Moving on…

- UML Classes => XML documents
- UML Use Cases => Xquery/Xpath queries

References

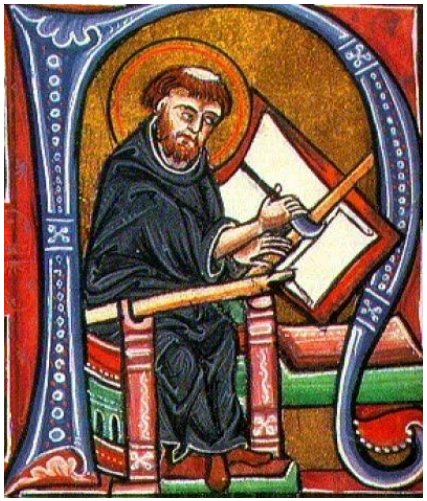
- XML
  - Home Page: http://www.w3.org/XML/
  - Tutorial: http://www.w3schools.com/xml/default.asp

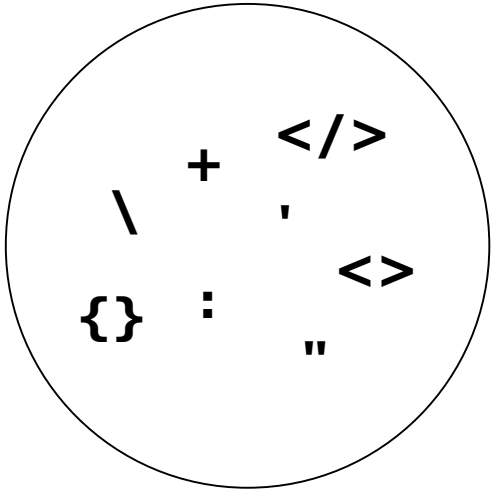
- XML Processing
  - Tutorial: http://www.w3schools.com/XML/dom_intro.asp
  - https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html
  - Home Pages: http://sax.sourceforge.net/ http://www.w3.org/DOM/

<LectureTopic>
Fundamentals of e**X**tensible**M**arkup**L**anguage
</LectureTopic>

What is Markup

- Sequence of characters within a text or word processing file to define
  - Print properties
  - Display properties
  - Document's logical structure
- Markup indicators are often called "tags"
  - Examples
    - RTF
    - EDIFACT
    - XML





Mark Up: RTF

```
\li0\ri0\sb240\sa60\keepn\widctlpar \aspalpha\aspnum\faauto\outlinelevel2 \a
djustright \rin0\lin0\itap0
\b\f1\fs26\lang2057\langfe1033\cgrid\langnp2057\langfenp1033
{\lang6153\langfe1033\langnp6153 Entity Relationship Diagram
\par }\pard\plain \s1\ql
\li0\ri0\sb240\sa60\keepn\widctlpar \aspalpha\aspnum\faauto\outlinelevel0 \a
djustright \rin0\lin0\itap0 \cbpat17
\b\f1\fs24\lang2057\langfe1033\kerning32\cgrid\langnp2057\langfenp1033
{\lang6153\langfe1033\langnp6153 Entity Type
\par }\pard\plain \ql
\li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0
\fs24\lang2057\langfe1033\cgrid\langnp2057\langfenp1033
{\b\fs20\ul\lang6153\langfe1033\langnp6153
Def.:}{\b\fs20\lang6153\langfe1033\langnp6153 }{
\fs20\lang6153\langfe1033\langnp6153 An object or co ncept that is
identified by the enterprise as having an independent existence.
\par }\pard\plain \s1\ql
\li0\ri0\sb240\sa60\keepn\widctlpar \aspalpha\aspnum\faauto\outlinelevel0 \a
djustright \rin0\lin0\itap0 \cbpat17
\b\f1\fs24\lang2057\langfe1033\kerning32\cgrid\langnp2057\langfenp1033
{\lang6153\langfe1033\langnp6153 Entity
\par }\pard\plain \ql
```

