Extensible Markup Language
XML
What, Purposes, How
Output
DTD, XML Schema
Web Services

The computer-age information world history
• Pre-Internet
• Internet started
• Web started

• Pre-Internet
Independent shops
• Own information
• Own structure
• Own form / format

How to obtain information from within or elsewhere?

• Internet started
– Connected broadly
– Different protocols for accessing information
  • telnet or some other remote login protocol
    Hand copy, paste, type
  • ftp
    Further processing

What types of information one typically gets?

• Web started
Connected, consistent, smooth transmission
– Simple clicking
– More recently, high power search engines

– Intense human intelligence still required
– High level application software/hardware still cannot easily communicate with one another on their own
• Consider parts of two separate HTML pages:

```
<h2>Home Addresses</h2>
<P>Richard Smith<br>
2004 Lakewood Drive<br>
Norfolk, VA 23556</P>

<P> Richard Smith found his friend’s home address in Norfolk from a telephone book:<br>
1125 Hampton Boulevard<br>
Norfolk, VA 23528</P>
```

What about searching for Richard Smith’s address?

• Consider the following possibility, not in HTML:

```
<person>
  <name> Richard Smith </name>
  <address> 2004 Lakewood Drive
             Norfolk, VA 23556 </address>
</person>
```

Something like this would definitely help.

---

**XML for the help**

• XML allows customized tags

http://www.w3.org/XML/
  – Users can define customized semantics.
  – For every pair of applications, in fact for each way in which they need to be linked, someone has to create an "XML to XML bridge."

What about among many different applications?

• XML allows building of hierarchy trees

```
<person>
  <name> Richard Smith </name>
  <address> 2004 Lakewood Drive
             Norfolk, VA 23556 </address>
</person>
```

This example is really a hierarchy tree, and it can be refined and/or built up further.

---

**What is XML?**

• A standard for languages to describe data with customized markup tags
  – Any user can define his/her own tags
  – The tags can be used for arbitrary purposes
  – Allows organizing the structure of data

Not the structure of display.

---

Same data in a more detailed tree structure:

```
<person>
  <name> <first>Richard</first> <last>Smith</last> </name>
  <address> <number>2004</number><street>Lakewood Drive</street><city>Norfolk</city><state>VA</state><zip>23556</zip> </address>
</person>
```
• **XML vs. HTML**
  – We may think of the contents represented in HTML on the Web as the (to be) presented information.
  – We may think of the contents represented in XML on the Web (or elsewhere) as described information (described) with meta-information

• **Meta-information** – the information that describes information.
  • XML allows creation of tags for describing semantics, and forming structures of them. It can describe different parts of the data and how the data is structured.
  • It does not describe how the data is shown.

• **Some basic concepts in XML**
  – **Element**  
    A pair of tags and what is in between
  – **Parent vs. Children elements**  
    One level down inside an element are its children elements. The outside one is the parent element of its children elements.
    
    A child element can have its own children elements.

  Consider the earlier example again:
  
  `<person>
  <name> <first>Richard</first> <last>Smith</last> </name>
  <address> <number>2004</number> <street>Lakewood Drive</street> <city>Norfolk</city> <state>VA</state> <zip>23556</zip> </address>
  </person>`
  
  `<person>` element has children elements `<name>` and `<address>`
  `<name>` element has children elements `<first>` and `<last>`
  
  *Remember the tree structure of it?*

• **The root element**  
  The top level element of a hierarchy tree

• **Parsed Character Data (PCDATA)**  
  • The text content of an element is referred to as Parsed Character Data, PCDATA
  • From the previous example, we have `<first>Richard</first>` in which `Richard` is PCDATA.

• **Character Data (CDATA)**
  For inclusion of scripts and other types of code.

  ```xml
  <![CDATA[
  Function clickme(){
  {Alert("This is a Javascript function")}
  ]]}>
  ```
  
  CDAT is not to be interpreted by XML.
– Tag names must follow naming conventions
  • names are case sensitive
  • names start with an English letter
    not underscore, nor xml
  • later characters can be letter, digit, or underscore, but no space is allowed

  Are the following three legal tag names?
  Social security number
  SocialSecurityNumber
  Social_security_number

– Matching open and close tags
  • Matching closing tag must be present, except
    – Empty elements
      Can use a shorthand notation, e.g.
      <address></address>
      may be represented as
      <address/>

