

1 XML (Introduction)

Tim Margush

Based on Programming the WWW by Robert Sebesta, 4th Edition

2 What is XML

- eXtensible Markup Language
- Uses tags to describe structure and content
 - XML allows user-defined tags
 - Its purpose is to define tags
- Universal data exchange language
 - Platform independent
- Uses?
 - Web documents
 - Documentation
 - Databases

3 XML - HTML

- Different goals
 - XML – a language for defining markup languages
 - HTML – a markup language for generalized layout of information
- XHTML is HTML reformulated as an XML Application
 - An XML Application is a markup language designed with XML

4 XML Processor

- Any program that inputs and outputs XML documents
 - These can be highly generalized or very specific
- Generating
- Parsing
- Restructuring
- Reporting

5 Basic Syntax

- All XML documents must obey these basic syntax rules:
 - `<?xml version='1.0'?>`
 - `<rootelement>`
 - content
 - `</rootelement>`
- In addition, an application specifies other details
 - DTD
 - Schema

6 XML Details

- `<!-- comment -->`
- `<tagname>`
 - Case-sensitive
 - Start with letter or `_`
 - May also contain digits, hyphens, and dots
- Tags must have a matching end tag
 - `</tagname>`
 - or they must use the shorthand `<tagname/>`

7 XML Details

- Tags must be properly nested
- Attribute values must be quoted
 - attributename = 'single'
 - and attributename = "double" are equivalent
- Attribute names have the same characteristics as tag names
- Have a single root element

- Well-Formedness

8 Sample

```
<?xml version='1.0'>
<guests>
<address number='234' street="First St."
  city='Akron' state='OH' zip='44325' />
<address>
  <street-address>
    <number>235</number>
    <street>First Street</street>
  </street-address>
  <city>Akron</city><state>OH</state>
  <zip>44325</zip>
</guests>
```

9 Auxiliary Documents

- Syntax rules
 - DTD
 - XML Schema
- Stylesheet
 - CSS

10 Entities

- Logical parts of a document
 - The document itself is a single entity
- Entities may contain or reference other entities
 - Entities allow multiple documents to reference standard entities for consistency
 - Repeated references within a document can be standardized using an entity
 - Binary entities cannot be directly held in an XML document

11 Examples

- Example of entities
 - A header or footer
 - A company's legal name and address
 - An image
 - A character that has another meaning in the markup language
- Text entities are replaced by the text they represent
- Binary entities must be handled separately

12 Entity References

- The entity name is used to refer to a named entity
 - &entity-name;

- There are many predefined entities
 - < © & ...
- Entities are defined in DTD's

13 Character Data Section

- Used when some text contains many special characters
- <![CDATA[content is <not> parsed]]>
- <! is the start of a declaration
 - Declarations usually appear in DTD's, but the above example might be considered an inline declaration

14 Document Type Definitions

- Can be embedded in an XML document or can be stored in a separate file
- Impose syntax rules on an XML document
- Provide a place to define entities
- General structure
 - <!DOCTYPE root_element_name [
 - <! A set of declarations >
 - <! something meaningful... >
 -]>

15 Declaration Keywords

- ELEMENT
 - Introduce a legal tag name
- ATTLIST
 - list the attributes that are legal in a specific context
- ENTITY
 - Define an entity
- NOTATION
 - Define data type notations


16 Element Declarations

- <!ELEMENT elt_name (#PCDATA) >
(this would be a leaf node)
- <!ELEMENT elt_name (list, of, elements) >
(this would be an internal node, order matters)
- <!ELEMENT elt_name (childa | childb)+ >
(this would be an internal node, at least one child, order does not matter)

17 Element Examples

- <!ELEMENT address (street, city, state, zip)>
- <!ELEMENT street (number?, name, name?)>
- <!ELEMENT number (#PCDATA)>
- <!ELEMENT name (#PCDATA)>

