Standard Generalised Markup Language (SGML)

- SGML is the parent of XML. It started as GML (within IBM) in late 70’s. Charles Goldfarb was the major architect
- The vast range of possible document tags could not be described in a single specification.
- SGML is thus a *metalanguage* used to describe tags with which documents can be marked up.
- So SGML is not a fixed tagset. It describes a standard set of ‘punctuation’ for tags, plus char. sets to be used etc.
- SGML ISO standard dates from 1986.
- SGML almost totally superseded by XML over last few years

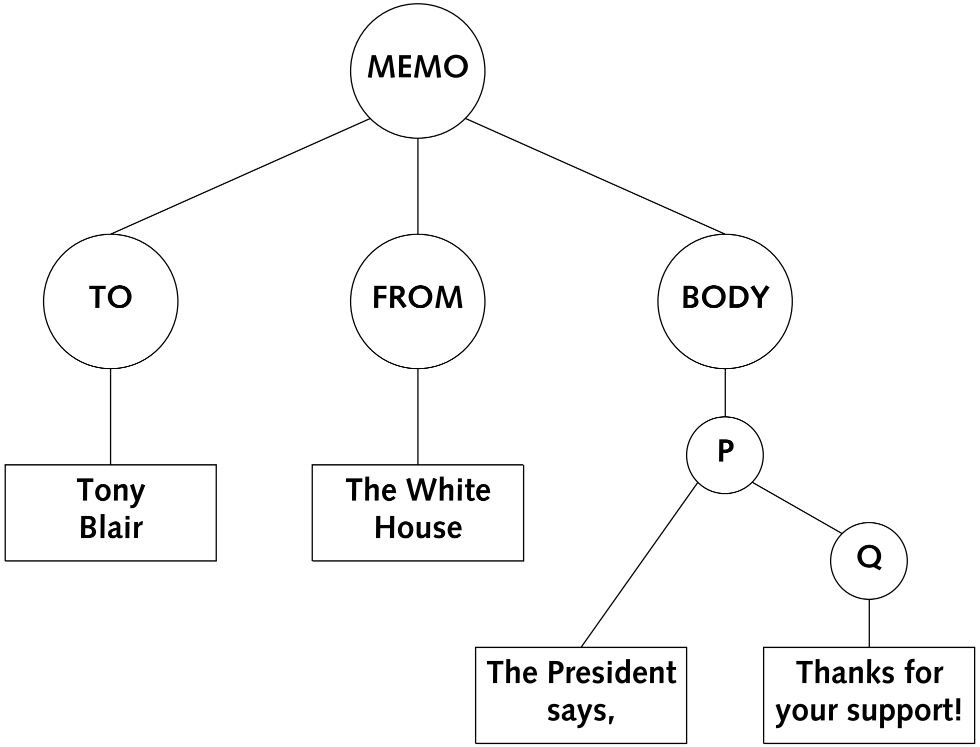
A SAMPLE MEMO

To: Tony Blair

From: The White House

Message: The President says, “Thanks for your support!”

The XML-compliant memo as a tree



- NOTE: All *well-formed* XML docts. have tree structure.
- NOTE: All *valid* XML docts. have a tree structure whose tag usage is 'correct' w.r.t. the DTD.

SGML: A simple Tagged Memorandum

```
<MEMO>
<TO> Tony Blair </TO>
<FROM> The White House </FROM>
<BODY>
<P> The President says,
<Q> "Thank you for your support!" </Q>
</P>
</MEMO>
```

- Note omission of **`</BODY>`** i.e. ‘end of the body’
  This needed to be requested in the ‘tag spec.’ (the DTD)
- Even permissible to omit start tags if DTD allows it!
- Omitted tags can easily lead to ambiguous grammars. Onus on DTD designer to avoid ambiguity because SGML parser can’t (in general) decide or detect it.

SGML: A simple DTD for a memo

```
<!ELEMENT MEMO -- ((TO & FROM),BODY) >
<!ELEMENT TO -O (#PCDATA) >
<!ELEMENT FROM -O (#PCDATA) >
<!ELEMENT BODY -O (P)* >
<!ELEMENT P -O (#PCDATA | Q)* >
<!ELEMENT Q -- (#PCDATA) >
```

Document Type Definitions (DTDs)

- A set of tag definitions forms a DTD.
- The DTD’s own metasyntax is *similar* to that of the tags — but not totally identical.
- SGML metasyntax similar in purpose to BNF for defining programming languages
- A DTD for a *memo* will obviously be different from a DTD for a *menu* or a DTD for a *report*.
- There are many existing DTDs (e.g. for HTML and in publishing) Also many ‘in house’ DTDs.