Mark Up: EDIFACT

```
'''ED2'''OPENET:1111111:OVT':003705655815:OVT'ABC1234567'0'TYP:ORDERS'N
RQ:1'''
UNA:+.?
'UNB+UNOC:2+003705655815:30+1111111:30+980729:2233+4++ORDERS911+++KKK
KATE+1'UNH+
1+ORDERS:001:911:UN:FI0030'BGM+640+1234567'DTM+4:19981201:102'DTM+2:199
90101:102'DTM+2:9901:616'RFF+BC:123'RFF+VN:123456'NAD+BY+003705655815:1
00'
NAD+SE+11111111::92'NAD+PL+53432::92++KAUPPA:KAUPUNKI+KATU
9+KAUPUNKI++00007'NAD+CN+-::ZZ++TERMINAALI+OVI 42+TOINEN
KAUPUNKI++00069'UNS+D'LIN+1++23442423234
:EN'PIA+5+3244:MF'PIA+5+2341234324:ZBU'PIA+5+234243:ZCG'IMD+F+8+-
::91:KUKKAPUR
KKI:SAVI'QTY+21:8:KPL'FTX+AAA+++T.HARMAA:V[RI'FTX+AAA+++10:KOKO'PRI+NTP
:7.23:+
RP:7.32:PE'TAX+7+VAT+++:::22.00'LIN+2++543434554345:EN'PIA+5+535:MF'PIA
+5+45:
PCE'UNT+38+2'UNZ+2+4'
'''EOF'''9'
```

Mark Up: XML

```
<fragment>
  <section>
    <title>Introduction</title>
    <para>Since the emergence of <acronym refid="xml">XML</acronym> in
    early 1998 and it's subsequent adoption across diverse application
    domains, one of the key benefits it enabled was the separation of
    content and presentation <bibref refloc="Bos97"/>. <acronym
    refid="xml">XML</acronym> borrowed this model (along with other
    important concepts) from the <acronym.grp><acronym
    refid="sgml">SGML</acronym><expansion id="sgml">Standard
    Generalised Markup Language</expansion></acronym.grp>. An
    <acronym  refid="sgml">SGML</acronym> document consists of
    logically structured content and uses a separate file (style
    sheet) to specify how the content should be formatted for
    [...]
    <figure id="img1">
      <title>ePublishing Components</title>
      <graphic href="02-04-03-fig01.jpg" width="321" height="214"/>
    </figure>
  </section>
</fragment>
```

What is SGML?

- Standard Generalised Mark-Up Language
- ISO standard since 1986
- Meta-language for defining document mark-up vocabularies
- Uses logical mark-up (structure, content) instead physical (how document looks on printed page)
- Platform-, system-, vendor- and version-independent documents
- Very powerful, but contains a number of complex features

```
<!DOCTYPE anthology [
 <!ELEMENT anthology - - (poem+)       >
 <!ELEMENT poem  - - (title?, stanza+)>
 <!ELEMENT title     - O (#PCDATA)     >
 <!ELEMENT stanza    - O (line+)       >
 <!ELEMENT line      O O (#PCDATA)     >
]]>

<anthology>
 <poem>
   <title> The SICK ROSE
   <stanza>
     <line>O Rose thou art sick.</line>
     <line>The invisible worm,</line>
     [...]
   </stanza>
   <stanza>
     <line>Has found out thy bed</line>
     <line>Of crimson joy:</line>
     [...]
   </stanza>
 </poem>
</anthology>
```

What is HTML?

- HTML, the *de facto* standard for publishing Web content, is an SGML vocabulary
- Supporting full SGML on the Web was too difficult so HTML made some simplifications
  - not extensible
  - limited structure
  - not content oriented
  - cannot be validated
- HTML is a simple language to understand and use
- Most of the content available on the Web has been created with HTML

```
<html>
  <head>
    <title>The SICK ROSE</title>
  </head>

  <body>
   <h1>The SICK ROSE</h1>

   <p>
     O Rose thou art sick.<br />
     The invisible worm,<br />
     [...]
   </p>

   <p>
     Has found out thy bed<br />
     Of crimson joy:<br />
     [...]
   </p>
  </body>
</html>
```

What is XML?

