Extensible Markup Language - XML

XML has exploded as one of the most important data management tools. Its evolution has been as follows:

1. It was designed by the WWW consortium as a stripped down version of SGML more suitable for web design -
   - In particular, it separates content structure from format
2. It then became the emerging standard for describing the structure of computerised textual documents
   - or for any semi-structured document
3. It has become a general purpose language for data description and interchange, which has led to:
   - the development of vocabularies for specific disciplines
   - the addition of database functionality - searching, etc.
   - the separate ability to add formatting to XML documents
4. HTML has been recast in XML as XHTML

Four "Languages" Compared

SGML - The original meta-language which is loosely defined in some respects and so is not good for computers

precise subset →

XML - A much more precise meta-language which is better for computers

defined using ↓

defined using ↓

HTML - The original web language which is loosely defined and so hard to validate

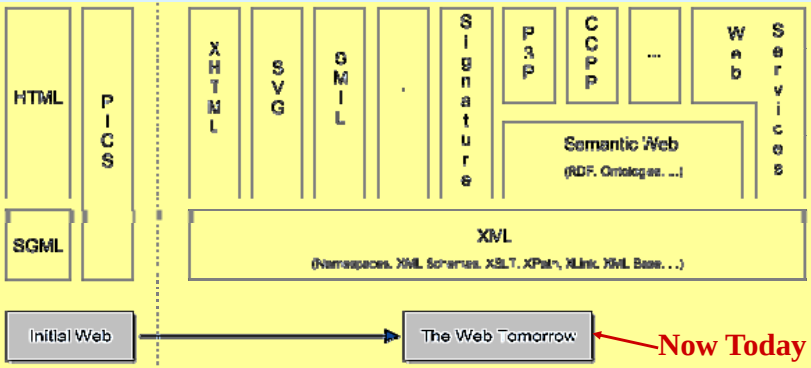
precise subset →

XHTML - A restriction of HTML to fit with the rules of XML and is easier to validate and to alter the presentation

Why XML is Important to the Web

Note XML is at the heart of everything:

everything else is described in terms of XML



How XML Differs from HTML

An XML document like HTML consists of a number of **tagged sections** called **elements**, each of which may have **attributes**. However:

HTML tags mix format and structure and are not about the meaning of the data, while in XML:

- **Tags** only define structure
- **Formatting** is entirely separated and is not part of XML

The structure of XML is more tightly controlled:

- Tags are **case sensitive**
- You **must have start and end tags** – a hierarchical structure of elements is enforced

You can freely introduce **new tag types** (and variables)

XML is more than a language - it has lots of **support tools**

formatting (XSL), querying (XQuery), schema definition (XML Schema), querying and integration with databases

Example

Here is an XML description of a CD

NB, the tags, CD, ARTIST and so on need not be predefined:

```
<CD CatalogNumber = "COOK CD 201">
   <ARTIST>AC Acoustics</ARTIST>
   <TITLE>Understanding Music</TITLE>
   <LABEL>
     <NAME>Cooking Vinyl</NAME>
     <ADDRESS>PO Box 1845, London W3 0ZA, England</ADDRESS>
   </LABEL>
</CD>
```

An XML document is a hierarchy of elements ultimately made up of text strings

Semi-Structured Data

Databases store **structured data**

- Every piece of data supplied must fit into the structure of the database and the structure is fixed and applies to all data instances
- To store CD information we will have to create a specific set of tables each having a specific set of columns.
- If we suddenly discover another attribute we want to store, we must edit the schema and create a new column and then fill this for the existing data

Semi-structured data is data for which individual parts can have their meaning identified, but they don't have to respect an external structure completely

- e.g., we can add `<LOCATION>Glasgow, Scotland</LOCATION>` to this CD only

Document Type Definition

An XML document marked up with hierarchically structured tags is called **well-formed XML**

- An XML parser will be able process it and make use of the structure that it identifies

However, for **sharing** data around a community, it will still be useful to pre-define the structure expected in the same way as a schema is produced for a database

- Then everyone can use software built on top of this structure
- The structural definition takes the form of a description of the tags which will appear
- This is called a **Document Type Definition** (DTD) or an **XML Schema**
- When a document has an associated DTD, the file is called **valid XML**

The DTD can either appear in the file (at the top) or separately

- With a DTD, associated software can make more extensive checks of the **correctness** of the file

Vocabularies and Namespaces

Vocabularies or **Namespaces** are sets of tag definitions which are shared amongst a community, e.g.:

- Wireless Markup Language (WML) – similar to XHTML for mobile phones
- Bioinformatic Sequence Markup Language – gene mapping & sequencing data
- Business and legal vocabularies
- Scaleable Vector Graphics – describe drawings
- SMIL – for multimedia presentations
- SOAP - describes language independent distributed object parameter passing

Vocabularies are shared by posting a DTD or XML schema on the internet:

e.g. `<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">`

defines xsd as a list of names used in XML Schema descriptions

so that in `<xsd:attribute name="assumed" type = "xsd:boolean" />`

the terms *attribute* and *boolean* are defined as in the XMLSchema namespace

XML Document Structure

An XML document consist of **three** parts:

- a **prolog** – describing the document and including:

  the **XML declaration**

  <?xml version="1.0" ?>

  with attributes:

  version – must be 1.0

  encoding – how characters are encoded in the file

  standalone – "yes" if this document is entirely self-contained, "no" if it has an external DTD

- the **Document Type Declaration** which describes the DTD (see later slides) or refers to an external DTD, e.g.