SGML Parsing

- SGML document needs to know character set to be used (e.g. `UTF8`)
- SGML *always* required DTD at parsing time
- SGML parses a document using a given tagset with respect to the DTD that defines that tagset.
- SGML parser can check that DTD conforms to the SGML standard
- Optional *tag minimisation* can make SGML parser's job very difficult indeed.
- Reliable parsers for full SGML began to appear in the 80's (Usually cost a fortune ... used by document professionals only)

SGML: Structure vs. Appearance

- The memorandum's *appearance* cannot be determined from the tagged source.
- SGML and XML often used to define abstract 'structural' markup — but *not always so* (see SVG later on).
- **Tag names have no intrinsic meaning, but all programs must agree on the *semantics* of what they mean**
- The structure of the document as defined by the tags makes it easy to apply database technology to class of documents defined in DTD.

SGML: 'Multi-purposing' an SGML document

- Possible to start with SGML tagged doct.
  Process SGML to Quark or Pagemaker for typesetting.
- But there is a big 'semantic gap' between
  SGML and PostScript (say).
- Hard to control both structure and typeset details
  from a single SGML DTD. (DTD gets enormous if you try).
- SGML best as source of structured data which
  can then be manipulated in a variety of ways.
- Styling best left to stylesheets (e.g. CSS2
  or XSL-FO for the Web).

SGML applications

- A specific tagset specified via a SGML DTD is called an *application* of SGML e.g. CALS, TEI.
- Some WYSIWYG software can generate SGML tagged docs. from their own internal data structures
- Examples include software from ArborText, SoftQuad, Adobe (Frame + SGML) etc. These now adapted for XML.
- This software was *never* cheap! Writing full SGML parsers is *hard*!
- HTML was the first ‘mass market’ SGML application. People at last realised what SGML notation could achieve.

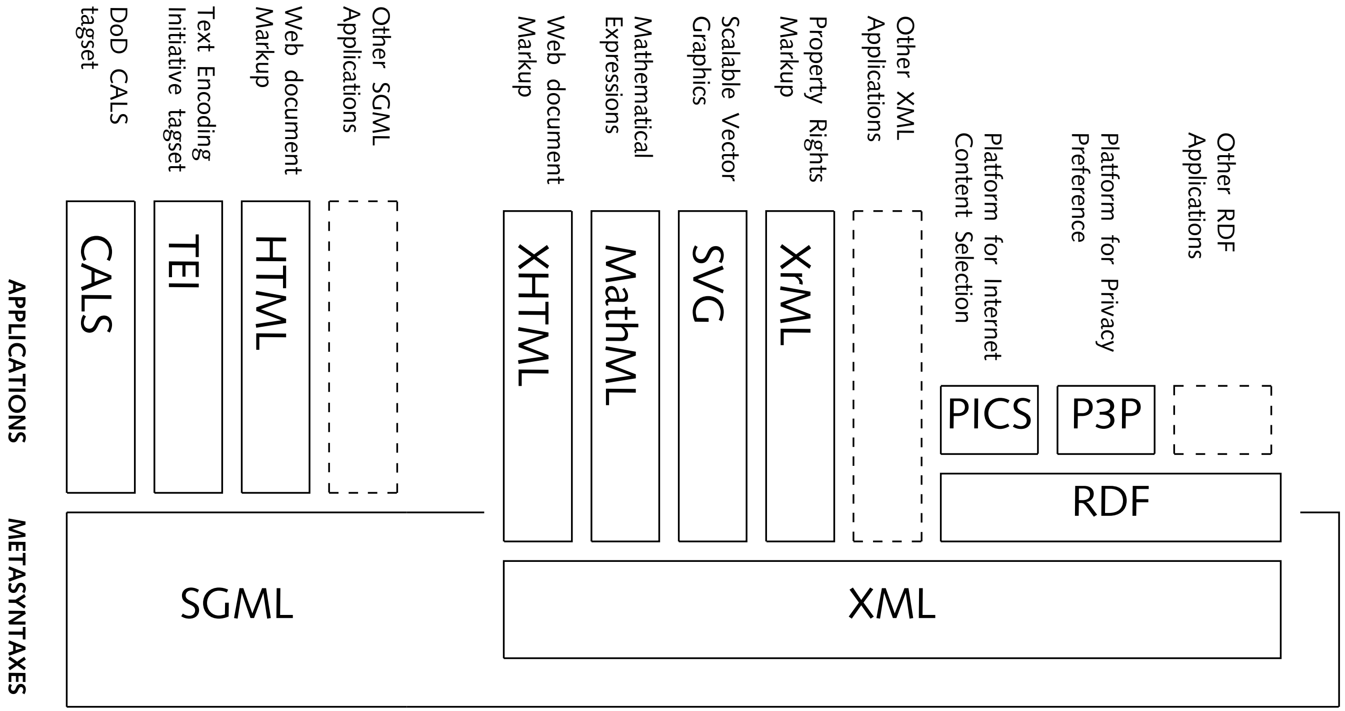


Diagram to show SGML/XML *application* and *subset* relationships.

Note that:

- XML is a *subset* of SGML
- HTML etc. are *applications* of SGML
- XHTML etc. are *applications* of XML
- RDF is an *application* of XML
- PICS, P3P etc. are *applications* of RDF

What about HTML?