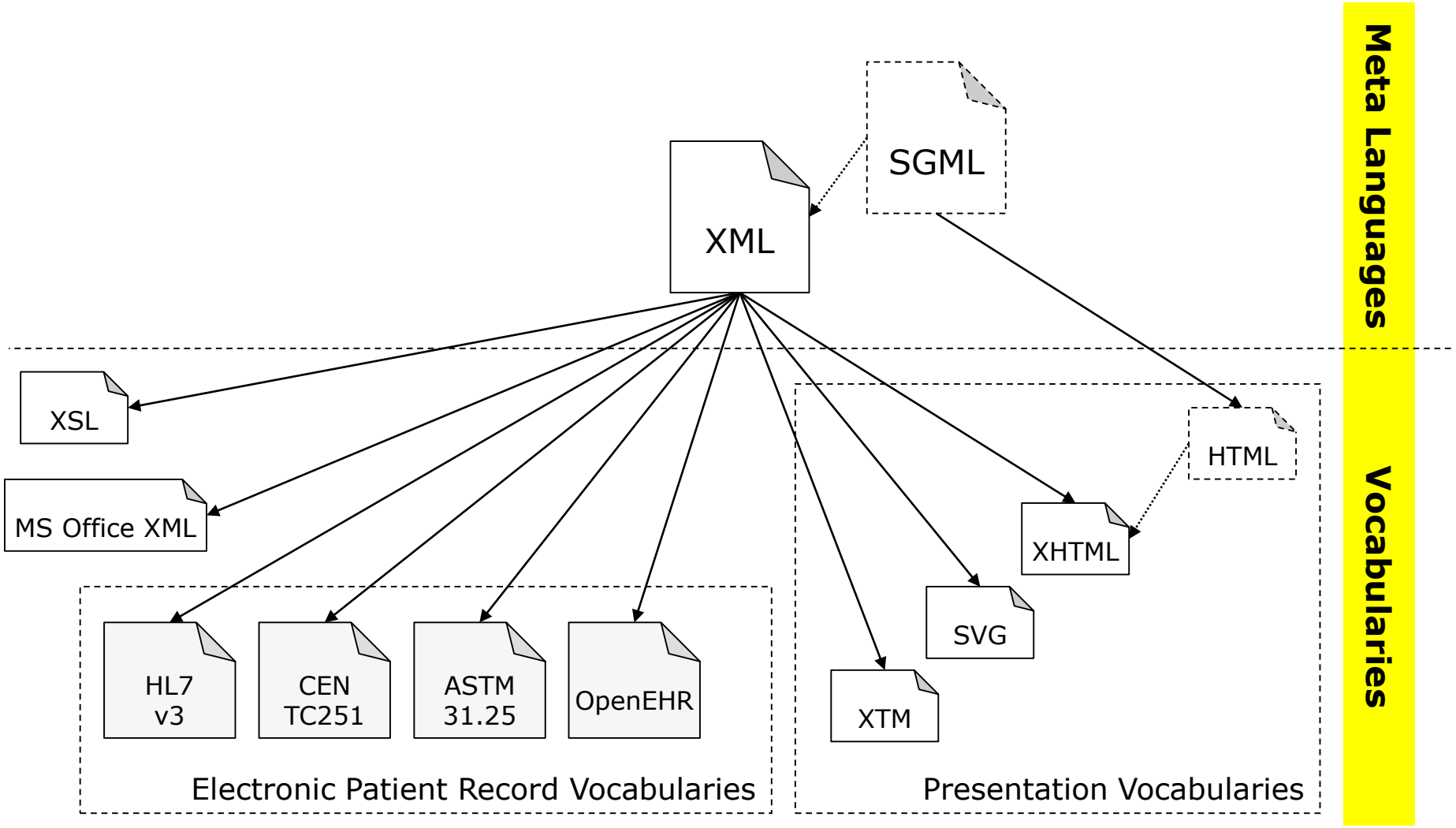
- eXtensible Markup Language
- XML is a simplified subset of SGML
- Can also be used to define document markup vocabularies (e.g. XHTML)
  - These can have a strictly defined structure (DTD)
- Retains the powerful features of SGML (extensibility, structure, validation)
- Ignores the complex features of SGML and is therefore easier to use and implement
- XML documents look similar to HTML documents
- Separates structure and presentation (like SGML)

XML Example

```
<?xml version='1.0' encoding='ISO-8859-1' standalone='yes' ?>
<doc type="book" isbn="1-56592-796-9" xml:lang="en">
  <title>A Guide to XML</title>
  <author>Norman Walsh</author>
  <chapter>
    <title>What Do XML Documents Look Like?</title>
    <paragraph>If you are ...</paragraph>
      <item>
        <paragraph>The document begins ...</paragraph>
      </item>
      <item>
        <paragraph>Empty elements have ...</paragraph>
        <paragraph>In a very ..</paragraph>
      </item>
    <section>...</section>
    ...
  </chapter>
  <chapter>...</chapter>
</doc>
```

