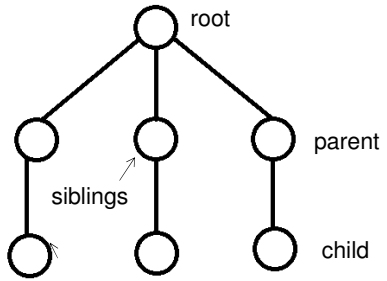
- tells a program to perform a specific task
- a PI begins with "<?" and ends with "?>"
- the XML declaration is also a processing instruction

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

XML TREE

Conceptually, an XML document is structured like a tree

- Node
- Root
- Child, parent
- Sibling



NODES IN XML TREES

Text nodes: the content between tags

Element nodes: the start and end tags

Attribute nodes: any name= “value” inside a start tag

Comment nodes: ignored by the processor

Processing instructions: <?target value?>

Root node: the XML tree has one root node

SAMPLE XML FILE #3

```
- <CATALOG>
  - <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
    </CD>
  - <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
    </CD>
  - <CD>
      <TITLE>Greatest Hits</TITLE>
      <ARTIST>Dolly Parton</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>RCA</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1982</YEAR>
```

- http://www.w3schools.com/XML/cd_catalog.xml

XML SYNTAX

Proper XML documents are *well-formed* (breaks no tag rules) and *valid* (XML data checked against its own DTD rules).

An XML file must be well-formed before it can be validated. A well-formed XML document obeys the syntax of an XML document.

A *broken* XML file is either not well-formed or not valid or both.

XML SYNTAX

In HTML some elements are not required to have a closing tag. The following is legal in HTML v4:

```
<p>It was a dark and stormy night.
<p>Susan went to the kitchen.
```

But in XML all elements must have a closing tag :

```
<p>It was a dark and stormy night.</p>
<p>Susan went to the kitchen.</p>
```

XML tags are case sensitive.

The tag <Letter> is different from the tag <letter>

ELEMENT NAME

Element names can contain letters, numbers, periods, underscores, hyphens, but not spaces

Element names must be defined uniquely within the XML document (i.e. no name duplication)

Element name must start with a letter or underscore, but cannot start with reserved words such as `xml`

Acceptable element name: bookstore

Unacceptable: 100book, $author, cost per

XML SYNTAX

Start and end tags must be written with the same case:

<Message> not well-formed </message>

<message> is well-formed </message>

All XML elements must be properly nested

In HTML some elements can be improperly nested within each other like this:

  This text is bold and italic

In XML all elements must be properly nested within each other like this

  This text is bold and italic

XML SYNTAX

Empty tags may use the backslash (similar to XHTML <hr /> and

It is legal in XML to use a start-tag/end-tag pair for empty tags <note></note>

XML comments similar to HTML

<!-- -->

BROKEN XML DOCUMENT – FIND THE ERRORS

```
<?xml version="1.0"?>
<note>
<heading>
<to> John
<from> Sally
 Reminder </heading> </from>
<body>Do you have my book?</body>
<date sent>May 30, 2010</date sent>
</Note>
```

Missing end tag </to>

Improper tag nesting

Unmatched tag name

Element name contains space

WELL-FORMED XML

The main reason you need well-formed XML is that an XML parser program reads the XML and generates a structure like a tree which represents the XML document.

Each "branch" of the tree must be properly defined so it can be examined.

If a tag is missing or unbalanced, the XML parser cannot create the tree.

XML VALIDATION

A valid XML document is a Well Formed XML document which conforms to the rules and constraints of a Document Type Definition (DTD)

Also called a schema or a grammar

The purpose of a DTD is to define the legal building blocks of an XML document

XML does not require a DTD but it defines the document structure with a list of legal elements.

A DTD can be declared inline in your XML document, or as an external reference.

INTERNAL DTD

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note     (to,from,heading,body)>
<!ELEMENT to       (#PCDATA)>
<!ELEMENT from     (#PCDATA)>
<!ELEMENT heading  (#PCDATA)>
<!ELEMENT body     (#PCDATA)>
]>
<note>
<to>John</to>
<from>Sally</from>
<heading>Reminder</heading>
<body>Do you have my book?</body>
</note>
```

Element note must contain the following children in this specific order.

PCDATA means parsed character data – text found between the start and end tags – tags inside the text will be treated as markup and entities will be expanded.

EXTERNAL DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
<note>
<to>John</to>
<from>Sally</from>
<heading>Reminder</heading>
<body>Do you have my book?</body>
</note>
```

The file note.dtd contains

```
<?xml version="1.0"?>
<!ELEMENT note
(to,from,heading,body
)>
<!ELEMENT to
(#PCDATA)>
<!ELEMENT from
(#PCDATA)>
<!ELEMENT heading
(#PCDATA)>
<!ELEMENT body
(#PCDATA)>
```


INVALID XML FILE EXAMPLE... WHY?

```
<?xml version="1.0"?>
<!DOCTYPE note [
<!ELEMENT note     (to,from,heading,body)>
<!ELEMENT to       (#PCDATA)>
<!ELEMENT from     (#PCDATA)>
<!ELEMENT heading  (#PCDATA)>
<!ELEMENT body     (#PCDATA)>
]>
<note>
<from>Sally</from>
<to>John</to>
<heading>Reminder</heading>
<sent>Today</sent>
<body>Do you have my book? </body>
</note>
```

Start and end tags for <to>…</to> appear after <from>…</from> tags.

