XML Technology: Overview

Examples and Reference

Jon Warbrick
University of Cambridge Computing Service
jon.warbrick@ucs.cam.ac.uk

July 2005
(2nd edition)

Introduction

These notes contain copies of all the example documents and programs used during the course, together with some related reference material. They ***don't*** include all the material covered and ***don't*** form a substitute for careful note taking during the sessions.

These examples were created to demonstrate particular aspects of XML technology - I make no claim in respect of their quality nor of the appropriateness of any of these examples as a basis for any particular use. In these notes, sections of some documents are presented in bold type – this is either to identify user input, to draw attention to important sections or to identify changes from previous versions of the same document.

The course has an associated web site at

```
http://www-uxsup.csx.cam.ac.uk/~jw35/courses/xml/
```

which includes up-to-date copies of these notes and of the course slides, and machine-readable copies of all the examples.

Example documents

A simple XML document, and variations thereon, is used throughout the course.

inst.xml

The basic document.

```
<?xml version="1.0"?>
<!DOCTYPE institutions SYSTEM "inst.dtd">

<institutions>

  <institution type="acad">
    <name>Computer Laboratory</name>
    <contact method="tel">+44 1223 763500</contact>
    <contact method="email">departmental-secretary@cl.cam.ac.uk</contact>
    <website>
      <url>http://www.cl.cam.ac.uk/</url>
    </website>
  </institution>

  <institution type="acad">
    <name>Division of Anaesthesia</name>
    <contact method="tel">+44 1223 217889</contact>
    <website>
      <url>http://www.medschl.cam.ac.uk/anaesthetics/</url>
    </website>
  </institution>

  <institution type="non">
    <name>Computing Service</name>
    <contact method="tel">+44 1223 334600</contact>
    <website>
      <url>http://www.cam.ac.uk/cs/</url>
      <url>http://web-support.csx.cam.ac.uk/</url>
      <url>http://www-tus.csx.cam.ac.uk/</url>
    </website>
  </institution>

  <institution type="coll">
    <name>Pembroke College</name>
    <founded>1347</founded>
    <contact method="tel">+44 1223 338100</contact>
    <contact method="email" purpose="Admissions enquiries">
      admissions@pem.cam.ac.uk
    </contact>
    <contact method="email">enquiries@pem.cam.ac.uk</contact>
    <website>
      <url>http://www.pem.cam.ac.uk/</url>
    </website>
  </institution>

</institutions>
```

inst.dtd

A DTD for the example document.

```
<!ELEMENT institutions (institution+)>

<!ELEMENT institution  (name,founded?,contact*,website?)>
<!ATTLIST institution  type (acad|non|coll) #REQUIRED>

<!ELEMENT name         (#PCDATA)>

<!ELEMENT founded      (#PCDATA)>

<!ELEMENT contact      (#PCDATA)>
<!ATTLIST contact      type    (tel|email|fax) #REQUIRED
                       purpose CDATA   #IMPLIED>

<!ELEMENT website      (url+)>

<!ELEMENT url          (#PCDATA)>
```

Example XSLT style sheets

example1.xslt

The simplest possible XSLT styles sheet.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

</xsl:stylesheet>
```

example2.xslt

A (largely useless) style sheet to output 'An institution' for each `<institution>` element.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="institution">An institution</xsl:template>

</xsl:stylesheet>
```

The result of applying `example2.xslt` to `inst.xml`:

```
$ xsltproc example2.xslt inst.xml
<?xml version="1.0"?>


  An institution

  An institution

  An institution

  An institution
```

example3.xslt

Output 'An institution' for each `<institution>` element and surround it with new tags.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="institution">
    <heading>An institution</heading>
  </xsl:template>

</xsl:stylesheet>
```

example4.xslt

Output the value of each `<institution>` element's `<name>` element.

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="institution">
    <heading>
      <xsl:value-of select="name"/>
    </heading>
  </xsl:template>

</xsl:stylesheet>
```

The result of applying `example4.xslt` to `inst.xml`:

```
$ xsltproc example4.xslt inst.xml
<?xml version="1.0"?>


  <heading>Computer Laboratory</heading>

  <heading>Division of Anaesthesia</heading>

  <heading>Computing Service</heading>

  <heading>Pembroke College</heading>
```

example5.xslt

This example shows how the processing order can be controlled using `<xslt:apply-templates>`

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

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="institutions">
    <heading>Here are a list of website URLs</heading>
    <xsl:apply-templates select="institution"/>
    <footing>Information provided by webmaster</footing>
  </xsl:template>