XML Vocabularies you have come across? – student pov

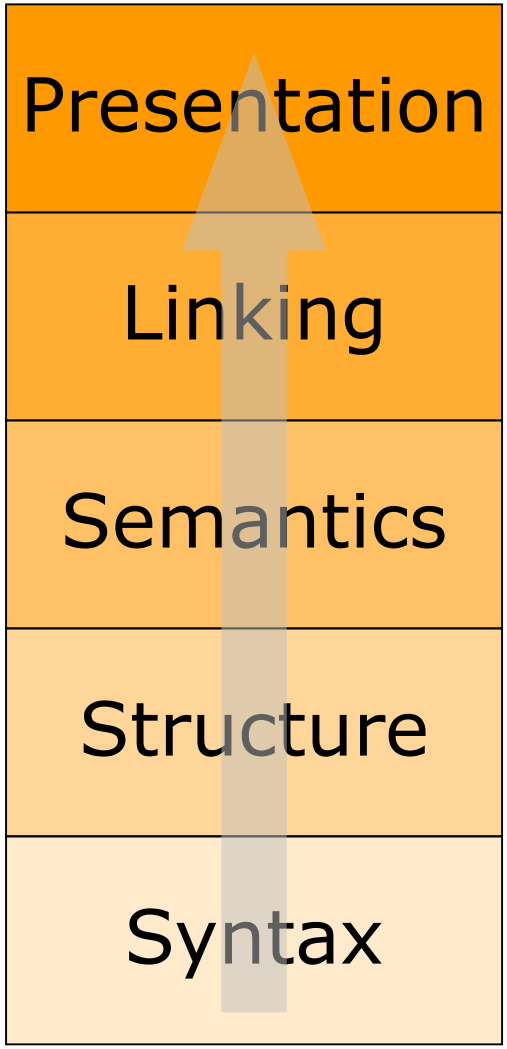
Meta Language vs. Vocabulary



Why is the emergence of XML an important development?

- XML is a tool for defining vocabularies
  - XML vocabularies are easy to read
  - XML is self describing
    - Parse tree embedded in document
    - Grammar for language referenced via DTD/Schema
- XML vocabularies are easy for computers to process, exchange and display
  - XML tools are ubiquitous, free and conform to established standards
  - Natural affinity with Object serialization
  - Data source neutral

XML technologies



CSS, Cascading Style Sheets
XSL, Extensible Stylesheet Language
XPath, XQuery

XLink, XBase
XPointer

Topics Maps, Ontology Web Language
Resource Description Framework (RDF)

XML Schema, RelaxNG, RDF Schema,
Document Type Definition (DTD)

XML Namespaces
XML 1.0

XML 1.0

- The XML 1.0 specification describes the syntax for XML documents (elements and attributes) and DTDs

- An XML document is a hierarchical data structure using self-definable tags
  - e.g. `<doc><author>[..]</author></doc>`

Design goals of XML 1.0 specification

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 in XML markup is of minimal importance.

Physical Parts of XML documents

Physical parts of XML documents

- XML Declaration or Prolog
- Document Type Declaration
- Elements
- Attributes
- Entities
- Character Data Sections
- Processing Instructions
- Comments

- XML Namespaces

XML Declaration or Prolog- 3 Examples

- Placed at the start of an XML document
- Informs XML software of
  - the version of XML the document conforms to
  - the character encoding scheme used in the document
  - whether or not a set of external declarations affect the interpretation of this document

https://www.w3schools.com/xml/xml_dtd_intro.asp

```
<?xml version="1.0" ?>
<?xml
   version="1.0“ encoding="UTF-
   8" ?>

<?xml
   version="1.0"
   encoding="UTF-8"
   standalone="yes" ?>
<!DOCTYPE person [
  <!ELEMENT person (name, adult,
   nationality)>
]>
<person> and so on </person>

<?xml version="1.0">
<!DOCTYPE person SYSTEM
   'person.dtd'>
<person> and so on </person>
```

