MODERN METHODS OF LEARNING TEXT PUBLICATION

Jiří Týř
Technical University of Liberec
Czech Republic

Abstract

Web-based learning is an important part of the e-learning ecosystem. Using the Internet as a medium for technical learning text publications can bring two major problems: the problem of learning text re-usability and the problem of interpreting a mathematical notation. Thanks to modern web technologies it is possible to solve both problems and suppress the difference with the paper-based learning text publications. This paper provides an overview of possible solutions emphasizing one particular implementation of technical learning text publication implemented in the TUL University E-learning System environment.

Introduction

Thanks to the expansion of the Internet in the last decade, the term "e-learning" is used more for web-based type learning than for any other means. More than ever before, it is important to have a possibility to share the source of the learning text between the web presentation and the printed form of textbooks. Both media are different and therefore they need a different approach and set of tools. This is especially true for the case of a technical learning text with mathematical notation.

Web-based Presentation

Learning text within an e-learning environment can be represented by a text document or even better by an HTML (W3C, 2008a) page. Obviously an HTML page can provide more interactive content than the standard text document. An HTML page can contain not only static texts with images but also dynamically generate texts with animated images, audio and video content, Java applets and other multimedia content. Jan Amos Komenský (1630), known as the teacher of nations, had a saying: "Learning by play." Exactly that should be the meaning and the purpose of the learning text represented by the interactive multimedia HTML page.

The World Wide Web (W3) was born at CERN (CERN, 2008) in 1989. There is no doubt that it was one of the most important inventions of last century. Since that time, there have been developed a large number of tools which can help produce an HTML page. Each of the HTML editors can be more or less user friendly with different levels of conformity and support of world-wide respected norms of web development. Many e-learning systems, which use HTML for learning text presentation, have usually some kind of HTML editor as a part of the

system. These systems have one important advantage. For editing of the HTML code it is not needed to have any other software than a web browser which is a standard part of each modern operating system today.

Printable Output

Development of interactive multimedia HTML pages can take a lot of effort. Printing of the learning text directly from the HTML page is not so simple because the printed document may not have the same layout and quality as the original. This can obviously lead to many problems during study of this printed material. Therefore many learning texts are available as static documents containing only text content. These documents are then exposed on the web page where they are available for download. The user can download this file and open it in a specialized viewer or editor. The advantage of having the document written in a specific format is that it can always be displayed or printed out in the same layout and quality.

Probably the best known printable document format these days is the Portable Document Format (PDF) created by Adobe Systems in 1983 for document exchange. This format is as known in the field of text documents as Google is in Internet search engines. This success is based on its open format, independent of hardware type, operating system and application software. PDF is supported by Microsoft Office, OpenOffice.org and many other Office suites which allow users to save their documents directly into the PDF format. This provides an easy way of text document publication in very good quality. The high quality of the printed output is reached mainly because each document encapsulates a description of the text, fonts and vector graphics that compose the documents.

Mathematical Motation

E-learning is used in many branches today — from psychology and language education through economics and computer science up to theoretical mathematics and physics. If the learning text tries to explain some parts of mathematics, it is usually necessary to use mathematical notation. The traditional tool for writing mathematical documents has always been LaTeX (LaTeX, 2008). The strong points of this typesetting system are the extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies. It supports also mathematics notation which allows description of any kind of mathematical expression in a very simple way. Because its output is PostScript (PS) or PDF, it is a perfect tool for creation of documents for printable output.

Implementation of LaTeX for the web is not directly possible, because it was not developed for it. There exists a project (LaTeX2HTML, 2001) which allows production of an HTML page from LaTeX but its implementation into the

e-learning environment is the least very complicated. Fortunately there exists Mathematical Markup Language (MathML) (W3C, 2008b) which is designed especially for integration of mathematical formulas into the web documents and becomes as a standard for mathematics on the web.

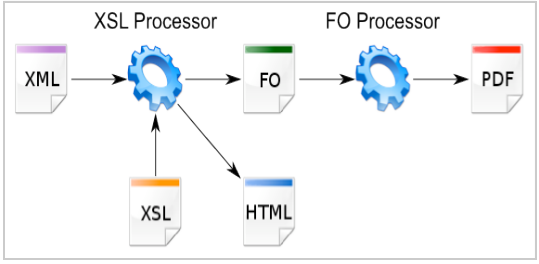
General Solution — DocBook

Once the learning text has been written as an HTML page, it is very difficult to transform this text into a printable document or share it without any modification with another e-learning system. A better solution is to have the learning text in such a form that would allow transforming it into the printable document, HTML or any other format. A format which allows this already exists and it is called DocBook (OASIS, 2008a).