– HTML tags without closing tags
  HTML tags are treated as XML tags. Any
  HTML tag without a closing tag must be
  escaped in the following manner
  <IMG src="…" />
  <BR/>

  • Pairs of tags must not cross each other, but
    different pairs may be nested

  Are they OK?
  <a> <b> </b> </a>
  <a> <b> </a> </b>
  <a> <b> </B> </a>

– Element vs. attribute values
  • An element may have attributes, which would have
    values.
  • Attribute values may be given in either single or
    double quotes:
    <note type="warning">Turn with caution</note>
    Or: <note type='warning'>Turn with caution</note>

  • An attribute may be replaced by a child element
    The previous example could be represented as:
    <note><type>warning</type>
    Turn with caution</note>

– Comments
  Given inside <!-- and -->

– A example XML file
  <?xml version="1.0" encoding="UTF-8"?>
  <people_list>
    <person>
      <name>Fred Bloggs</name>
      <birthdate>27/11/2008</birthdate>
      <gender>Male</gender>
    </person>
  </people_list>
XML Namespaces

- Namespace
  - A namespace is a set of unique names
  - A namespace is also called a context, and in which each name has only one meaning, i.e. unique meaning.
  - For many programming languages, a namespace is a context for identifiers.

- XML namespace
  - An XML namespace is a W3C standard for providing uniquely named elements and attributes in an XML application instance
  - An XML application instance may contain element or attribute names from more than one XML namespace
  - Names in an XML namespace represent real-world things or concepts

- XMLNS
  - xmlns is a reserved XML attribute, it means XML namespace
  - It is used to associate with a namespace used in the document

- An example of using xmlns
  <movie
    xmlns:nelson="http://nelson.com/ratings"
    xmlns:xyahoo="http://xyahoo.com/movieratings">
    <title>Great Adventure</title>
    <nelson:rating>A</nelson:rating>
    <xyahoo:rating>G</xyahoo:rating>
  </movie>

  This movie has/uses two types of ratings, each is from a corresponding namespace.
  The two URIs in here happen to be URLs – most are.

- The value of xmlns must be a URI (Uniform Resource Identifier) reference
  - URI is an Internet protocol element consisting of a short string which indicates a name or address that can be used to refer to an abstract or physical resource, e.g., person, product, event, etc.
  - The URI syntax
    - a URI scheme name like "http", "ftp", "mailto", "urn", etc.,
    - a colon character
    - a scheme-specific part
  - URI is a superset of URL
XML Editors

- Many XML editors are now available
  - Find them via a search engine
  - Use MS Word 2003 as a XML editor

Displaying XML Data

- Using Cascading Style Sheets (CSS)
  http://www.w3.org/Style/CSS/
  - CSS can be applied to HTML and XML as well

- Using Extensible Stylesheet Language (XSL)
  http://www.w3.org/Style/XSL/
  - Specifically designed for use with XML
  - Can rearrange and reorder elements
  - Can hide some elements and display others
  - Can be based on various factors
    - the tag
    - the contents and attributes of the tag
    - the position of the tag

Examples of outputting XML documents

- Viewing in the raw form
  <?xml version='1.0' ?>
  <document>
    <body>
      My name is:<name>John Smith</name>
      My address is:<address>123 Lakeland Street, Norfolk, VA</address>
      My phone number is:<phone>757 444-5555</phone>
    </body>
  </document>
  http://www.cs.odu.edu/~shen/cs312/xml/example0.xml

- Displaying with the CSS file: example1Style.css
  <?xml-stylesheet type="text/css" href="example1Style.css" ?>
  <document>
    <body>
      My name is: <name>John Smith</name>
      My address is: <address>123 Lakeland Street, Norfolk, VA</address>
      My phone number is: <phone>757 444-5555</phone>
    </body>
  </document>
  http://www.cs.odu.edu/~shen/cs312/xml/example1.xml

The CSS file, example1Style.css:

name {background-color:green}
address {background-color:blue}
phone {background-color:red}
• Displaying using CSS based on attribute value and for different media devices

```xml
<?xml-stylesheet type="text/css" href="example3Style.css" ?>
<document>
<body>
  My name is: <student type="name">John Smith</student>
  My address is: <student type="address">123 Lakeland Street, Norfolk, VA</student>
  My phone number is: <student type="phone">757 444-5555</student>
</body>
</document>
```

The CSS file: example3Style.css
```
@media screen, print {
  student[type="name"] {background-color:green;}
  student[type="address"] {background-color:blue;}
  student[type="phone"] {background-color:red;}
}
@media aural {
  student[type="name"] {volume:medium; voice-family:FriendlyAdvice;}
  student[type="address"] {volume:loud; voice-family:Directions;}
  student[type="phone"] {volume:soft; voice-family:Directions;}
}
```

Here it defines for screen and sound devices. We can not demonstrate it on sound devices. Unfortunately, it does not work on IE yet.