  <xsl:template match="institution">
    <xsl:apply-templates select="website"/>
  </xsl:template>

  <xsl:template match="website">
    <site>
      <xsl:value-of select="url"/>
    </site>
  </xsl:template>

</xsl:stylesheet>
```

The result of applying `example5.xslt` to `inst.xml`:

```
$ xsltproc example5.xslt inst.xml
<?xml version="1.0"?>
<heading>Here are a list of website URLs</heading><site>http://www.cl.cam
.ac.uk/</site><site>http://www.medschl.cam.ac.uk/anaesthetics/</site><sit
e>http://www.cam.ac.uk/cs/</site><site>http://www.pem.cam.ac.uk/</site><f
ooting>Information provided by webmaster</footing>
```

Programmatic XML processing

dom.pl

A simple Perl program using a DOM (Document Object Model)

```
#!/usr/bin/perl -w

use strict;

use XML::LibXML;

my $fn = $ARGV[0] or die "Usage: $0 <filename>\n";
my $parser = XML::LibXML->new();
my $doc = $parser->parse_file($fn);

my $root = $doc->getDocumentElement;

do_process($root,0);

# --------------------------------------------------------------------

sub do_process {

  my ($element,$indent) = @_;

  # Print out text values from text nodes

  if (ref($element) =~ /Text/) {
    my $text = normalise($element->nodeValue);
    if ($text !~ /^\s*$/) {
      print ' ' x $indent;
      print '"', $text, '"', "\n"
    }
  }

  # Otherwise print the element name

  else {
    print ' ' x $indent;
    print $element->nodeName, "\n";
  }

  # Process any children

  foreach my $node ($element->getChildnodes) {
    do_process($node,$indent+2);
  }

}

# --------------------------------------------------------------------

sub normalise {

  my ($text) = @_;

  $text =~ s/[\s\n]+/ /g;
  $text =~ s/^\s*//;
  $text =~ s/\s*$//;

  return $text;

}
# --------------------------------------------------------------------
```

The result of applying `dom.pl` to `inst.xml`:

```
$ ./dom.pl inst.xml
institutions
  institution
    name
      "Computer Laboratory"
    contact
      "+44 1223 763500"
    contact
      "departmental-secretary@cl.cam.ac.uk"
    website
      url
        "http://www.cl.cam.ac.uk/"
  institution
    name
      "Division of Anaesthesia"
    contact
      "+44 1223 217889"
    website
      url
        "http://www.medschl.cam.ac.uk/anaesthetics/"
...more...
```

sax.pl

Much the same functionality as `dom.pl`, but this time using a SAX (Simple API for XML):

```
#!/usr/bin/perl -w

use strict;

use XML::SAX::ParserFactory;

my $fn = $ARGV[0] or die "Usage: $0 <filename>\n";

my $handler = My::Handler->new();
my $parser = XML::SAX::ParserFactory->parser( Handler => $handler );
$parser->parse_uri($fn);

# ------------------------------------------------------------------

package My::Handler;

sub new {
  my $class = shift;
  my $self = {@_};
  $self->{indent} = 0;
  return bless($self,$class);
}

sub start_element {
  my $self = shift;
  my $data =  shift;
  print ' ' x $self->{indent};
  print $data->{Name}, "\n";
  $self->{indent} += 2;
}

sub end_element {
  my $self = shift;
  my $data =  shift;
  $self->{indent} -= 2;
}

sub characters {
  my $self = shift;
  my $data =  shift;
  my $text = normalise($data->{Data});
  if ($text !~ /^\s*$/) {
    print ' ' x $self->{indent};
    print '"', $text, '"', "\n"
  }
}

# ------------------------------------------------------------------

sub normalise {

  my ($text) = @_;

  $text =~ s/[\s\n]+/ /g;
  $text =~ s/^\s*//;
  $text =~ s/\s*$//;

  return $text;