DocBook is a semantic markup language for technical documentation which was originally intended for writing technical documents related to computer hardware and software. It also can be used for any other sort of documentation including learning texts. As a semantic language, DocBook enables the creation of document content in a presentation-neutral form that captures the logical structure of the content. That content can then be published in a variety of formats, including HTML, PDF and Rich Text Format (RTF), without the need to modify the source.

DocBook is actually just a XML language described by a set of schemes (OASIS, 2003; W3C, 2008c; W3C, 2007) which are defining the structure of the document. DocBook document do not describe how its content look like, but it describe the meaning of the content. That is why for a more presentational format of a DocBook document it is necessary to use the DocBook XSL Stylesheets (OASIS, 2008b). These are XSLT (W3C, 2008d) stylesheets that transform DocBook documents into a number of formats. The principle of XSL transformation is shown in Figure 1. These stylesheets are very adaptable and intelligent. By setting several parameters (Stayton, 2007) it generates automatically the title page, chapter and figure numbering, tables of contents and any other parts that are extracted from the content and structure of the document.

Figure 1: DocBook Publishing Model



Since DocBook is a semantic language designed not only for a description of the information but also for easy use, the creation of the document is intuitive and fast. Anyone who looks for the first time at a DocBook document immediately understands what that part of the document expresses. A simple document is visible in Example 1, where the basic structure of a DocBook document is shown.

