The PracTEX Journal
TPJ 2005 No 02, 2005-04-15
Rev. 2005-04-18

What does XML give the LaTeX user?

Peter Flynn

Abstract

LaTeX users are being faced with the suggestion that they learn Yet *Another* Markup Language. Where does it all end? If LaTeX is so smart, why bother? This article is intended to provide some guidance for the confused.

1 In the beginning was TeX...

...and Knuth saw that it was good. So good, in fact, that although the internals have changed somewhat over the years, the user interface and the majority of the commands have stayed as they were first envisaged, because they did exactly what it said on the label.

By the 1980s, computers had become big enough (or small enough), and fast enough, and their interfaces friendly enough for people to start thinking about using them for other things than numerical computation.[1] Several things were developing at once:

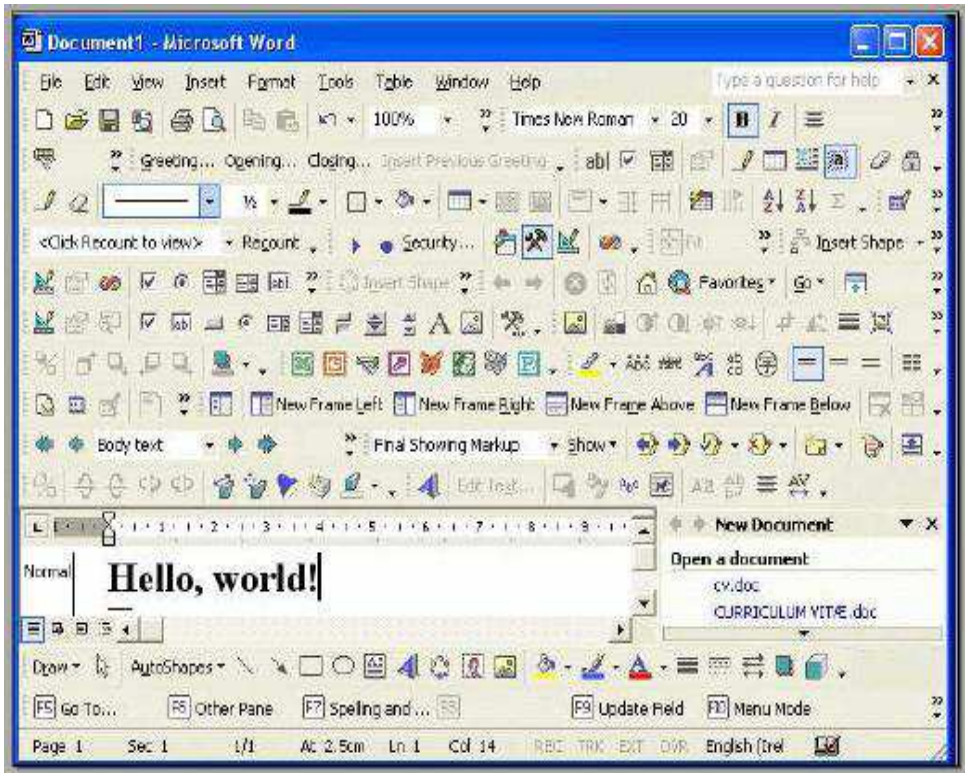
- word-processing (WP) and desktop publishing (DTP);
- document processing and document markup.

We know what happened to word-processing (I've used this illustration before): it either ended up like Figure 1 or as more complex but more powerful systems which have become the publishers' mainstay and nightmare. We also know what happened to document markup: it ended up as the underlying format of word-processors, DTP systems, TeX and LaTeX, SGML and then XML.[2]

[1] Actually, this had happened in isolated instances before but it hadn't caught on.

[2] SGML is the Standard Generalized Markup Language, ISO 8879:1985, the international stan-

Figure 1: Microsoft Word with all its nearest equivalent features to LaTeX enabled.