}

# ------------------------------------------------------------------
```

example6.xslt:

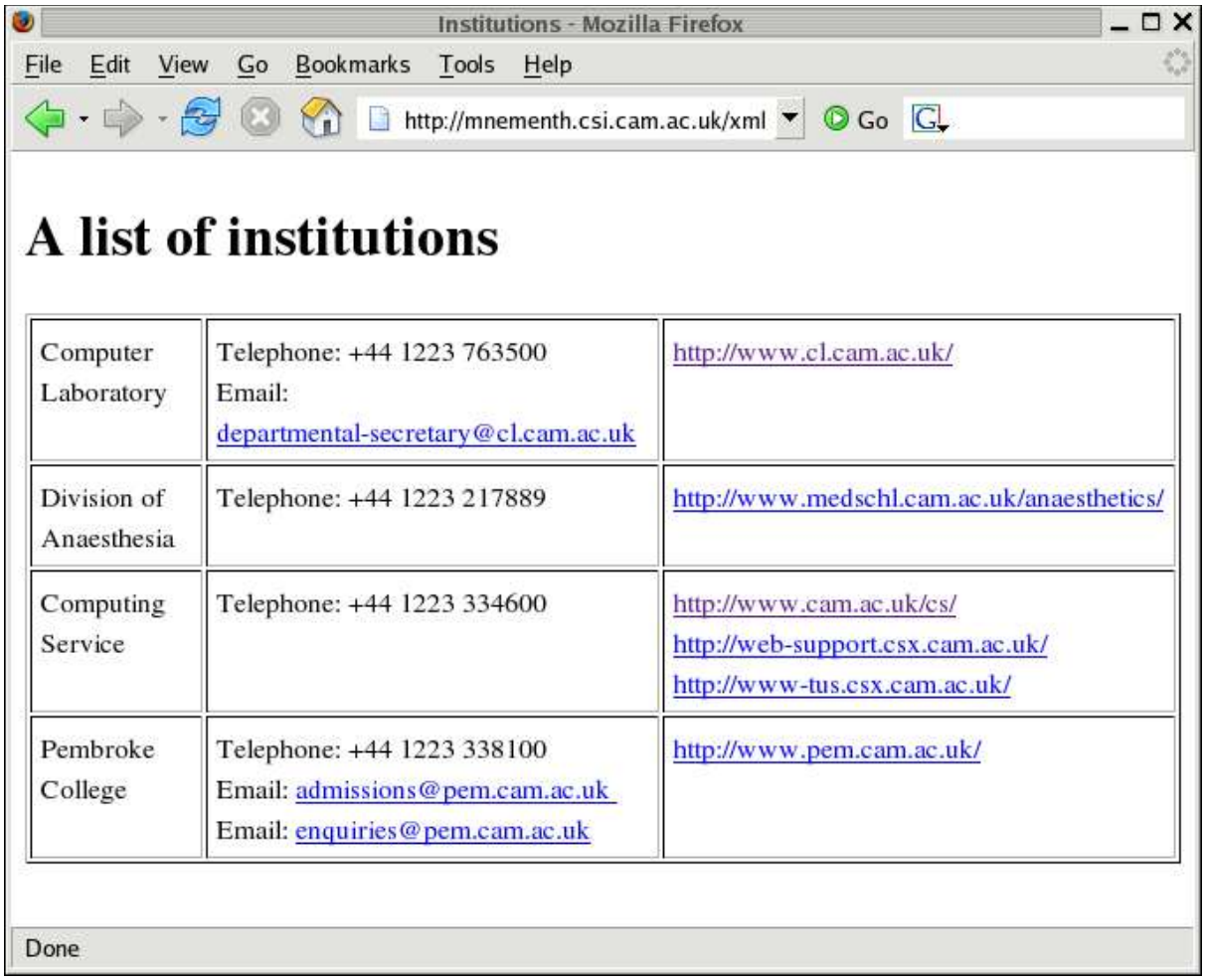
An XSLT style sheet to transform inst.xml into XHTML:

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

<xsl:stylesheet
  version="1.0"
  xmlns="http://www.w3.org/1999/xhtml"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output
    doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
    doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd"
    method="xml"
    omit-xml-declaration="yes"
    indent="yes" />

  <xsl:template match="institutions">
    <html lang="en">
      <head>
        <title>Institutions</title>
      </head>
      <body>
      <h1>
        A list of institutions
      </h1>
      <table border="1" cellpadding="5">
        <xsl:apply-templates select="institution"/>
      </table>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="institution">
    <tr valign="top">
      <td>
        <xsl:apply-templates select="name"/>
      </td>
      <td>
        <xsl:apply-templates select="contact"/>
      </td>
      <td>
        <xsl:apply-templates select="website/url"/>
      </td>
    </tr>
  </xsl:template>

  <xsl:template match="contact">
    <xsl:choose>
      <xsl:when test="@method='tel'">
        Telephone:
        <xsl:value-of select="."/>
      </xsl:when>
      <xsl:when test="@method='email'">
        Email:
        <a>
          <xsl:attribute name="href">
            mailto:<xsl:value-of select="."/>
          </xsl:attribute>
          <xsl:value-of select="."/>
        </a>
      </xsl:when>
```

```
      <xsl:otherwise>
        <xsl:value-of select="."/>
      </xsl:otherwise>
    </xsl:choose>
    <br/>
  </xsl:template>

  <xsl:template match="url">
    <a>
      <xsl:attribute name="href">
        <xsl:value-of select="."/>
      </xsl:attribute>
      <xsl:value-of select="."/>
    </a>
    <br/>
  </xsl:template>

</xsl:stylesheet>
```