18 Sample DTD

 1

```
<address>
<street>
  <number>347</number>
  <name>East Ave</name>
</street>
```

```

<city>Akron</city>
<state>OH</state>
<zip>44325</zip>
</address>
2 <!ELEMENT address (street, city, state, zip)>
<!ELEMENT street (number?, name, name?)>
<!ELEMENT number (#PCDATA)>
<!ELEMENT name (#PCDATA)>

```

19 Leaf Node Content

- PCDATA - Parseable character data
 - Content will be parsed
 - May contain any characters except <, >, and &
 - Use entities for these characters
- EMPTY
 - No content
- ANY
 - Any content

20 Attribute Lists


- <!ATTLIST elt_name
 - att_name att_type def_value
 - att_name att_type def_value
 - ... >
- <!ATTLIST street
 - name CDATA #REQUIRED
 - direction (oneway|two-way) 'two-way'
 - lanes CDATA '2'
 - shops CDATA #IMPLIED >

21 Attribute Details


- 1 ■ Types
 - CDATA
 - ID
 - ENTITY
 - list of names (enumeration)
 - ...
- 2 ■ Values
 - value
 - Default attribute value if not specified
 - #FIXED value
 - All elements have this whether specified or not
 - #REQUIRED
 - Must include
 - #IMPLIED
 - Optional

22 Entities

- <!ENTITY % entity_name "entity value">
 - Parameter entity – has meaning only within the DTD itself
- <!ENTITY entity_name "entity value">
 - General entity – has meaning in the XML document
- <!ENTITY % entity_name SYSTEM "file location">
 - External text entity - represents the contents of a file
- <!ENTITY sig "Dr. Ex Em Ell">
- <p>sincerely,</p><p>&sig;</p>

23  **Sample DTD**


```
<!ELEMENT quiz (heading, (mcquestion | question)+)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT mcquestion (question, answer*)>
<!ELEMENT question (#PCDATA)>
<!ELEMENT answer (#PCDATA)>
<!ATTLIST mcquestion type CDATA #FIXED 'mc'>
<!ATTLIST question
  type (tf|sa) "tf"
  answer CDATA #IMPLIED>
<!ATTLIST answer correct (yes|no) "no">
<!ENTITY title "XML Quiz with DTD">
```

24  **DTD in an External File**

dtd.dtd (file)


```
<!-- Comments describing the DTD -->
<!ELEMENT rootname ...
...

dtd.xml (file)
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE rootname SYSTEM "dtd.dtd">
<rootname> ...
```


25  **Internal DTD**

dtd.xml (file)

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE rootname [
<!-- Comments describing the DTD -->
<!ELEMENT rootname ...
]>
<rootname> ...
```

26  **Schema**

- XML Application specifying the structure of other XML applications
- Alternative to DTD
 - More powerful – more data types
 - Easy to validate syntax (it is XML)
 - Utilizes namespaces

27  **Using XML Files**

- Display
 - Stylesheets
 - XML Transforms
 - Custom processing

28  **CSS Stylesheet**

```
quiz {color:blue;}
question {display:block; margin-top:3em;}
answer {display:block; margin-left:3ex;}
etc...
```

- display:block
- display:inline (default)

```
<?xml-stylesheet type='text/css' href="loc.css"?>
```

29 XSLT

- eXtensible Stylesheet Language Transformations
 - A report language
 - Transforms XML to XML (or XHTML) according to templates populated from data

```
<?xml-stylesheet type='text/xsl' href="loc.xsl"?>
```

30 Sample XSLT

```
<?xml ...?>
<xsl:stylesheet version='1.0' xmlns:xsl=
  'http://www.w3.org/1999/XSL/Transform'>

<!-- templates -->

</xsl:stylesheet>
```

31 Sample Template

```
<xsl:template match='mcquestion'>
<li><xsl:value-of select='question' /><ol>
  <xsl:for-each select='answer'>
    <li><xsl:value-of select='.' /></li>
  </xsl:for-each>
</ol></li>
</xsl:template>
```

32 XML Processors

- Simple API for XML (SAX)
 - Document is scanned from top to bottom
 - Recognitions of selected structures trigger an event
 - Event handler processes the data
- DOM
 - XML converted to a tree data structure
 - Methods allow traversals and restructuring