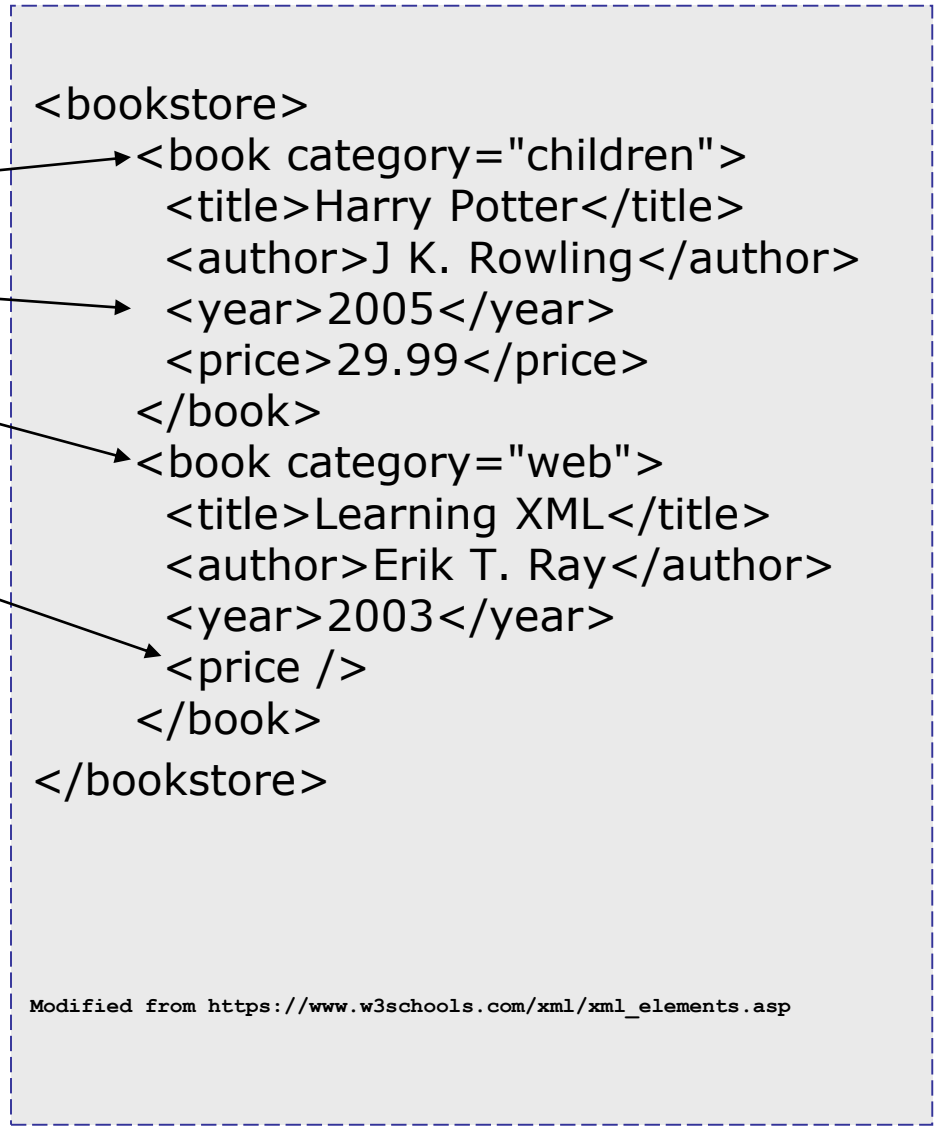
Elements

- Define logical structure and sections of XML documents
- Different content types:
  - Attribute Content
  - Text content
  - Element content
  - Empty.
  - A Mix of the above

- Each element must be completely enclosed by another element, except for the root

- Any Element name must start with a letter or underscore but after that can include also digits, fullstops, hyphens.

- Don't start with colon due to namespaces
- Don't include spaces



Attributes

- Provides additional information about an element
- Attributes are contained within the start-tag
- Consists of a name and associated value separated by an equals sign
- The attribute value must always be enclosed by quotes
- The order of attributes is insignificant

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

<doc type="book"
     isbn="0-201-71050-1">

 <title>Java Gently</title>
 <author>Judy Bishop</author>
 <chapter>
  <paragraph type="abstract">
    In this book ...
  </paragraph>
 </chapter>