With a minor addition to inst.xml, this transformation can even happen within a browser:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xml" href="example6.xslt"?>
<!DOCTYPE institutions SYSTEM "inst.dtd">

<institutions>

  <institution type="acad">

...etc...
```



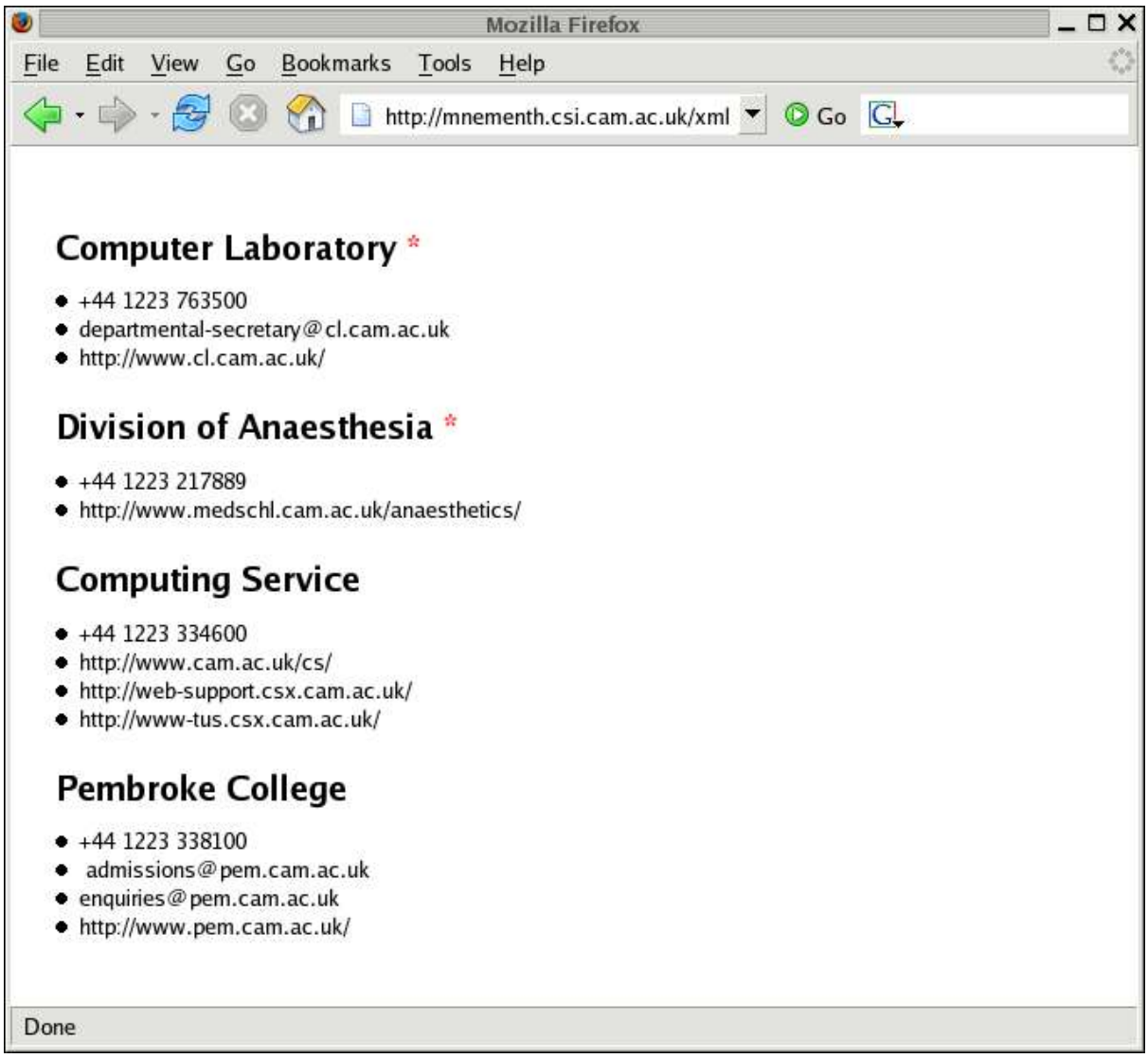
Formatting XML with CSS

An example CSS style sheet for formatting inst.xml:

```
institutions {
  font-family: sans-serif;
  font-size: 10pt;
  margin: 20pt;
}

name {
  display: block;
  font-size: 15pt;
  font-weight: bold;
  margin-top: 15pt;
  margin-bottom: 8pt;
}

institution[type="acad"] name:after {
  color: #f33;
  content: " *";
}

contact, url {
  display: list-item;
  list-style-position: inside;
}

founded {
  display: none;
}
```

With a minor addition to inst.xml, this styling can be applied by a browser:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="example1.css"?>
<!DOCTYPE institutions SYSTEM "inst.dtd">

<institutions>

  <institution type="acad">

...etc...
```


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://mnementh.csi.cam.ac.uk/xml
Go
Computer Laboratory *
+44 1223 763500
departmental-secretary@cl.cam.ac.uk
http://www.cl.cam.ac.uk/
Division of Anaesthesia *
+44 1223 217889
http://www.medschl.cam.ac.uk/anaesthetics/
Computing Service
+44 1223 334600
http://www.cam.ac.uk/cs/
http://web-support.csx.cam.ac.uk/
http://www-tus.csx.cam.ac.uk/
Pembroke College
+44 1223 338100
admissions@pem.cam.ac.uk
enquiries@pem.cam.ac.uk
http://www.pem.cam.ac.uk/
Done

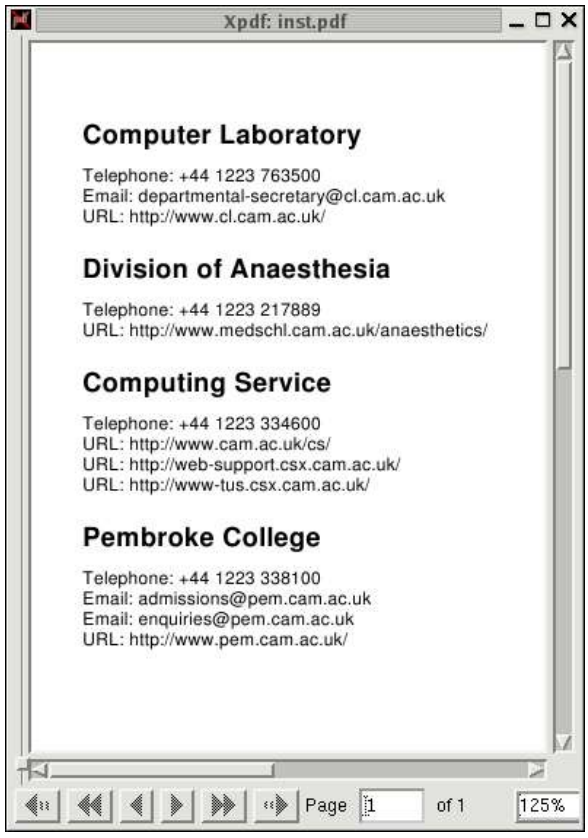
example7.xslt:

An XSLT style sheet to transform inst.xml into XSL-FO:

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

<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">

  <xsl:template match="institutions">
    <fo:root>
      <fo:layout-master-set>
        <fo:simple-page-master master-name="first"
          margin-right="10mm"  margin-left="10mm"
          margin-top="10mm"    margin-bottom="10mm"
          page-width="210mm"   page-height="297mm">
          <fo:region-body/>
        </fo:simple-page-master>
      </fo:layout-master-set>

      <fo:page-sequence master-reference="first">
        <fo:flow flow-name="xsl-region-body">
          <xsl:apply-templates/>
        </fo:flow>
      </fo:page-sequence>

    </fo:root>
  </xsl:template>

  <xsl:template match="institution">
    <fo:block font-family="sans-serif" font-size="10pt">
      <xsl:apply-templates select="name"/>
      <xsl:apply-templates select="contact"/>
      <xsl:apply-templates select="website/url"/>
    </fo:block>
  </xsl:template>

  <xsl:template match="name">
    <fo:block font-size="15pt" font-weight="bold"
      space-before="15pt" space-after="8pt">
      <xsl:value-of select="."/>
    </fo:block>
  </xsl:template>