• Displaying xml with xsl

We have a set of film descriptions in the XML file: films.xml (displaying with: films.xsl)

```xml
<?xml version = "1.0" ?>
<?xml-stylesheet type="text/xsl" href="films.xsl"?>
<FILMS>
  <FILM FILM_GENRE="Comedy">
    <NAME>When Hunky met Silly</NAME>
    <FILM_RATING>4</FILM_RATING>
    <AMBIANCE_RATING>2</AMBIANCE_RATING>
    <AVERAGE_ENTRY_COST>$6</AVERAGE_ENTRY_COST>
    <REVIEW>Great film, not very intellectual.</REVIEW>
  </FILM>
  <FILM FILM_GENRE="Western">
    <NAME>The Good, the Bad and the Ugly</NAME>
    <FILM_RATING>3</FILM_RATING>
    <AMBIANCE_RATING>4</AMBIANCE_RATING>
    <AVERAGE_ENTRY_COST>$8</AVERAGE_ENTRY_COST>
    <REVIEW>Classic western.</REVIEW>
  </FILM>
  <FILM FILM_GENRE="Chinese">
    <NAME>Enter the Dragon</NAME>
    <FILM_RATING>4</FILM_RATING>
    <AMBIANCE_RATING>5</AMBIANCE_RATING>
    <AVERAGE_ENTRY_COST>$9</AVERAGE_ENTRY_COST>
    <REVIEW>Excellent for Kung Fu lovers.</REVIEW>
  </FILM>
</FILMS>
```

The corresponding xsl file, films.xsl:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <BODY>
        <TABLE BORDER="1">
          <TR>
            <TH>Film Title</TH>
            <TH>Film Genre</TH>
            <TH>Film Rating (max 5)</TH>
            <TH>Ambiance (max 5)</TH>
            <TH>Average Entry Cost</TH>
            <TH>Review</TH>
          </TR>
          <xsl:for-each select="FILMS/FILM">
            <TR>
              <TD><xsl:value-of select="NAME"/></TD>
              <TD><xsl:value-of select="@FILM_GENRE"/></TD>
              <TD><xsl:value-of select="FILM_RATING"/></TD>
              <TD><xsl:value-of select="AMBIANCE_RATING"/></TD>
              <TD><xsl:value-of select="AVERAGE_ENTRY_COST"/></TD>
              <TD><xsl:value-of select="REVIEW"/></TD>
            </TR>
          </xsl:for-each>
        </TABLE>
      </BODY>
    </HTML>
  </xsl:template>
</xsl:stylesheet>
```


```xml
<?xml version="1.0" ?>
  <xsl:template match="/">
    <html>
      <head>
        <title>Films</title>
      </head>
      <body>
        <h1>Films</h1>
        <table border="1">
          <tr>
            <th>Film Title</th>
            <th>Film Genre</th>
            <th>Film Rating (max 5)</th>
            <th>Ambiance (max 5)</th>
            <th>Average Entry Cost</th>
            <th>Review</th>
          </tr>
          <xsl:for-each select="FILMS/FILM">
            <tr>
              <td><xsl:value-of select="NAME" /></td>
              <td><xsl:value-of select="@FILM_GENRE" /></td>
              <td><xsl:value-of select="FILM_RATING" /></td>
              <td><xsl:value-of select="AMBIANCE_RATING" /></td>
              <td><xsl:value-of select="AVERAGE_ENTRY_COST" /></td>
              <td><xsl:value-of select="REVIEW" /></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
Document Type Definition (DTD)

- **DTD**
  - An optional specification of the syntax of an application of XML
    - Guide application programmers to write, maintain, or use an XML application
    - Processors could use it to verify the validity of an XML application

- Physical placement of DTD
  - May be included within an XML document file
  - Normally stored in a separate ASCII-text file and can be used by multiple XML files

**HTML 4.0 DTD W3C recommendation:**
http://www.w3.org/TR/REC-html40/sgml/dtd.html

**Tutorial:**
http://www.w3schools.com/dtd/

- Example DTD statements
  ```xml
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT name (#PCDATA) >

  A person element contains the elements name, birthdate, gender and socialsecuritynumber and is called a container element.
  - The "?" indicates that an element is 0 or 1 occurrence.
  - Element name above can only have pcdta.
  ```

- Current status of DTD
  - Its implementation is not broad.
  - It is less powerful than the newer XML Schema.
  - The modern browsers Mozilla and IE do implement them, but only verifies syntactic correctness of DTDs and do not check the conformity of a XML document against its DTD.

