XML — Architecture, Tools & Techniques

Extensible Markup Language
XML and DTD
Alphabet Soup

- **DTD**, defined with XML; describe data structure.
- **XSD**, Hollander et al. 1999, W3C Recommendation 2001; describe data structure and constraints.

http://www.w3.org/MarkUp/SGML/
http://www.w3.org/MarkUp/
http://www.w3.org/XML/
http://www.w3.org/XML/Schema
Design Objectives

1. XML shall be straightforwardly usable over the Internet.
2. XML shall support a wide variety of applications.
3. XML shall be compatible with SGML.
4. It shall be easy to write programs which process XML documents.
5. The number of optional features in XML is to be kept to the absolute minimum, ideally zero.
6. XML documents should be human-legible and reasonably clear.
7. The XML design should be prepared quickly.
8. The design of XML shall be formal and concise.
9. XML documents shall be easy to create.
10. Terseness is of minimal importance.
More Alphabet Soup

- **DSSSL**, Bosak et al. 1995; manipulate SGML.
- **Jade**, Clark; implements DSSSL.
- **XSLT**, Clark et al., W3C Recommendation 1999, 2007; transform XML data.
- **XSL-FO**, Adler et al., W3C Recommendation 2001; target for high-quality printing.
- **XQuery**, Boag et al., W3C Candidate 2007; query (and transform) XML data.

http://www.jclark.com/dsssl/
http://www.jclark.com/jade/
http://www.w3.org/Style/XSL/
http://www.w3.org/XML/Query/
Well-Formed

- Optional header, one element.

- Elements may be (fully) nested and may contain text.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- Hello, World -->
<hello>Hello, World!</hello>
```

- Element may be empty:

```xml
<empty/>
```

- Start (or only) tag may contain attributes:

```xml
<greeting text="hello" terminator='!'/>
```
Lexical Aspects

- Tag and attribute names: case-sensitive, letters and digits and some other characters, e.g. colon.

- (Some) whitespace is significant as part of text or an attribute value.

- Comment “elements”:

```
<!-- no embedded double-minus -->
```

- Entities for text replacement (code points in Unicode):

```
&amp;amp; &apos; &gt; &lt; &quot; &gt; #123; &gt;#xabcd;
```

- Attribute values cannot contain `< >` delimiting quotes and `&`. 
Checking

- **XML Applet**

- **rxp**

```
$ rxp -o b hello.xml
At 39: comment:  Hello, World
At 61: start: hello
At 68: pcdata: Hello, World!
At 81: end: hello
At 119: EOF
```
JAXP

SAXParserFactory factory =
   SAXParserFactory.newInstance();
factory.setValidating(false); // or true
factory.setNamespaceAware(false); // or true
SAXParser parser = factory.newSAXParser();
parser.parse(
   new InputSource(new StringReader("<?xml ...")),
   new DefaultHandler() { Locator locator;
      public void setDocumentLocator (Locator locator) {
         this.locator = locator;
      }
      public void fatalError (SAXParseException se) {
         System.err.println(locator.getLineNumber()+
            ": fatal: "+se.getMessage());
      }
   });

- Observer pattern: fatalError, error, and warning.

Well-formed and conforming to a DTD:

```xml
<!DOCTYPE family [
  <!ELEMENT family (parent, parent, child*)>
  <!ATTLIST family home CDATA #FIXED 'renting'>
  <!ELEMENT parent (#PCDATA)>
  <!ELEMENT child (#PCDATA)>
]>
<family>
  <parent>Jack</parent>
  <parent>Jill</parent>
  <child>Jo</child>
</family>
```

The DTD can be external (and even combined):

```xml
<!DOCTYPE family SYSTEM 'family.dtd'>
```
ELEMENT

<!ELEMENT a EMPTY>
empty element

<!ELEMENT b ANY>
arbitrary well-formed content

<!ELEMENT c (#PCDATA)>
text content

<!ELEMENT d (#PCDATA)>
text content

<!ELEMENT e (#PCDATA | a)>
text mixed with many elements

<!ELEMENT f (a)?>
optional nested element

<!ELEMENT g (a)>
one or more nested elements

<!ELEMENT h (a)>
any number of nested elements

<!ELEMENT f (a | b? | c+ | d*)>
choice of elements

<!ELEMENT f (a , b? , c+ , d*)>
nested elements in order
Content models can be nested:

```xml
<!ELEMENT nest (( a, b ) | ( c, d ))+ >
```

Result can be ambiguous:

```xml
<!ELEMENT bits (( zero | one )*, ( zero | one )*) >
```

Mixed-content model cannot specify order of elements between text.

```xml
<!ELEMENT mixed (#PCDATA | a | b | c)* >
```

No duplicate definitions.
**ATTLIST**

<!ATTLIST element

  a CDATA 'default value'
  b CDATA #FIXED 'only value'
  c CDATA #IMPLIED
  d CDATA #REQUIRED
  e (u | v | w) #REQUIRED
  f NMTOKEN
  g ID #REQUIRED
  h IDREF #REQUIRED

>

- Attributes appear (at most) once, in any order.
- First declaration is binding. Only one ID.
ENTITY

- Internal or external, for document or DTD, referenced as &name; or %name; but not everywhere.

```
<!ENTITY name 'value'>
<!ENTITY % name 'value'>
<!ENTITY name SYSTEM 'url'>
<!ENTITY % name SYSTEM 'url'>
```

- Technically an include mechanism, rarely used as such.

- Entities reside in a DTD which a non-validating parser might not consult...
Namespaces

- `<eg xmlns="http://www.cs.rit.edu/~ats/"
      xmlns:n1="http://www.w3.org/
      xmlns:n2="http://www.w3.org/">
  <ok a="A" b="B"/>
  <no n1:a="A" n2:a="B"/> <!-- error -->
  <ok n1:a="A" a="B"/>
</eg>`

- `xmlns` links prefix (or default namespace) to URL.
- Scope is block structured, beginning with element.
- Attribute keys default to no namespace.
- Names are equal based on URL and local name, not prefix.
- A parser need not be namespace-aware.
Parameter entities can only be used in external DTDs.

DTDs and namespaces are not meant for each other...
Gory Details

- http://www.w3.org/TR/REC-xml

- http://www.w3.org/TR/REC-xml-names

- EBNF grammar attempts to combine lexical and syntactic aspects.

- Extensive verbal restrictions.