

XML/XHTML/HTML

Vermont State University
Peter Chapin

Markup Languages

- “Plain text” documents with special commands
 - PRO
 - Plays well with version control and other program development tools.
 - Easy to manipulate with scripts and third party programs.
 - WYSIWYG in the sense that no control codes are hidden.
 - CON
 - Requires additional processing to create final output
 - Can be hard to learn
 - Not WYSIWYG in the sense that the document doesn't look like its final form.

Example: nroff

- Used for Unix manual pages...
 - .SH NAME
watch \- watch for a user to log in or out of the system
.SH SYNOPSIS
.BR "watch " [-d " delay"] [-s|-q] [-f " logfile"] "
username"

.SH DESCRIPTION
.B watch
is used to watch for the log in and log out activity of a particular user. It can record its findings to a file and/or display information to stdout.

Example: LaTeX

- Used extensively for technical literature...
 - ```
\begin{document}
\title{CIS--3152 Lab \#4\UDP and the Domain Name System}
\author{\copyright\ Copyright 2009 by Peter Chapin}
\date{Last Revised: January 31, 2009}
\maketitle

\section*{Introduction}
In this lab you will explore UDP and the domain name system
by writing a program that does DNS queries manually.
Although there are well known library functions for sending
queries to a name server (\texttt{gethostbyname()} and \
\texttt{gethostbyaddr()}), you will not use those functions
in this lab...
```

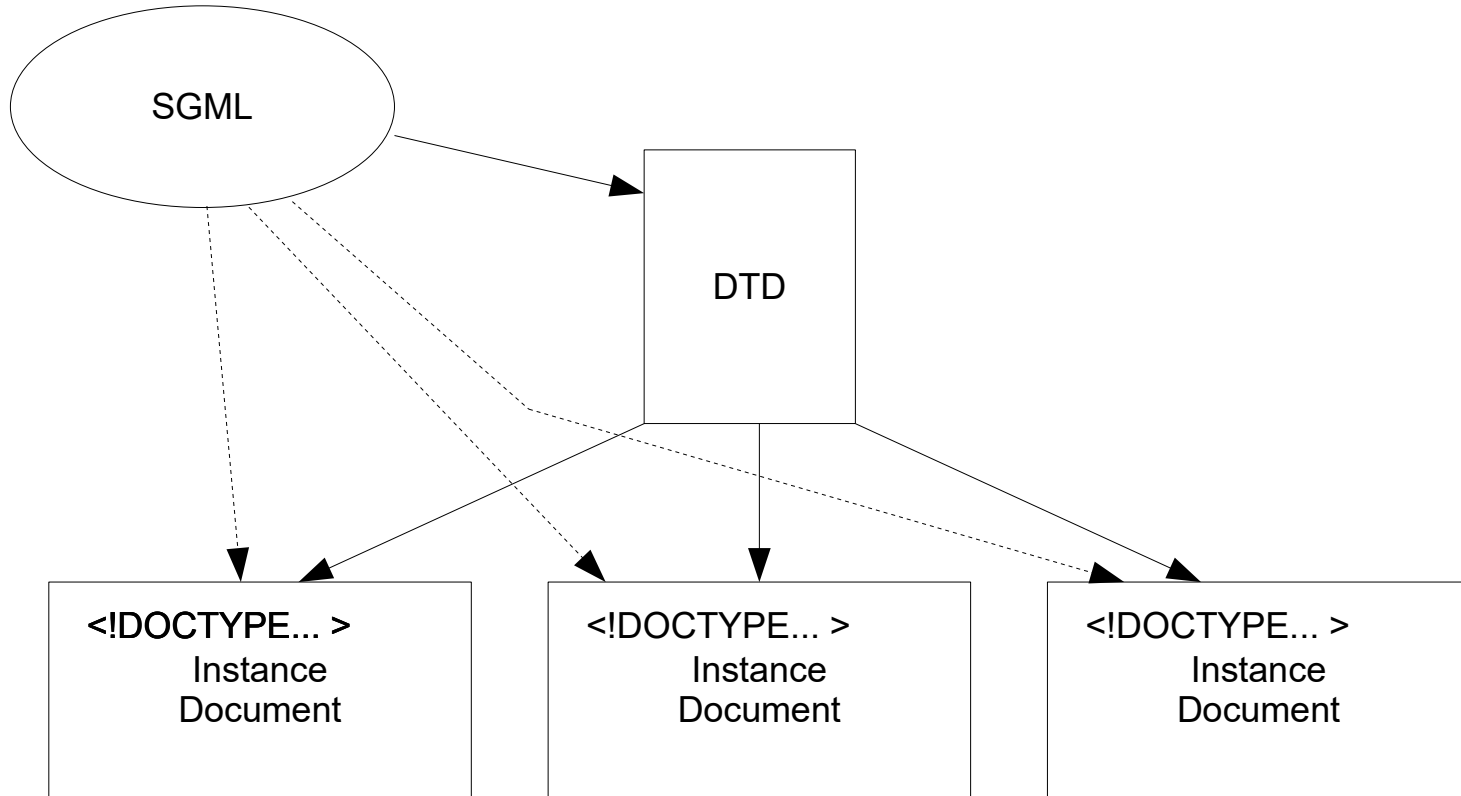
# Who Cares About XML?

