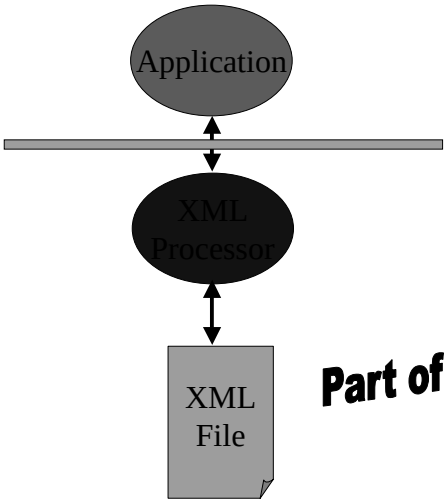
XML

- Definitive specification at
  - http://www.w3.org/TR/REC-xml
  - http://www.xml.com/axml/axml.html (annotated)
- Extensible Markup Language
  - a subset of SGML
  - enable generic SGML to be served, received and processed on the Web as for HTML
  - Designed for
    - ease of implementation
    - interoperability with SGML and HTML

Uses

- A new basis for defining HTML = XHTML
- Storing and transmitting structured data
  - for coupling application file formats: input and output
  - for the content of messages
  - See also SOAP for OO RPC
- As a basis for standardizing data in the semantic realm
  - e.g., how to describe a financial instrument
- Allows for arbitrary XML to XML translation
  - translating data amongst interfaces
  - translating data into XHTML for display
- Example
  - Raw XML: resume.xml
  - Processed using XSL: resume.xml

XML Reference Architecture



SAX
(Simple API for XML Parsing)
- DOM Interface
  (Document Object Model)
- Raw Interface

Part of the power of XML

XML Syntactic Elements

- comment tag: <!-- …-->
- start tag: <tagname attribute="value" …>
- end tag: </tagname>
- empty element tag: <tagname attribute="value" … />
- entity reference: &entity-ref;
- processing instruction: <? … ?>
- doctype declaration: <! .. >

resume.xml

Tags Must Nest

- Tags must nest
  - Illegal:

```
<test>
    <a>
      <b><c>words</b></c>
    </a>
</test>
```


Document Type Declaration

- The XML DTD contains or points to markup declarations that provide a rudimentary grammar for a class of documents.
  - element type declaration
    - declares tags
  - attribute-list declaration
    - declares attributes that go with tags
  - entity declaration
    - declares a reference to something
      - especially replacement text

<!DOCTYPE

```
<!DOCTYPE test SYSTEM "test.dtd">
```
or
```
<!DOCTYPE test [
      markupdecl*
]>
```
or
```
<!DOCTYPE test SYSTEM "test.dtd" [
      markupdecl*
]>
```

- The name given must be the same as the root element type of the XML document.
- resumedtd.xml

Element Declarations

<!ELEMENT *tag-name content*>

- Declares a new element tag.
  - All tag names should be declared (but not an error)

content: EMPTY | ANY | *mixed* | *children*

- e.g.,
  - <!ELEMENT a EMPTY>
    - illegal
      - <a>hello
      - <a>
      - <a>
    - legal
      - <a>
      - <a/>

Children

children: [choice|sequence] [?|*|+]?

choice: (cp ['|' cp]+)

sequence: (cp [',' cp]*)

cp: [name|choice|sequence] [?|*|+]?

- e.g.,
  - <!ELEMENT a (b,(c|d)+)+>
    - illegal
      - <a><b/><b/>
      - <a>
    - legal
      - <a> <b/><c/><d/> <b/><d/>

Mixed

mixed: (#PCDATA ['|' name]*)*

|

(#PCDATA)

- e.g.,
  - <!ELEMENT a (#PCDATA)>
    - illegal
      - <a><b/>
    - legal
      - <a>hello world
      - <a/>
      - <a>
  - <!ELEMENT a (#PCDATA|a)>
    - legal
      - <a> text <a>more

Attribute-List Declarations

- Used to associate name-value pairs with elements
- Attribute specifications may appear only within start-tags and empty-element tags

<!ATTLIST tag-name att-def*>
att-def: att-name att-type att-default
att-default: #REQUIRED | #IMPLIED | [#FIXED] "value"
att-type: CDATA | builtin | (token ['|' token]*)
builtin: ID | ENTITY | NMTOKEN | …

Attribute Example

```
<!ATTLIST a
        id      ID          #REQUIRED
        name    CDATA       #IMPLIED
        type    (A|B|C)     "C"
        kind    (ka|kb)     #FIXED "ka">
```

- legal
  -
  -
- illegal
  - <a/>
  -
  -

Entity Declaration

<!ENTITY [%] name entity-value>

- <!ENTITY threeOverFour "¾">
- <!ENTITY % versionnumber "4.3.2.1">
- <!ENTITY version "Version %versionnumber;">