Example 1: Simple DocBook Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
"http://www.oasis-open.org/docbook/xml/4.4/docbookx.dtd">
<book>
  <bookinfo>
    <title>My first book</title>
    <author>
      <firstname>John</firstname>
      <surname>Doe</surname>
    </author>
  </bookinfo>
  <chapter>
    <title>First chapter</title>
    <para>Some text of the first chapter.</para>
  </chapter>
  <chapter>
    <title>Second chapter</title>
    <para>Some text of the <emphasis>second</emphasis> chapter.</para>
    <section>
      <title>First section</title>
      <para>Some text of the section.</para>
    </section>
  </chapter>
</book>
```

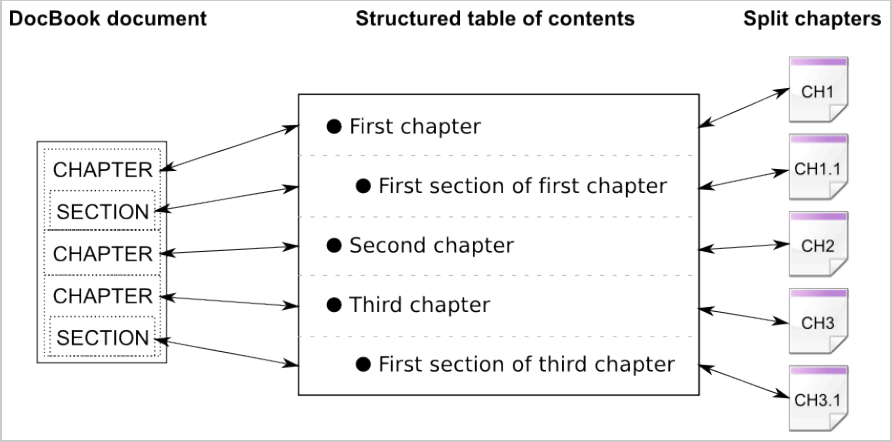
DocBook Implementation

Use of XML-based authoring methodology in an e-learning environment is not uncommon as discussed by Ger et al., 2006. At the Technical university of Liberec, DocBook has been implemented into the University E-learning System (UES). UES makes it possible to publish a learning text in a classical HTML format as well as in a DocBook format. In the case of DocBook use, it is possible to choose between several document types: book, chapter, article and reference page. In the third line of Example 1, each type is specified as a DOCTYPE definition.

Because the learning text in an e-learning environment is often used as a complete text or as a single chapter, the most suitable document type is book and chapter. To keep the data integrity and to improve the search performance of the e-learning system it is better to have all the learning texts stored in a database management system (DBMS). Because the learning text is browsed by chapters, they all have to be stored in the DBMS separately. This should be taken into account during the database modeling in order to be able to restore the original structure of the learning text from the various chapters.

Adding a new learning text into the UES takes place by the creation of a structured table of contents. If the learning text is split into chapters, each chapter actually represents a new learning text which can be displayed separately. The newly created structured document can then be called a theme which will group all the parts of the learning text together.

Figure 2: Theme Structure



Each item of the themes represents a chapter or a section of a DocBook document. Each chapter of the theme is inserted as a text of the specific item of the table of contents of the theme. This situation is visible in Figure 2 (from left to right). The initial creation of the theme structure can be automatized by importing of the whole learning text as a DocBook document into the e-learning system. Updates can be done by editing the source code of the chapter directly in the internal UES editor or in another editor having access to the DBMS.

Web presentation. Each chapter in the UES is represented by a piece of DocBook document which belongs to a particular theme. For visualization of the DocBook code, it is necessary to use DocBook XSL Stylesheets to create the HTML by using an XSL transformation (XSLT). To avoid doing the XSLT every time the HTML page is displayed, the result of the XSLT can be saved into the database. Reading of the pre-processed HTML code from the database is much faster than the process of the XSLT and it has significant influence on the system response time. For easier implementation of the theme structure, the name of the chapter is stored in the database separately from the rest of the chapter. This allows the user to write only the inner part of the chapter which consists of a set of paragraphs or similar elements.

Because each chapter in the system is not fully defined, it is necessary to create the full DocBook document structure before the XSLT. This means creating the chapter document type definition, creating the body of the chapter including its title, and inserting below the title the part of the chapter created by the user. Only documents following this procedure can be transformed into the HTML output.

DocBook XSL Stylesheets produce by default a complete HTML page output. Because the UES itself is displayed in the web browser as an HTML page and because the output of the XSLT is also an HTML page, it is bad practice to insert the output of the XSLT as it is into the page. Therefore it is necessary to chop off the parts of the XSLT output, which are not a content of the body of the HTML document. This preempts a collision with the HTML definition of the UES page. It is only this modified output of the XSLT that can be saved into the database and later shown as a part of the UES output.

Printable output. Since the web presentation and the printable output share the same learning text source, one has to create the DocBook document first. In the environment of the UES it is possible to create printable output of each chapter separately or group all the chapters into one complete document and create a book. Compared to the creation of the web presentation, the XSLT output for printable output is XSL Formatting Objects (FO) which are the source for the FO Processor that produces a PDF file. In the UES the PDF file is accessible on the same page as the web presentation of the chapter.

The process of book creation is more difficult. Because the book consists of all the separate chapters which exist in the theme, it is necessary to reproduce the same structure of the future book into a single DocBook document. This structure is taken from the structured table of contents of the theme. The process flow (from right to left) is shown in Figure 2. In comparison to the creation of a single chapter DocBook document, the DocBook document of a complete book contains information which includes book title, name of authors, publisher name and copyright. From this information and from the structure and the content of the chapters a title page, table of contents, abstract with keywords, acknowledgment, preface, list of acronyms, chapters, sections of the chapters, bibliography and appendixes are automatically created. The document prepared in this way is transformed with XSLT and FO processing into the PDF file and stored in the UES. The user can click on this PDF file on the page with the structured table of contents of the theme.

DocBook and Mathematical Notation

Mathematical equations can be described in the DocBook format by three environments. These are ‘Equation’, ‘InformalEquation’ and ‘InlineEquation’. The ‘Equation’ environment is used for referenced equations, as shown in Example 2.

A suitable environment for non-referenced equations is ‘InformalEquation’. This two mentioned environments do not provide a facility to construct in-line mathematical formulas but these can be inserted into the document by using the 'InlineEquation' environment.

Example 2: MathML Equation in DocBook

```
<para>
  <equation>
    <title>First MathML equation</title>
    <math>
      <mrow>
        <msup>
          <mi>c</mi>
          <mn>2</mn>
        </msup>
        <mo>=</mo>
        <msup>
          <mi>a</mi>
          <mn>2</mn>
        </msup>
        <mo>+</mo>
        <msup>
          <mi>b</mi>
          <mn>2</mn>
        </msup>
      </mrow>
    </math>
  </equation>
</para>
```

Example 3: LaTeX Equation in DocBook