</doc>
```

ELEMENT vs. ATTRIBUTE

- Lexically little difference,
- application specific,
- no hard/fast rules available.

https://www.ibm.com/developerworks/library/x-eleatt/

ELEMENT

- Constituent data,
- Used for content,
- White space can be ignored or preserved
- Nesting allowed (child elements),
- Convenient for large values, or binary entities.

ATTRIBUTE

- Inherent data,
- Used for meta-data,
- No further nesting possible (atomic data),
- Default values,
- Minimal datatypes

Entities

- Storage units for repeated text
  - Defined in a DTD
- Character entities are used to insert characters that cannot be typed directly
- XML contains a number of 'built-in' entities
  - "
  - '
  - <
  - >
  - &

```
<math>
  5 &lt; 6 and 6 &gt; 5
</math>

<copyright>
  &copyright-notice;
</copyright>

<bullet>
  XML contains a number
   of &apos;built-in&apos;
   entities
   <list>
     <item>&amp;quot;</item>
     <item>&amp;apos;</item>
      <item>&amp;lt;</item>
      <item>&amp;gt;</item>
      <item>&amp;amp;</item>
   </list>
</bullet>
```

Character Data Sections

- Data which is to be parsed is called PCDATA
- An XML parser will not treat the contents of a CDATA section as markup
  - Used to simplify mark-up by escaping a selection of text
- Entity references are not resolved
- Useful for including source code in XML

```
<![CDATA[
  You don't need to escape
  special characters in CDATA
  sections, such as <, >, &, ,
  ' and ".
]]>
```

```
<![CDATA[<<< STOP now >>>]]>
```

```
<![CDATA[<?xml version='1.0'?>

<person>
  <name>Mike</name>
  <age>24</age>
</person>]]>
```

Processing Instructions

- Pass additional information to application (e.g. parser)
- Application-specific instructions
- Consists of a PI Target and PI Value
- Processed by applications that recognise the PI Target

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

<?xml-stylesheet
   type='text/css'
   href='style.css'?>

<?xml-stylesheet
   type='text/xsl'
   href='style.xsl'?>

<?myapp filename='test.txt'?>
```

Comments

- Used to comment XML documents
- Not considered to be part of an XML document
- An XML parser is not required to pass comments to higher-level applications

```
<!-- one-line comment -->

<!--
This
is a
multi-line comment
-->
```

Well formed XML

- XML Declaration required
- At least one element
  - Exactly one root element
- Empty elements are written in one of two ways:
  - Closing tag (e.g. "<Name></Name>")
  - Special start tag (e.g. "<Name />")
- For non-empty elements, closing tags are required
- Attribute values must always be quoted
- Start tag must match closing tag (name & case)
- Correct nesting of elements
  - Example incorrect nesting and incorrect case

```
<full_name>
<first_name>
John </Full_name>
</first_name>
```

Exercise

- Spot the 6 deliberate errors in the XML opposite

```
<?xml version="1.0"!>
<!DOCTYPE catalog SYSTEM "books.dtd">
<catalog>
  <book id=’bk101’ type=’softback’>
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
      <price>44.95</price>
    <publish_date>2000-10-01</publish-date>
    <description>An in-depth look at creating applications with XML.
  </book></description>
<book id=’bk102’ >
    <author nationality=irish>Jenkins, Fred</Author>
    <title>XML Technology Guide</title>
      <price>50.00</price>
    <publish date>2000-10-01</publish date>
    <description>An in-depth look at using XML
technologies.</description>
      <stocked_by>Easons</stocked_by>
      <stocked_by>Amazon</stocked_by>
  </book type=’hardback’>
</catalog>
```

```
<?xml version="1.0"!>
<!DOCTYPE catalog SYSTEM "books.dtd">
<catalog>
  <book id=’bk101’ type=’softback’>
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
      <price>44.95</price>
    <publish_date>2000-10-01</publish-date>
    <description>An in-depth look at creating applications with XML.
  </book></description>
<book id=’bk102’ >
    <author nationality=irish>Jenkins, Fred</Author>
    <title>XML Technology Guide</title>
      <price>50.00</price>
    <publish date>2000-10-01</publish date>
    <description>An in-depth look at using XML
technologies.</description>
      <stocked_by>Easons</stocked_by>
      <stocked_by>Amazon</stocked_by>
  </book type=’hardback’>
</catalog>
```

Exercise and Hand up

- Write down a “well formed” XML snippet, using elements and/or attributes, describing:
  - Your name (distinguishing first, middle, surname)
  - Student ID
  - Favourite music groups
  - County
  - Expected date of graduation

```
XML Declaration required
Exactly one root element
Empty elements are written in
    one of two ways:
   Closing tag or Special
       start tag
For non-empty elements,
    closing tags are required
Attribute values must always
    be quoted
Start tag must match closing
    tag (name & case)
Correct nesting of elements
```

What is a DTD?

