CSE 530A

XML

Washington University
Fall 2013
XML

• Extensible Markup Language
  – Rules for encoding documents
  – Both human readable and machine readable
  – Based on SGML (Standard Generalized Markup Language)
    • "Simpler" than SGML
  – Used as both a document storage format and a data interchange format
XML

- Used as both a document storage format and a data interchange format
  - RSS, Atom, SOAP, AJAX, XMPP
  - MS Office, Open Office, iWork
XML

• Like HTML, XML uses tags in angle brackets $<>$
  – Similarity ends there

• HTML
  – Fixed set of tags
  – Tags describe document structure but not content
    • No semantic information

• XML
  – Tags are defined by the creator of a particular document type
  – Tags should have semantic information
Example

• HTML
  – Describes how data should be formatted, but not what the data is

    <p>
    <b>John Smith</b><br>
    123 Main Street<br>
    St. Louis, MO 63130
    </p>

• XML
  – Describes what the data is (but not how it should be formatted)

    <address>
    <name>John Smith</name>
    <street>123 Main Street</street>
    <city>St. Louis</city>
    <state>MO</state>
    <zip>63130</zip>
    </address>
Example

• HTML
  – Cannot programmatically tell that the example is an address

• XML
  – The tags indicate that this is an address
    • Even indicates the constituent parts
Basic Parts

• Tags
  – Designate the start and end of elements
  – Should be semantically meaningful

• Elements
  – Logical document components

• Attributes
  – Additional data (often metadata) associated with tags
Basic Parts

• Tags
  – Consists of text in angle brackets
    • <name>
  – Must occur in matching pairs
    • <name> … </name>
  – Shorthand for tag pair with no content
    • <name />
• Tag names
  • Cannot contain some characters:
    !"#$%&'()*+,-./;<=?@[\]^`{|}~
  • Cannot contain a space
  • Cannot start with -, ., or a numeric digit
Basic Parts

• Elements
  – Consists of a matching tag pair and everything in between
    • <name>John Smith</name>
Basic Parts

• Attributes
  – Consists of a name=value pair inside a starting tag
    • Attribute values are placed in quotation marks
  – <recipe type="dessert"> 
    • type="dessert" is an attribute
Declaration

• XML documents can start with an optional declaration
  – `<?xml version="1.0" encoding="UTF-8"?>`
  – Version and encoding are optional
  – Current versions include 1.0 and 1.1, but 1.1 is rarely used

• Can also contain a DOCTYPE declaration
  – More on this later
Characters

• Strong support for Unicode
• The NULL character \0 is not supported
• < and & can never appear in content (outside of a CDATA section) without being escaped
  – > should generally be escaped also
Character

• Five predefined entities
  – \&lt; \ represents <
  – \&gt; \ represents >
  – \&amp; \ represents &
  – \&apos; \ represents '
  – \&quot; \ represents "

• Any other legal character can be designated using its Unicode numeric code
  – # for decimal, \#x for hexadecimal
  – \&160#; and \&#xa0; both represent a non-breaking space
CDATA

• CDATA provides an alternate syntax for including data without having to escape the special characters
  • `<code>0 &lt; 1</code>`
    – Can also be written
      • `<code>&lt;![CDATA[0 &lt; 1]]&gt;</code>`

• Useful for embedding strings without having to sanitize them

• The data in the CDATA section cannot contain the string `]]>`
  – Means that CDATA sections cannot be nested
Comments

- Comments in XML
  - Start with <!--
  - End with -->
    - <!-- This is a comment -->
- Comments cannot contain --
  - Therefore no nested comments
- Comments can span multiple lines
- Must not break the structure of the document
  - Cannot comment out one tag of a pair and not the other tag
Structure

• Every start tag must have a matching end tag
• Elements must be properly nested

```html
<address>
  <name>
    ...
  </name>
  <street>
    ...
  </street>
  ...
</address>
```

• HTML parsers generally allow sloppiness in tag matching
• XML parsers are strict
Structure

- The proper nesting of the elements means a document can be viewed as a tree
  - The outermost element is the root, nested elements are children

```xml
<address>
  <name>
    ...
  </name>
  <street>
    ...
  </street>
  ...
</address>
```
Well-formed XML