- XML is one of many competing structuring tools
  - PROs:
    - Standardized by the W3C (World Wide Web Consortium)
    - Strong typing (many schema languages to choose from)
    - Extensible (Create your own “vocabularies”)
    - Many related standards: XInclude, XQuery, XSLT, XPath
  - CONS:
    - Verbose (i.e., high overhead)
    - Text (binary formats such as XDR, ASN.1 can be more compact)

# In the Beginning: SGML

- Standard Generalized Markup Language
  - A markup language to define other markup languages (a “meta markup language”).
    - ISO standard since the 1980s.
    - Powerful and expressive.
    - Complicated and difficult to implement fully.
  - How it works...
    - Use SGML to write a “document type definition” (DTD) for your markup language.
    - DTD describes the vocabulary of the markup.
    - SGML rules apply in the marked up documents too.

# How It Works



# “The Look”

- All SGML documents have a similar look.
  - `<body>`  
`<p>Mark up is in the form of  
tags surrounding blocks of  
material.</p>`  
  
`<p nature="important">Start tags  
can be decorated with attributes.  
My name is &author;. "Entities" are  
prefixed with &amp;.</p>`  
`</body>`
- The precise names used in the tags depends on the DTD.

# HTML

- If “the look” seems familiar, I'm not surprised.
  - HTML 4 is an SGML application.
    - Meaning that HTML is described by an SGML document type definition.
    - <http://www.w3.org/TR/REC-html40/sgml/dtd.html>
  - SGML DTD's are a formal specification of what instance documents look like.
    - Documents either do or do not conform to this specification. There is no room for ambiguity.

# Problem

- Many people wanted more from HTML
  - Mathematical formula
  - Music
  - Chemical formula
  - etc, etc...
- W3C didn't want to add everything to HTML
  - HTML would become huge!
- Instead we need a way to let people define their own mark up languages.

# SGML?

- Why not SGML?
  - Already existed.
  - Already standard.
  - Already used by HTML
- BUT...
  - SGML processing programs are large and complex.
  - W3C wanted to allow for smaller (hand held?) systems with less powerful CPUs.
  - Needed something different.

# XML

- eXtensible Markup Language
  - Essentially SGML “light”
    - Originally used the same sort of DTDs
    - Same sort of “look” in the instance documents.
    - Simplified semantics.
      - Features removed
      - Other features given fewer options.
    - Easier to process.
  - XML is to SGML as Java is to C++.

# XHTML v1.0

- ... a reformulation of HTML 4 as an XML “application.”
  - Defined by an XML DTD instead of an SGML DTD
    - Looks very similar to the SGML DTD
  - Instance documents look very similar
    - More limitations because XML is more limited than SGML.
- Now users can define other XML markups as needed
  - Resulting markup languages are easier to handle than full SGML based languages.

# Markup Languages to Know

- Many XML markup languages exist.
  - MathML
    - Allows you to describe and display mathematical formula.
  - SVG (Scalable Vector Graphics)
    - Allows you to describe vector graphics with an XML vocabulary.
  - XSL (XML Style Language)
    - Allows you to describe transformations from one XML vocabulary to another.
  - ODF (Open Document Format)
    - Office documents used by LibreOffice.
  - Many others... define your own!

# XHTML v1.1

- Modularization!
  - Allows mixing XML vocabularies. Islands of “alien” markup can appear in an XHTML 1.1 document
  - MathML for equations
  - SVG for graphics (image files not needed!)
  - ChemML for chemical formula
  - Custom markup languages if a suitable CSS or XSLT style sheet provided
- No browsers implemented this (except for [Amaya](#))

# Elements vs Tags

- `<p level="1">This is text</p>`
  - The whole thing is called an “element.”
  - The `<p level="1">` is the “start tag.”
  - The `</p>` is the “end tag.”
  - The `level="1"` is an attribute (with its value).
- Most of the time you talk about “elements.”
  - Many people call them “tags” incorrectly.
    - It makes XML specialists cringe.

# More Terminology

- Consider the following
  - `<book>`
    - `<title>XML is cool</title>`
    - `<author>Jill`
      - `<bold>Jones</bold></author>`
  - `</book>`
    - The `title` element has text-only content.
    - The `book` element has element-only content.
    - The `author` element has mixed content.
- In general white space in text-only or mixed content is *significant*.
  - Each XML application decides what to do with it.

# XML Restrictions

- As compared to SGML
  - All elements *must* have an end tag.
    - `<p>This is text.`
      - The above is wrong: there is no `</p>` tag.
      - SGML markups can define elements with optional end tags.
      - The above is okay in HTML4 because `</p>` is optional there.
  - Element names are case sensitive.
    - `<p>This is text.</P>`
      - The above is also wrong: the name used in the end tag does not match that used in the start tag.
      - SGML markups can define case insensitive elements. HTML4 does this.