- Document Type Definition,
- Defines structure/model of XML documents
  - Elements and Cardinality
  - Attributes
  - Aggregation
- Defines default ATTRIBUTE values
- Defines ENTITIES
- Stored in a plain text file and referenced by an XML document (external)
- Alternatively a DTD can be placed in the XML document itself (internal)

Element Type Declaration

- Define grouping of elements
  - (", ")"
- Define sequence of elements
  - ",": followed-by (Sequence)
  - "|": logical or (Choice)

```
<!ELEMENT doc
   (title, author, editor,
   chapter, appendix)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author
   (name | synonym)>

<!ELEMENT image EMPTY>

<!ELEMENT paragraph
   (#PCDATA | bold | italic)*>
```

Element Type Declaration

- Define occurrences of elements
  - ?: zero-or-one
  - +: one-or-more
  - *: zero-or-more

```
<!ELEMENT doc
   (title, author+, editor?,
   chapter+, appendix*)>

<!ELEMENT chapter
   (title,
    (section+ | paragraph+))>

<!ELEMENT list
   (item?, item?, item)>

<!ENTITY % list "ordered |
   unordered | definition">

<!ELEMENT paragraph
   (#PCDATA | %list;)*>
```

Entity Declaration

- Internal entities
  - Built-in
- External entities
  - References to a file (text, images etc.)
- Parameter entities
  - Used inside DTDs

```
<!ENTITY author
   "Norman Walsh, Sun Corp.">
```

```
<!ENTITY copyright
   SYSTEM "copyright.xml">
```

```
<!ENTITY % part
   "(title?, (paragraph |
   section)*)">
```

Attribute List Declaration

- Define type of attribute
  - CDATA
  - ID
  - IDREF
  - ENTITY
  - NMTOKEN
  - NOTATION
- Define default values of attributes
  - #REQUIRED
  - #IMPLIED
  - #FIXED
  - A list of values with default selection

```
<!ATTLIST person
   ssn ID #IMPLIED>

<!ATTLIST adult
   age CDATA #REQUIRED>

<!ATTLIST mml
   version '1.0' #FIXED>

<!ATTLIST person
    sex (m | f) #REQUIRED>

<!ATTLIST day
   temperature (l | m | h) "l">
```

Simple DTD Example

```
<!DOCTYPE doc[
<!ENTITY % part "(title?, (paragraph | section)*)">

<!ELEMENT doc (title, author+, chapter+, appendix*)>
<!ATTLIST doc type (book | article) "book“
              isbn CDATA #REQUIRED>

<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT chapter %part;>
<!ELEMENT appendix %part;>
<!ELEMENT section %part;>
<!ELEMENT paragraph (#PCDATA | url | ol)*>
<!ATTLIST paragraph type CDATA #IMPLIED>
<!ELEMENT ol (item+)>
<!ELEMENT item (paragraph+)>
<!ELEMENT url (#PCDATA)>
]>
```

Over to you…

- Possible DTD for following?

```
<database>
<person age='34'>
      <name>
            <title> Mr </title>
            <firstname> John </firstname>
            <firstname> Paul </firstname>
            <surname> Murphy </surname>
      </name>
      <hobby> Football </hobby>
      <hobby> Racing </hobby>
</person>

<person >
      <name>
            <firstname> Mary </firstname>
            <surname> Donnelly </surname>
      </name>
</person>
</database>
```

Over to you…

```
<database>
<person age='34'>
     <name>
          <title> Mr </title>
          <firstname> John </firstname>
          <firstname> Paul </firstname>
          <surname> Murphy </surname>
     </name>
     <hobby> Football </hobby>
     <hobby> Racing </hobby>
</person>

<person >
     <name>
          <firstname> Mary </firstname>
          <surname> Donnelly </surname>
     </name>
</person>
</database>
```

```
<!DOCTYPE database [

<!ELEMENT database (person*)>

<!ELEMENT person (name,hobby*)>
<!ATTLIST person age CDATA #IMPLIED>

<!ELEMENT name (title?, firstname+,
   surname)>

<!ELEMENT hobby (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT surname (#PCDATA)>

]>
```