  <!DOCTYPE Catalog SYSTEM "http://www.dcs.gla.ac.uk/~rich/Catalog.dtd">

- the **body** – containing one or more elements

- an optional **epilog** – containing comments and processing instructions

Example

The DTD

```
<?xml version="1.0" standalone="yes" ?>
<!ELEMENT CONVERSATION (TITLE, SPEECH*)>  //TITLE then 0 or more SPEECHes
<!ELEMENT TITLE  (#PCDATA)>               //TITLE is just a piece of text
<!ELEMENT SPEECH  (PERSON+)>              //SPEECH is one or more PERSONs
<!ELEMENT PERSON  (#PCDATA)>              //PERSON is just a piece of text
<!ATTLIST PERSON  name #REQUIRED>         //PERSONs must have a name attribute
```

The Data

```
<CONVERSATION>
<TITLE>A Conversation between Richard and Annie</TITLE>
<SPEECH>
<PERSON name="Annie">Why XML?</PERSON>
<PERSON name="Richard">XML conforms to information, allowing document authors to create
    markup languages that work for them.</PERSON>
</SPEECH>
<SPEECH>
<PERSON name="Richard">With HTML, document authors become frustrated, trying to fit their
    information sets into a fixed markup language.  </PERSON>
<PERSON name="Annie">Can authors who are used to HTML, but only want a few more elements,
    use XML? </PERSON>
<PERSON name="Richard">With XML, anything is possible. You could extend or even contract
    HTML, depending on your needs. </PERSON>
</SPEECH>
</CONVERSATION>
```


SVG Example – Three Circles

Prolog

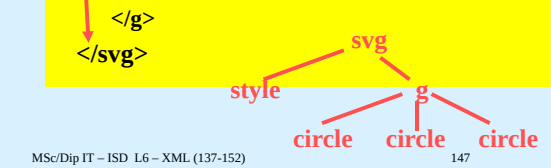
DTD

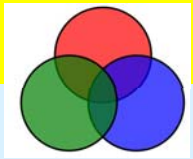
```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
    <style type="text/css"> circle:hover {fill-opacity:0.9;} </style>
    <g style="fill-opacity:0.7;">
        <circle cx="6.5cm" cy="2cm" r="100" style="fill:red; stroke:black;
                stroke-width:0.1cm" transform="translate(0,50)" />
        <circle cx="6.5cm" cy="2cm" r="100" style="fill:blue; stroke:black;
                stroke-width:0.1cm" transform="translate(70,150)" />
        <circle cx="6.5cm" cy="2cm" r="100" style="fill:green; stroke:black;
                stroke-width:0.1cm" transform="translate(-70,150)" />
    </g>
</svg>
```

List of tag names

Attributes – cx, cy, r, style, transform

Document Element





Elements

An element is a typed fragment of the document delimited by a start tag (<tag>) and an end tag (</tag>).

Elements can contain:

- **character data content** (CDATA) text only - e.g. TITLE
- **element content** – including other elements as children
  e.g. <SPEECH> or <CONVERSATION>
- **mixed content** – a mixture of elements, text, etc. –
  A paragraph withsome emphasisedtext
- **empty elements** which contain nothing at all – this is indicated as <tag></tag> or more simply <tag/>
  <EOF/> or <hr/>

Attributes

Elements can have **attributes** and each of these is defined to have:

- a **name**
- a **type**
  - not integers, booleans, etc. but whether it identifies the element (ID), or is a piece of text (CDATA), among other things
- a **qualification**
  - whether it must have a value or not, a default values, etc.

So some of the text in an XML file is held between tags and some in attributes

- it can be hard to decide what to put where!

What's in a DTD?

A DTD contains definitions of element types and their attributes

An element type is defined in terms of:

- its **name**
- its **content**, which is either empty, a string, a set of components or a mixture of components and strings
- the components can be in a **sequence** or as a set of **alternatives**
- each component can be specified to occur exactly once, at most once, at least once or any number of times
- the **attributes** it has (as an attribute list defined in terms of the element type)

  <!ATTLIST PERSON name #REQUIRED>

An improved mechanism for describing an XML document structure uses XML itself – this is **XML Schema**

Example DTD vs XML Schema

```
<!ELEMENT Name  (Title?, First, Initial?, Last)>
<!ATTLIST Name assumed (“Yes” | “No” ) “No”>
<!ELEMENT Title (#PCDATA)>                    …ditto for First, Initial, Last
```

```
<xsd:schema xmlns:xsd=″http://www.w3.org/2001/XMLSchema″>
    <xsd:element   name = “Name″ type="nameType" />
    <xsd:complexType name = “nameType″>
        <xsd:sequence>
            <xsd:attribute name=″assumed″ type = “xsd:boolean″ />
            <xsd:element name = “Title″  type = “xsd:string″
                                    minOccurs=″0″  maxOccurs = “1″/>
            < xsd:element name = “First″  type = “xsd:string″/>
            ..... etc. for Initial and Last
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

User defined type replaces Model Group

Types can be specified precisely

Number of repetitions can be specified precisely

Programming XML

There are two programming techniques used for managing XML documents (both usually achieved in Java)

- **DOM programs** read the whole file in and then build a hierarchical structure which you can then navigate about
  - e.g. start from the whole document, go down to the first book, then to the title and test if it is “Emma″, if it is go back up and down to the author of that book and print it out
- **SAX programs** read the file a fragment at a time and contain methods which are called when the following are encountered:
  - the start and end of the document
  - a start tag and an end tag
  - a sequence of character data
    - e.g. the start tag method could count every time a <title> element is encountered