- The DTDexample.dtd file:
  ```xml
  <!ELEMENT people_list (person*)>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT socialsecuritynumber (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT gender (#PCDATA)>
  ```

- The following XML file refers to the previous DTD file:
  ```xml
  <?xml version="1.0" encoding="UTF-8"?>
  <!DOCTYPE people_list SYSTEM "DTDexample.dtd">
  <people_list>
    <person>
      <name>Fred Bloggs</name>
      <birthdate>27/11/2008</birthdate>
      <gender>Male</gender>
      <extra>just to illustrate</extra>
    </person>
  </people_list>
  ```
<person>
  <name>John Smith</name>
  <socialsecuritynumber>111223333</socialsecuritynumber>
  <gender>Male</gender>
</person>

<person>
  <name>Mary Jones</name>
  <socialsecuritynumber>333223333</socialsecuritynumber>
  <gender>Female</gender>
  <degree>BS</degree>
</person>

<people_list>

http://www.cs.odu.edu/~shen/intl
IDB/SampleCodings/xml/DTDex
ampleXMLfile.xml

---

XML Schema

- XML Schema
  http://www.w3.org/XML/Schema
  - Similar to DTD but newer and more powerful.
  - Definition in XML syntax
  - May be referred to as W3C XML Schema, since it is the first to be published as a W3C Recommendation, and there are others coming out after it, e.g. RELAX NG.

- It is also often referenced with the acronym WXS.
- An XML Schema instance may be referenced as an XSD (XML Schema Definition), and typically has the filename extension ".xsd".
- Their sponsors, Microsoft and Commerce One, respectively, have both announced that they would support XML Schema for their new developments.

- The XML-Schema language allows groups with common interests to express a schema on line

A company, or even a particular business sector, can develop a standard set of XML mappings (i.e. a particular XML schema), and therefore be able to represent their content using a common structure.

XML Applications

- We can create markup languages in a variety of disciplines
- Called XML applications
- Following rules of XML, each application has its own syntax and vocabulary
- To manage information in specific disciplines
  - To describe complex objects
  - To send complex data over the web

- Examples of XML applications
  - Chemical Markup Language (CML)
    http://www.xml-cml.org/
    - For many aspects of chemical informatics
    - For data-handling and publication
    - To allow definitions of many chemical structures.
• Mathematical Markup Language (MathML)
  http://www.w3.org/1999/07/REC-MathML-19990707/
  – Describing mathematical notation
  – Capturing both its structure and content.
  – Enabling mathematics to be served, received, and processed on the Web

• Open Software Description (OSD)
  http://www.w3.org/TR/NOTE-OSD
  – Describing software packages
  – Describing their dependencies for heterogeneous clients
  – For automated software distribution and maintenance

Web Services
http://www.w3.org/2002/ws/
• The programmatic interfaces for application to application communication on the Web are referred to as Web services.
• Communication using the Simple Object Access Protocol, SOAP, over HTTP.
  • Based on XML data.
  • A business application sends a request to a service at a given URL. The service receives the request, processes it, and returns a response.
  • Highly automated.

Review: Working with XML
• Designing tags
  – Custom designed tags
  – Polling multiple XML vocabularies
  Using Namespaces to distinguish a common name
• XML editors
• Definition of a XML
  – Using Document Type Definitions (DTDs)
  – Using Schemas (definition in XML syntax)
  • Interfacing with other applications
    – Using a Document Object Model (DOM)
• Querying parts of a XML document
  – Using XPath
• Displaying data
  – Using Cascading Style Sheets (CSS)
  – Using Extensible Stylesheet Language (XSL)
• Applications of XML
  – Various applications
    • Chemical Markup Language (CML)
    • Mathematical Markup Language (MathML)
    • Open Software Description (OSD)
  – Web Services
    • Request
    • Response