```
<para>
  <equation>
    <title>First MathML equation</title>
    <math role=”latex”>$$c^2 = a^2 + b^2$$</math>
  </equation>
</para>
```

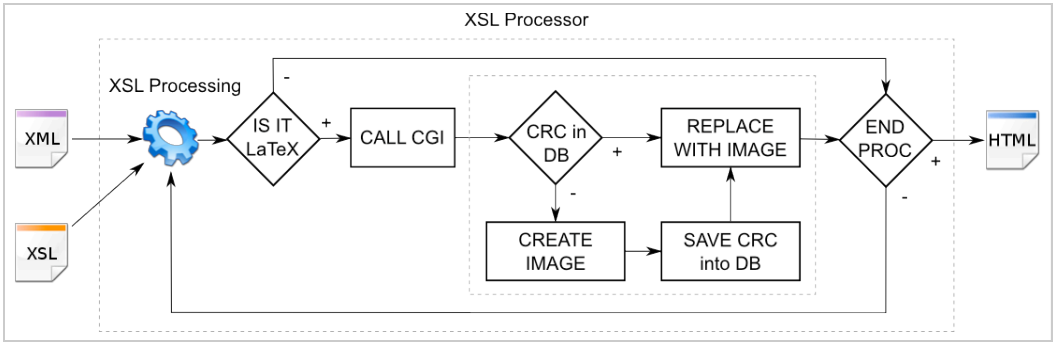
All of the environments can contain a definition of mathematical formulas. In principle, there are two ways of writing formulas. The first one is based on the pure XML language MathML. The advantage of MathML is that it is still XML so it is possible to process it like the rest of the DocBook document. The disadvantage is that the XML description of any mathematic formula is too descriptive and can take a lot of space. Also for a human it is harder to create and to read. The second possibility is to write the mathematical formula in old-fashioned LaTeX form. The advantage of this solution is that this form is very compact and transparent. The disadvantage is that the mathematical formula is not XML anymore and it has to be processed as text. Both forms of the same equation are shown in Examples 2 and 3.

The use of LaTeX mathematical notation in DocBook has other reasons. The first reason is more or less historical, because there were not many possibilities for writing a accurate mathematical formula in the context of a typesetting system other than LaTeX. The second reason is that people know LaTeX and they want to use it even in the XML context. Good examples of this approach are the existence of several projects that are trying to convert XML documents into LaTeX (DB2LaTeX, 2004), (PassiveTeX, 2005) or vice versa (TeXt4ht, 2007).

Web presentation. Even if MathML is specially designed for implementing web documents, its implementation faces several problems. The first problem is the lack of direct support in all web browsers. MathML is only supported by web browsers based on the Gecko engine or by the newest version of the Opera web browser. In other browsers it is usually possible only with a particular plug-in (Design Science, 2008). The second problem is the need to have particular fonts installed which allow the display of special mathematical symbols. A better solution is the possibility to transform the MathML into Scalable Vector Graphics (SVG) which have much a wider support in web browsers. This situation has been described by Goossens (2002) in the project "Tools for Innovative Publishing in Science." Even if this review is old, not much has changed since.

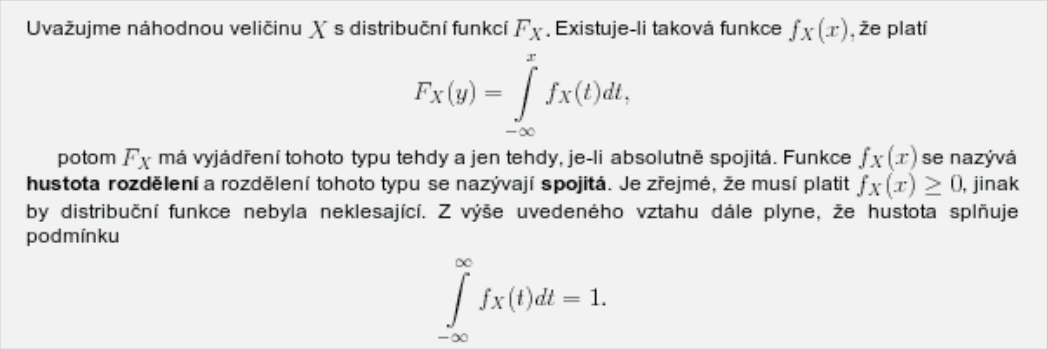
Due to the poor MathML support in the web browsers, the only viable way for many users is the use of images. This is the only universal solution for all web browsers today. On the other hand, this has also some limitations that can bring some problems. First of all, it is very difficult to align correctly an equation to the baseline of the surrounding text. The second problem is that it is just a picture, which has to be generated somehow and stored somewhere. In the environment of the UES a procedure is used that fixes at least the second problem. This procedure is shown in Figure 3.

Figure 3: XSL Transformation into HTML with LaTeX Mathematical Formulas



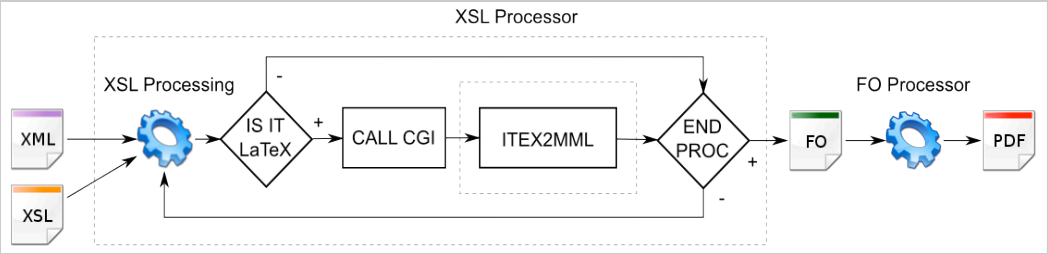