  <xsl:template match="contact">
    <fo:block>
      <xsl:choose>
        <xsl:when test="@method='tel'">
          Telephone:
          <xsl:value-of select="."/>
        </xsl:when>
        <xsl:when test="@method='email'">
          Email:
          <xsl:value-of select="."/>
        </xsl:when>
      </xsl:choose>
    </fo:block>
  </xsl:template>

  <xsl:template match="url">
    <fo:block>
      URL:
```

```
      <xsl:value-of select="."/>
    </fo:block>
  </xsl:template>

</xsl:stylesheet>
```

The resulting XSL-FO can be converted to PDF and displayed:



References

Standards, Recommendations, Guielines

Covered in the course:

- XML: http://www.w3.org/XML/
    - 1.0 (Feb 1998): http://www.w3.org/TR/2004/REC-xml-20040204/
    - 1.1 (Fen 2004): http://www.w3.org/TR/2004/REC-xml11-20040204/
- Namespaces in XML:
    - (Jan 1999): http://www.w3.org/TR/1999/REC-xml-names-19990114/
- XSL and XSLT: http://www.w3.org/Style/XSL/1.0
    - XSLT 1.0 (Nov 1999): http://www.w3.org/TR/1999/REC-xslt-19991116
    - XPath 1.0 (Nov 1999): http://www.w3.org/TR/1999/REC-xpath-19991116
    - XSL-FO: 1.0 (Oct 2001) http://www.w3.org/TR/2001/REC-xsl-20011015/
- DOM: http://www.w3.org/DOM/
    - Level1/2/3 recommendations: http://www.w3.org/DOM/
- SAX: http://sax.sourceforge.net/

Otherwise mentioned

- XForms: http://www.w3.org/MarkUp/Forms/
- XLink: http://www.w3.org/XML/Linking
- XML Schema: http://www.w3.org/XML/Schema
- Xinclude:
    - 1.0 (Dec 2004): http://www.w3.org/TR/2004/REC-xinclude-20041220/

Example applications

- TEI (Text Encoding Initiative) : http://www.tei-c.org/
- DocBook: http://www.oasis-open.org/docbook/
- OpenOffice: http://xml.openoffice.org/
- XHTML: http://www.w3.org/MarkUp/
    - 1.1 (May 2001): http://www.w3.org/TR/2001/REC-xhtml11-20010531/
- SVG (Scalable Vector Graphics): http://www.w3.org/Graphics/SVG/
- Jabber: http://www.jabber.org/
- RSS: see http://www.xml.com/pub/a/2002/12/18/dive-into-xml.html
- SOAP: http://www.w3.org/2000/xp/Group/
- XML-RPC: http://www.xmlrpc.com/

Books

- XML for the World Wide Web (Visual Quickstart Guide). *Elizabeth Castro*. Peachpit Press, 2001. 0-201-71098-6

Also the following from O'Reilly (http://xml.oreilly.com/):

- XML in a Nutshell (third edition). *Elliotte Rusty Harold, W. Scott Means*. O'Reilly, 2004. ISBN: 0-596-00764-7
- Learning XML (second edition). *Erik T. Ray.* O'Reilly 2003. ISBN: 0-596-00420-6
- Perl and XML. Erik T. Ray, *Jason McIntosh*. O'Reilly, 2002. ISBN: 0-596-00205-X
- XSLT. *Doug Tidwell*, O'Reilly, 2001. ISBN: 0-596-00053-7
- XSL-FO Making XML Look Good in Print. *Dave Pawson.* O'Reilly 2002. ISBN: 0-596-00355-2
- Programming Jabber - Extending XML Messaging. *DJ Adams*. 2002. ISBN: 0-596-00202-5

... and many others from the same source.

Other sources of reference

- O'Reilly's XML.COM: http://www.xml.com/ for articles and tutorials
- Cover Pages: http://xml.coverpages.org/

Software

XML parsers/validators

- xmllint – part of Gnome libxml - http://xmlsoft.org/
- Xerces – from the Apache Software Foundation - http://xml.apache.org/
- onsgmls – part of the OpenJade package - http://openjade.sourceforge.net/
- http://www.stg.brown.edu/service/xmlvalid/

XSLT processors

- xsltproc – part of Gnome libxslt - http://xmlsoft.org/XSLT/
- Xalan - from the Apache Software Foundation - http://xml.apache.org/
- SAXON - http://saxon.sourceforge.net/