<a>

&version; and &threeOverFour;

entitydtd.xml using entitydtd.dtd

External Entities

- Parsed external entities

```
<!ENTITY toinclude SYSTEM "file.xml">
…
&toinclude;
```

- Unparsed external entities

```
<!NOTATION jpeg SYSTEM "Joint Photographic Experts
  Group">
<!ENTITY dogpic SYSTEM "dog.jpeg" NDATA jpeg>
<!ELEMENT dog EMPTY>
<!ATTLIST dog picture ENTITY #REQUIRED>
…
<dog picture="dogpic"/>     <!--can ONLY appear here-->
&dogpic;                    <!--forbidden-->
```

Entity Replacement Text

- Literal entity value
  - the quoted string actually present in the entity declaration.
- Replacement text
  - the literal entity value after replacement of character references and parameter-entity values (but not general entity references)
    - allows for mix and match of DTDs by document constructor

Pre-defined Entitites

```
<!ENTITY lt    "&#38;#60;">
<!ENTITY gt    "&#62;">
<!ENTITY amp   "&#38;#38;">
<!ENTITY apos  "&#39;">
<!ENTITY quot  "&#34;">
```

- when the XML processor parses the *entity declaration* for `lt` it stores `&#60;` as its value.
- when `<` is used in the document, it therefore expands to `&#60;` (the five bytes) and not to < (the one byte) which would start a start tag.
- not necessary for the > and quotes

- All XML processors must recognize these entities whether they are declared or not.
  - If they are declared, they must be declared as above.
  - All Unicode numeric entities are recognized by default

'<?' Processing Instructions

- In XML, tags starting with <? are knows as "PI's": processing instructions.
- These are passed through without handling.
- <?xml …> are special, reserved processing instructions.
- The first thing in any XML document is the XML declaration which states the XML standard being used.

```
<?xml version="1.0"?>
```


<?xml PI

- Can also specify
  - standalone nature

```
<?xml standalone="yes"?>
```

    - Indicates there is no external DTD which affects the information passed from the XML processor to the application
  - character encoding used

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

Stylesheet PI

- An important PI is the one that associates a *stylesheet* with an XML document

```
<?xml-stylesheet type="text/xsl" href="resume.xsl"?>
<?xml-stylesheet type="text/css" href="resume.css"?>
```

- A Web browser is an example of an application that reads XML and processes it for display.
  - The above PI tells the browser what type of stylesheet to apply and where to find it.
  - CSS = cascading style sheets
    - declarative
    - straightforward formatting
  - XSL
    - more powerful - generic transformations
    - especially into HTML

Examples of Stylesheets

- This version of resume.xml uses
  - resume.xsl
- This version of resume.xml uses
  - resume.css

CDATA Section

- Used to escape blocks of text containing characters which would otherwise be recognized as markup.

```
<![CDATA[<greeting>hello, world!</greeting>]]>
```

  - tags recognized as character data, not markup

- Especially useful for inserting whitespace which would otherwise be gobbled up by an XML parser:

```
<![CDATA[ ]]>
```


Pre-defined attributes

```
<!ATTLIST poem
    xml:space (default|preserve) "preserve">
    xml:lang NMTOKEN #IMPLIED "fr">
```

- The specification gives special meaning to the attributes xml:space and xml:lang
  - must be declared before being used, however
  - xml:space indicates if XML processor should preserve white space in tags or process it normally (Usually means collapsing and/or re-formatting)
  - xml:lang is a standardized hint to the application as to the language of the content
    - 2-letter language codes ISO639
    - 2-letter country codes ISO3166
    - language identifiers registeres with the Internet Assigned Numbers Authority (IANA-LANGCODES)

XML Namespaces

- Required to avoid name collisions when many are defining their own XML applications.
- An XML namespace is a collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names.

Declaring Namespaces

- A namespace is **declared** using a family of reserved attributes.
- Such an attribute's name must either be xmlns or have xmlns: as a prefix.
  - 'xmlns' sets a default namespace
  - 'xmlns:' is used to declare a namespace which will be used in the document

Qualified Names

- Qualified Names:
  - namespace prefix : local part
  - the namespace prefix maps to a namespace URI.

```
<html:html xmlns:html='http://www.w3.org/TR/REC-html40'
  <html:head>
    <html:title>title</html:title>
  </html:head>
  …
</html:html>
```

- Available in that element and all nested elements.

Default Namespaces

- A default can be set for a tag and all its nested tags:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>title</title>
  </head>
…
</html>
```

- The default applies to the element in which it is delcared, and all nested elements
  - provided they have no namespace prefix

Meaning of Namespace URI's

- No meaning: an arbitrary set of characters following certain formatting rules.
- Should be used to point to a document that describes (any way you want) the tags available in that namespace
  - not required