• An XML document is considered *well formed* if it adheres to the proper syntax
  – Single root element
  – Properly nested elements
• This does not mean that the document is *valid*
  – Uses the user-defined tags in the proper way
Valid XML

• An XML document is valid if
  – It is well formed
  – It conforms to the rules of a Document Type Definition (DTD) or XML Schema Definition (XSD)
Parsing vs Validating

• An XML parser does not determine if a document is *valid* but merely *well formed*
  – It determines if a document adheres to the proper XML syntax
  – It does *not* determine if a document conforms to a DTD or XSD
  – This means an XML parser can parse *any* well-formed document

• An XML validator will determine if a (well-formed) document conforms to a DTD or XSD
DTD and XSD

• The tags and nesting structure of XML documents are user defined

• A DTD or XSD defines which elements and references may appear in a document
  – Allows for validation of a document beyond just syntax
  – Especially useful for data interchange
    • A web service (or other service) can
      – Publish a DTD for clients to use
      – Validate that received documents conform to the DTD
DTD

• Document Type Definition
  – Old way of defining document types
    • Sometimes considered out of date, but still in use
• Designated in the XML document using a DOCTYPE declaration
  – Generally three types
    • PUBLIC: indicates a well-known DTD
    • SYSTEM: URL to a file containing the DTD
    • Embedded: DTD is embedded directly in the document
• Root element must be `<html>`
• References the public HTML 4.01 Transitional doctype
DOCTYPE

<!DOCTYPE addresses SYSTEM "http://www.myserver.com/addresses.dtd">

- Root element must be addresses
- References a DTD at the specified URL
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

- PUBLIC doctypes also usually include a SYSTEM identifier
Embedded doctypes include the DTD directly in the XML
DTD Limitations

• Do not have support for newer features of XML
  – E.g., namespaces

• Are not themselves XML
  – Based on a regular expression syntax
    • Cannot express non-regular structures

• Only support rudimentary datatypes
XSD

• Multiple successors to DTDs have been defined
• XML Schema Definition is the most popular
  – Official W3C successor to DTD
XSD Example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema version="1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="addresses">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="address" maxOccurs="unbounded" type="addressType"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="addressType">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="street" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xs:schema>
```

• Note that the schema itself is an XML document
XML Example

<?xml version="1.0" encoding="utf-8"?>
<addresses xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
    xsi:noNamespaceSchemaLocation="addresses.xsd">
    <address>
        <name>John Smith</name>
        <street>123 Main Street</street>
    </address>
</addresses>
In addition to document validation, XSD can be used to generate code for XML data binding

- E.g., the various elements in the schema can be mapped to Java classes
  - An Addresses class which contains a list of Address objects
  - An Address object which contains a name field and a street field
  - Automatic conversion back and forth from an XML document to an object collection can be done
XML Processing

• Various ways of processing XML
  – Document Object Model
  – Simple API for XML
  – XML data binding
  – XSLT
  – XPath
  – XQuery
• Document Object Model
  – Represents the entire XML document as a tree data structure
  – Can process the document by traversing the tree
  – Memory intensive for large documents
SAX

• Simple API for XML
  – Stream-oriented callback model
  – Single pass over document, callbacks called when elements are hit
  – Less memory intensive than DOM, can be faster
  – Requires more work from the user to keep track of document structure
XSLT

• Extensible Stylesheet Language Transformations
  – A language for transforming XML documents into other XML documents or other objects (e.g., plain text, PDF)
XPath and XQuery

- XML Path Language
  - A query language for selecting nodes from an XML document

- XQuery
  - A query and functional programming language
  - Extract and manipulate data from XML documents
    - Or any data source that can be viewed as XML
XML Databases

• XML-enabled databases
  – Traditional databases with XML support
    • Provides functions for storing and retrieving XML
    • Cannot generally search on XML data types
      – Text search support can sometimes be used

• Native XML databases
  – Use XML documents as the fundamental unit of storage
  – A type of *document-oriented* database
    • Designed around the idea of a document rather than relations