Tag <sent> not declared in the note element

ELEMENT TYPE DECLARATIONS

Element type declarations identify the names of elements and the nature of their content

```
<!ELEMENT meal (appetizer+ entree dessert?)>
<!ELEMENT appetizer     (#PCDATA)>
<!ELEMENT entree        (#PCDATA)>
<!ELEMENT dessert       (#PCDATA)>
```

- The commas between element names indicate that they must occur in succession.
- The plus after appetizer indicates that it may be repeated more than once but must occur at least once.
- The question mark after dessert indicates that it is optional (it may be absent, or it may occur exactly once). An asterisk * means 0 or more.
- A name with no punctuation, such as entree, must occur exactly once.

ATTRIBUTE LIST DECLARATIONS

Identify: Which elements may have attributes? What attributes they may have? What values may the attributes hold? What value is the default?

```
<!ATTLIST dessert
  name      ID     #REQUIRED
  label     CDATA  #IMPLIED
  category  ( cake | pie | ice-cream )
  'pie'>
```

#IMPLIED means label does not have to be included

category choices for dessert – pie is the default

ATTRIBUTE DECLARATIONS

Another example:

```
<!ATTLIST  IMG
      src      %URI;     #REQUIRED
      name     CDATA     #IMPLIED
      id       ID        #IMPLIED
      class    CDATA     #IMPLIED
      alt      %Text;    #REQUIRED
  >
```


XML ENTITIES

Entities are shortcuts to content text

```
<!ENTITY  restaurant  "ABC Steak House">
<!ENTITY  owners   "Bill and Sue Smith">
<!ENTITY  menu  SYSTEM
  "http://www.abcSteakHouse.com/entities/
                  entities.xml">

  <dining> &restaurant; &owners; &menu;
  </dining>
```


WHY USE A DTD?

- XML provides an application independent way of sharing data.
- With a DTD, independent groups of people can agree to use a common DTD for interchanging data.
- Your application can use a standard DTD to verify that data that you receive from the outside world is valid.
- You can also use a DTD to verify your own data

DTD ALTERNATIVE

- XSD (XML Schema Document) is a W3C technology for defining XML schemas (rules)
- Unlike DTD XSD has namespace awareness and can use data types such as string, boolean, float, date, time, etc – 25 derived data types in all
- All the named schema components belong to a target namespace, and the target namespace is a property of the schema

DISPLAYING XML

- IE and Firefox have differing approaches to displaying XML.
- Can display XML data inside an HTML page by using JavaScript to import data from an XML file.
- Can also use CSS files with XML, but XSL files (Extensible Style Language) are the better way to go

XSD EXAMPLE

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema elementFormDefault="qualified“
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element name="PostCode" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```


XML FORMAT

Notice that at this point nothing has been said about the format of the final document.

From the neutral format provided by XML users can either chose to display the memo:

- On a screen, the size can be varied to suit user preferences,
- To print the text onto a pre-printed form,
- ...o r to generate a completely new form, positioning each element of the document where needed.

XML VALIDATION

http://www.xmlvalidation.com

Other XML schema validators

http://www.w3schools.com/dom/dom_validate.asp

http://www.validome.org/xml/

XML NAMESPACE

You may want to use the same tag name or attribute for different types of information

HTML tag <table> vs something else that uses the same tag name

XML uses *namespace* to differentiate

Defining the namespace:

xmlns:*prefix = 'namespace identifier'*

XML NAMESPACE

```
<meal xmlns:d='http://www.mandl.com/'>

  <appetizer> salad </appetizer>
  <appetizer> calamari </appetizer>
  <entree> pepperoni pizza </entree>
  <dessert> vanilla ice cream </dessert>

</meal>
```


XML NAMESPACE

```
<d:meal xmlns:d='http://www.mandl.com/'>

  <d:appetizer> salad </d:appetizer>
  <d:appetizer> calamari </d:appetizer>
  <d:entree> pepperoni pizza </d:entree>
  <d:dessert> vanilla ice cream </d:dessert>

</d:meal>
```

(same meaning as prev slide)

XML NAMESPACE IDENTIFIERS

XML namespace identifiers must conform to a specific syntax—the syntax for Uniform Resource Identifier (URI) references.