# More XML Restrictions

- As compared with SGML
  - Attribute values *must* be quoted.
    - `<table border=1> ... </table>`
      - The above is wrong because the attribute value “1” is not quoted.
      - SGML allows unquoted attributes if they are not ambiguous.
  - Attribute values *must* be present.
    - `<table border> ... </table>`
      - The above is wrong because the attribute border does not have a value given.
      - SGML allows this syntax in certain cases.
- Restrictions make processing easier.

# Minimization End Tags

- Always requiring an end tag is a burden.
  - `<br></br>`
    - HTML defines a br element for break. In XHTML the end tag *must* be present (in HTML it is optional).
  - `<br/>`
    - XML defines the above minimization form that combines both tags in one.
    - Strictly this is an XML feature and thus not allowed in HTML4 (SGML) documents.
    - Web browsers are forgiving about this.
- These restrictions and features apply to *all* XML markup languages.

# “Well Formed” vs “Valid”

- Well Formed...
  - An XML document is *well formed* if it obeys the syntax requirements of XML.
- Valid...
  - An XML document is *valid* if it is well formed and if it obeys the requirements of a particular *schema*.
    - A schema defines what elements are allowed, their allowed attributes, and the relationships between elements. DTDs are a kind of schema.
- Which do we want?
  - It depends on the application.

# Character Sets

- XML is a modern markup language.
  - Unicode is the default character set.
  - UTF-8 encoding is the default document encoding.
- Implications...
  - XML documents might contain “arbitrary” binary data (strictly speaking they are not ASCII files).
    - If the document is in Chinese it could be loaded with non-ASCII bytes.
  - Should be handled as such.
    - application/xml is the MIME type used, not text/xml.

# Entities

- Certain characters can't be represented directly.
  - A literal `<` character
    - Must be represented as `&lt;`;
      - This is called a “character entity.” Note the semicolon at the end.
  - A literal `&` character
    - Must be represented as `&amp;`;
  - Can also encode arbitrary characters this way.
    - Character with Unicode code point U+ABCD can be represented as `&#xABCD;`
    - You could also just store the Unicode character directly in the file!
- XML DTDs can define other entities.

# What Makes XML Cool?

- Generic way to described structured data.
  - Many documents contain structure. Plain text obscures this structure.
  - *XML allows you to expose the structure so that programs can “see” and manipulate it.*
- Good for...
  - Managing and processing complex documents
  - Storing and sharing data.
  - Dealing with structured information in general.

# XHTML?

- Originally XML was designed to replace HTML via XHTML 1.0, 1.1, 2.0, etc.
- Capitalize on XML technologies during web development:
  - XQuery, XSL, XInclude, XPath, Modularization, etc.
- For various reasons, political, technical, and inertial, this idea did not materialize
  - XHTML 2.0 was abandoned before being finalized

# What of XML?

- XHTML never got traction, but XML is useful:
- Database People
  - Structured markup provides a way of exchanging data reliably.
    - No loss of information
    - Potentially fully typed
    - Standardized reading/writing independent of databases
- Networking People
  - A standard format for exchanging information between potentially very different systems.
    - Essentially a very high level protocol

# What of XML?

- Programmers
  - Useful for configuration files (although JSON has become more popular).
    - Maven
    - Ant
    - Hadoop
- Publishers
  - A machine manipulable format for text prior to publication
    - DocBook

# HTML5

- Instead of XHTML, the web embraced HTML5
  - More compatible with existing sites using HTML 4
  - Easier (than with XHTML) to do simple things
    - The transformational styling language, XSL, is a Turing-complete functional language!
  - More flexible error handling
    - XHTML required renderers to completely fail on error. This avoids the problem of “tag soup” that proliferated in the HTML 4 days
    - HTML5 prescribes error handling. Unusual for a standard.

# HTML5: From Whence Did It Come?

- HTML5 is based on:
  - HTML 4, the existing HTML standard at the time
  - XHTML 2.0, the next XHTML that was never finished
  - Additional features and ideas unique to HTML5
- Adds new capabilities relative to HTML 4
- Doesn't tap into the XML eco-system as XHTML does/did
- More flexible with prescribed error handling requirements

# SGML? XML?

- HTML 4 document type declaration

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- XHTML1.1 document type declaration

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

# HTML5 DOCTYPE

- HTML5 is based on neither SGML nor XML

```
<!DOCTYPE html>
```

- This is an invalid DOCTYPE declaration for HTML 4 or XHTML
  - Signals to browsers that HTML5 is being used
  - Similarity to the SGML-based or XML-based DOCTYPE declarations of HTML 4 and XHTML is “coincidence.”

# The Future

- HTML5 + CSS3 is the current standard on the web
- XHTML and associated XML technologies may vanish
  - ... but not yet. XHTML 1.0/1.1, XSL still supported by browsers, although that may change before much longer.
- Today “HTML” is HTML5 + various additions
  - **HTML is now a “living standard”** that evolves continuously without version numbers.
  - Maintained by the Web Hypertext Application Technology Working Group (WHATWG).