Every part of the DocBook document is processed by an XSL Processor. When a LaTeX mathematical environment is detected, the mathematical notation is encoded by a Base64 algorithm and sent to the CGI script. This script decodes the text and checks if the same mathematical notation was already processed. If not, it saves the text into a LaTeX file. This file is processed by LaTeX to create a DVI file. From the DVI file a PS file is created which is then processed by GhostScript (GhostScript, 2008) into the transparent PNG image. Because the generation of the image is time consuming, the image is saved on the disk. On top of that, the Cyclic Redundancy Check (CRC) of the decoded text is saved into the database for later use. After that, the original definition of the mathematical environment is replaced by a corresponding graphical environment that includes the link to the PNG file. All the mathematical environments in the document are processed in the same way. The result of the XSLT is an HTML page with pictures in the place of the definition of the original mathematical notation as shown in Figure 4.

Figure 4: Example of HTML Output of the Text Containing Mathematical Formulas



Printable output. As shown in Figure 5, the creation of a DocBook printable output with mathematical formulas uses almost the same procedure as in the case of the web presentation. Because PDF is giving the best results when using vector graphics, it is not desirable to have mathematical formulas as an image. The use of MathML is preferred because it is easy to transform into the SVG format. For users, describing mathematical formulas in LaTeX is much easier, so it is necessary to have a tool that allows the transformation from LaTeX into MathML (itex2MML, 2008; ORCCA, 2002a). This tool is called during the process of the XSLT to replace the LaTeX formula with its MathML equivalent.

Figure 5: XSL Transformation into PDF with MathML Mathematical Formulas



The output of the XSLT is an FO that will be transformed into the printable output. If the learning text contains mathematical formulas, it is necessary to use the FO Processor (Apache, 2008) with MathML support (JEuclid, 2008). An example of the PDF file including mathematical formulas is shown in Figure 6.

Figure 6: Example of PDF Output of the Text containing Mathematical Formulas

Uvažujme náhodnou veličinu X s distribuční funkcí F_X. Existuje-li taková funkce $f_X(x)$, že platí

$$F_X(y) = \int_{-\infty}^{x} f_X(t)\mathrm{d}t,$$

potom F_X má vyjádření tohoto typu tehdy a jen tehdy, je-li absolutně spojitá. Funkce $f_X(x)$ se nazývá **hustota rozdělení** a rozdělení tohoto typu se nazývají **spojitá**. Je zřejmé, že musí platit $f_X(x) \geq 0$, jinak by distribuční funkce nebyla neklesající. Z výše uvedeného vztahu dále plyne, že hustota splňuje podmínku

$$\int_{-\infty}^{\infty} f_X(t)\mathrm{d}t = 1.$$

Conclusion

As shown in this paper, XML technologies are suitable to describe information in the e-learning environment. It can be used not only as a description of the learning text presentation, but also for sharing of learning objects within another e-learning system. DocBook, as one of the XML languages, is a ready-to-use tool which allows transforming the learning text into several formats. It helps the maker of the learning text to create a single source text and transform it into the web presentation as well as into the printable output.