A URI is defined as a compact string of characters for identifying an abstract or physical resource. In most situations, URI references are used to identify physical resources (Web pages, files to download, etc), but in the case of XML namespaces, URI references identify abstract resources, specifically, namespaces.

XML NAMESPACE

```
<mandl>
   <m:menu xmlns:m='http://www.mandl.com/menu'>
     <m:name> Daily Special </m:name>
     <m:cost> 12.50 </m:cost>
  </m:menu>
  <serv:Server xmlns:serv =
   "http://www.mandl.com/servers">
  <serv:name>Smith, John</serv:name>
  <serv:address>11 North Rd</serv:address>
  </serv:Server>
</mandl>
```


XML NAMESPACE IDENTIFIERS

Two general types of URIs: URL and URN

URL: http://www.pizza.com/menu

URN: urn:www-pizza-com:menu

Most important aspect is that they must be unique or the namespace will be confused

XSL LANGUAGES

XSL = Extensible Stylesheet Language

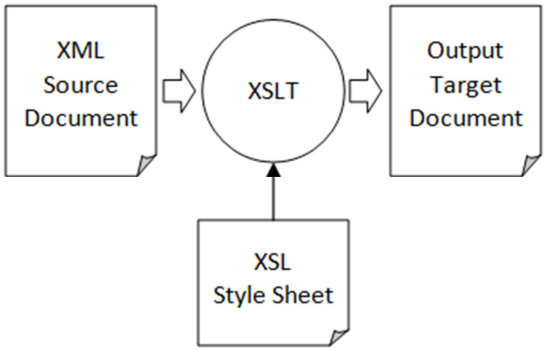
Need for an XML-based stylesheet

XSL is to XML what CSS is to HTML

But XSL is more ... consists of three parts:

- XSLT – transform XML documents
- XPath – navigating in XML documents
- XSL-FO – formatting XML documents

XSL – STYLE SHEET



XSLT TRANSFORMATION

Most important part of XSL is XSLT

Transforming XML into XHTML

```
<xsl:stylesheet  version="1.0"

  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  ... xsl style ...
  <xsl:stylesheet>
```


CREATE AN XSL STYLE SHEET

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

<xsl:stylesheet version="1.0"  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
     <tr bgcolor="yellow">
       <th>Title</th>
       <th>Artist</th>
     </tr>
     <xsl:for-each select="catalog/cd">
     <tr>
       <td><xsl:value-of select="title"/></td>
       <td><xsl:value-of select="artist"/></td>
     </tr>
     </xsl:for-each>
  </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

Saved in a file named cdcatalog.xsl

SAMPLE XML TO TRANSFORM

```
<?xml version="1.0" encoding="UTF-8"?>
   <catalog>
     <cd>
       <title>Empire Burlesque</title>
       <artist>Bob Dylan</artist>
       <country>USA</country>
       <company>Columbia</company>
       <price>10.90</price>
       <year>1985</year>
     </cd>
</catalog>
```

LINK XSL STYLE SHEET TO XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
  <catalog>
    <cd>
      <title>Empire Burlesque</title>
      <artist>Bob Dylan</artist>
      <country>USA</country>
      <company>Columbia</company>
      <price>10.90</price>
      <year>1985</year>
    </cd>
</catalog>
```


LOAD XML FILE IN BROWSER

My CD Collection

Title	Artist
Empire Burlesque	Bob Dylan

XSL TEMPLATE ELEMENT

The <xsl:template> element builds templates

The match attribute associates a template with an XML element via an XPath expression

`<xsl:template match="/">` associates the template with the root of the XML

XSL VALUE OF ELEMENT

The <xsl:value-of> element extracts the value of an XML element and add it to the output of the transformation (to browser)

The select attribute contains an Xpath expression where the forward slash is used similarly to selecting subfolders

`<xsl:value-of select="catalog/cd/title"/>`

XSL FOR EACH ELEMENT

The <xsl:for-each> element enables 'looping' through each element in the xml

The select attribute is an XPath expression

```
<xsl:for-each select =“catalog/cd”>
  ...
</xsl:for-each>
```

loop for each cd element within the root catalog element

XSL SORT ELEMENT

The <xsl:sort> element sorts output

```
<xsl:for-each select=“catalog/cd”>
     <xsl:sort  select=“artist”/>
     ...
 </xsl:for-each>
```


XSL IF ELEMENT

The <xsl:if> element provides a conditional test of the transformed XML content

The test attribute contains the logical expression to evaluate

```
<xsl:if test=“price &gt; 20”>
   <tr>
   ...
  </tr>
</xsl:if>
```

Display the cd row if this cd's price value is greater than 20.