- HTML adopted SGML metasyntax from the outset so it's an *application* of SGML.
- But it is essentially a *fixed* tagset so use of 'ML' suffix lamentable! Initial tagset was arbitrarily extended, in different ways by IE and Netscape.
- In early days Tim Berners-Lee and browser vendors didn't fully realise importance of having DTD for HTML
- Net result was chaos. IE and Netscape had different tags and different minimisation possibilities
- Allowing 'overlapping hierarchies' as well as omitted end tags is *deadly*. More than 95% of Web HTML was illegal! XHTML cleanup is now under way (see later).

More problems in HTML

- Lack of a DTD and SGML knowledge means IE and Netscape allow overlapping hierarchies though SGML forbids this e.g **`<P> <BOLD> ... </P> </BOLD>`**
- SGML is essentially a specification for tree-structured docts. with nested features.
- Overlapping hierarchies coupled with end tag omissions are **bad news**. Ambiguity even worse.
- HTMLs problems led to a call for it to be re-specified in SGML and for DTD to be available to browsers and rigidly enforced!
- Browser vendors said emphatic *No!* to full SGML parser in all browsers. So – enter XML.

SGML/XML: What is XML?

- Work started in 1996 on e**X**tensible **M**arkup **L**anguage.
- 80 SGML experts under aegis of W3C and chaired by Jon Bosak (Sun Microsystems) aided by Tim Bray.
- XML to be easy to parse and yet be proper subset of SGML
- XML parser to be able to check *well-formedness* (without DTD) and *validity* (with DTD).
- Must support stylesheet mechanism and syntax for hyperlinking and *namespaces* e.g **`<memo:from>`**

SGML/XML: XML design goals

(1) XML to be easy to use over Internet

(2) XML to support wide variety of apps.

(3) XML to be proper subset of SGML

(4) Must be easy to write progs. that process XML

(5) XML ‘optional features’ to be kept to minimum.

SGML/XML: More XML design goals

(6) XML to be human legible and reasonably clear

(7) XML design to be prepared quickly

(8) Design of XML to be formal and concise

(9) XML docs. to be easy to create

(10) Terseness in XML markup to be of minimal importance

Some SGML/XML differences

- In XML start and end tags must **always** be present.
- This allows well-formedness check without a DTD
- No ‘comments within comments’ or ‘comments within element declarations’. *Nightmare* to parse in SGML.
- Element attributes e.g. **`colour="blue"`** must be quoted
- **`&`** connector forbidden in element declarations. Must use **`,`** and | only.
- Lots of other more detailed differences (e.g. no ‘inclusions’ and ‘exclusions’)
- XHTML is HTML tagset made XML compliant

XML background reading

- Tim Bray's annotated XML spec is good start
  `http://www.xml.com/axml/axml.html`
- *Just XML* by John E. Simpson is very good for complete beginners (XML/DTDs/CSS2/Xlink/Xpointer).
- *XML— how to program* by Deitel et al. is comprehensive on XML applications.
- *Essential XML* by Box, Skonnard and Lam has useful extra material on *schemas*. See also:

`http://msdn.microsoft.com/msdnmag/issues/0800/XSLT/XSLT.asp`

- Many parsers and toolkits available. IE 6.0 supports XML with DTDs/Schemas and CSS2. Support not yet complete for XSL, XLink, Xpointer.

XML—forms of character data

- **#PCDATA** is Parsed Character Data. This means that the XML parser will seize on 'reserved characters' such as **<**, **>** and **&** unless they are escaped.
- **CDATA** is just 'character data'. You can have any characters, with no need to escape except for **<** which must be escaped.
- Remember that characters set will be as declared at the top of your document. Often UTF8 but XML supports full Unicode set.
- **NMTOKEN** characters are restricted to charset that can be used in a tag: letters, digits, underscores, hyphens, periods, colons. (But 'letters' can potentially be Unicode).
- SGML folded tags to upper case so **<MENU>**, **<menu>** and **<MeNu>** all mean the same thing. In XML (case sensitive) *they are all different*. Take care!

Revised (XML) DTD for a memo

```
<!-- Note that -O for ‘omittability’ now absent-->
<!-- Note that & has vanished in MEMO element declaration-->
<!ELEMENT MEMO ((TO,FROM)|(FROM,TO),BODY) >
<!ELEMENT TO (#PCDATA) >
<!ELEMENT FROM (#PCDATA) >
<!ELEMENT BODY (P)* >
<!ELEMENT P (#PCDATA | Q)* >
<!ELEMENT Q (#PCDATA) >
```

Revised XML-compliant memorandum

```
<?xml version="1.0"?>
<!DOCTYPE MEMO SYSTEM "memo.dtd">
<MEMO>
<TO> Tony Blair </TO>
<FROM> The White House </FROM>
<BODY>
<P> The President says,
<Q> "Thanks for your support!" </Q>
</P>
</BODY>
<!-- Notice the above line now essential -->
</MEMO>
```

- Note the *required markup declaration* (RMD) for version of XML to be used
- The next line (the *document type declaration*) must stipulate where DTD is to be found if the doct. is to be *validated*