Although mathematical formulas are supported by web browsers, displaying them correctly can be very problematic, even if the standards exist already for some time. Therefore it is unfortunate to say that the most reliable solution is still the use of images. For printable output containing mathematical formulas, the

situation is very different because most of the documents are produced in LaTeX. This strong influence is also felt in mathematic related XML technologies where it is still possible to write mathematical formulas in the old-fashioned LaTeX form. We can only hope that in the future the support of mathematical notation on the web will grow and reach a level that can compete with the quality of LaTeX.

References

Apache Sotware Foundation, Inc. (2008). *Apache FOP.* Retrieved Jan. 10, 2009, from http://xmlgraphics.apache.org/fop/

CERN. (2008). *CERN — Where the web was born.* Retrieved Jan 10, 2009, from http://public.web.cern.ch/Public/en/About/Web-en.html

DB2LaTeX. (2004). *Welcome to the DB2LaTeX XSLT Stylesheets.* Retrieved Jan. 10, 2009, from http://db2latex.sourceforge.net/

Design Science, Inc. (2008). *MathPlayer*. Retrieved Jan. 10, 2009, from http://www.dessci.com/en/products/mathplayer/

Ger, P. M., Manjón, B. F., Ortiz, I. M., & Rodríguez, J. L. S. (2006). Using DocBook and XML technologies to create adaptive learning content in technical domains. *International Journal of Computer Science and Applications*, *3*(2), 91–108.

GhostScript. (2008). GhostScript Website. Retrieved Jan. 10, 2009, from http://ghostscript.com/

itex2MML. (2008). *MathML and Mozill.* Retrieved Jan. 10, 2009, from http://pear.math.pitt.edu/mathzilla/itex2mml.html

Goossens, M. (2002). *An XML Web strategy for scientific documents.* Retrieved Jan. 10, 2009, from http://goossens.web.cern.ch/goossens/tipsmalaga.pdf

JEuclid. (2008). *JEuclid.* Retrieved Jan. 10, 2009, from http://JEuclid.sourceforge.net/

Komenský, J. A. (1630). *Škola hrou.*

LaTeX. (2008). *LaTeX — A document preparation system.* Retrieved Jan. 10, 2009, from http://www.latex-project.org/

LaTeX2HTML. (2001). *latex2html.org.* Retrieved Jan. 10, 2009, from http://www.latex2html.org/

OASIS. (2008a). *DocBook.org*. Retrieved Jan. 10, 2009, from http://www.docbook.org/

OASIS. (2003). RELAX NG home page. Retrieved Jan. 10, 2009, from http://relaxng.org/

OASIS. (2008b). *DocBookXslStylesheets.* Retrieved Jan. 10, 2009, from http://wiki.docbook.org/topic/DocBookXslStylesheets/

ORCCA. (2002a). *TeX/LaTeX to MathML Online Translator.* Retrieved Jan. 10, 2009, from http://www.orcca.on.ca/MathML/texmml/textomml.html

PassiveTeX. (2005). *PassiveTeX.* Retrieved Jan. 10, 2009, from http://www.tei-c.org.uk/Software/passivetex/

Stayton, B. (2007). *DocBook XSL: The complete guide* (4th ed.). Santa Cruz, CA: Sagehill.

TeX4ht. (2007). *TeX4ht: LaTeX and TeX for Hypertext.* Retrieved Jan. 10, 2009, from http://www.cse.ohio-state.edu/~gurari/TeX4ht/

W3C. (2008a). *W3C XHTML2 Working Group Home Page.*Retrieved Jan. 10, 2009, from http://www.w3.org/MarkUp/

W3C. (2008b). *W3C math home.* Retrieved Jan. 10, 2009, from http://www.w3.org/Math/

W3C. (2008c). *XML schema.* Retrieved Jan. 10, 2009, from http://www.w3.org/XML/Schema

W3C. (2007). *Recommended DTDs to use in your web document.* Retrieved Jan. 10, 2009, from http://www.w3.org/QA/2002/04/valid-dtd-list.html

W3C. (2008d). *XSL Transformations (XSLT).* Retrieved Jan. 10, 2009, from http://www.w3.org/TR/xslt