But document processing was different. People who write long or complex documents, documents with life-or-death importance, documents which have to last a long time, or which are needed in many different forms, have to be able to do things with their information other than just print it, which is all word-processing and DTP can do.

LaTeX and SGML developed at approximately the same time, but took different

dard syntax for describing markup languages. It has been in widespread use in some industries, but not in general use outside them, partly due to its complexity and partly due to the approximately contemporaneous rise of wordprocessing and LaTeX. XML is the Extensible Markup Language, a much simplified form of SGML designed for real-time use on the Web. It retains all the core features of SGML but is significantly simpler to process.

yet related stances. LaTeX provided a framework for standardizing the way that an author could instruct TeX what to do — how to format the document. Although it provides a way to structure a document by naming the different components consistently, it also allows the author to intermingle the details of formatting with the details of structure, and it requires the TeX or pdfTeX program, exclusively, (with the LaTeX macros) to untangle what has to be done at any given point.

SGML and XML provide a standardized way to identify the different components consistently, too. But they can codify the way of doing this for each type of document in a form that a program can check, so that certain rules can be enforced, such as making sure every figure has a caption, or that you cannot immediately go from a chapter to a subsubsection without having established a section and a subsection to contain it first.[3]

SGML provided a syntax for inventing specific markup languages in a rational and controlled manner for all kinds of applications. As SGML's successor, XML provides the same features, but in an even more reusable form. SGML grew up in the days when keystrokes and storage were expensive, so it has copious means of allowing abbreviation, and many complex rules about when and how you can minimize the markup, which is why the paragraph end-tag `</p>` is omissible in HTML.[4] XML simplifies this on the basis that machines and networks are now big enough and fast enough to handle it in full, so it threw out all the complex and hard-to-program bits in favor of doing things one way, and one way only.

2 It's all about control

The result of this rigor and robustness is that an XML document is machine-controllable. Unlike a LaTeX document, where a program reading it hardly ever knows what is coming next, an XML processor can work out exactly what stage it is at, at any point in the document. It's therefore possible to treat the document structure as a kind of map, with signage at every junction and all the items neatly labelled.

[3] Actually, you *could* do this in LaTeX, but it's not inherent in the design.

[4] The HyperText Markup Language is an application of SGML which emerged as part of the Web in order to describe simple documents for network use, and now forms the basis of most Web pages. Its successor, XHTML, is effectively the same language expressed in the fractionally different syntax of XML.

Suddenly, you're back in control of your document. If you decide that section 5.2 would make more sense to the reader being placed in chapter four between sections 4.7 and 4.8 you can just cut and paste it. An XML editor already knows where every element starts and ends, so there's no scrolling back and forth trying to find the right `\subsection` that signals the end of the preceding subsection. You can make it compulsory to have an Abstract, or you can prevent authors creating lists with only one item, or almost any other kind of structural check you want to impose. In seriously critical documents (aircraft maintenance manuals, for example), you can ensure that it is impossible to write part-number references in the disassembly instructions without also including the same references in the re-assembly instructions, thereby minimizing the chances of the engineer on the tarmac at Logan having two washers and a nut left over after putting it all back together.

At a deeper level, you can define complex descriptive structures like a project coordination schedule, or a literary comparison of different versions of a novel, or a computer program subroutine, in such a way that each component can be located and identified accurately — not just for formatting, but so that components can be referenced and reused elsewhere in the document. The result is that document structures, as well as their content, can become infinitely more reliable and robust than with LaTeX or any other DTP or word-processing system.

3 Cooperation, not competition

It is this ability to identify and reuse which lies at the heart of the most popular processing specification language for XML, the Extensible Stylesheet Language (XSL). This lets you process a document both in and out of sequence, so the natural flow of sections and paragraphs can be combined with the use and reuse of components elsewhere in the document, or in other documents, even live off the Web, in a single pass. Section headings can be "cherry-picked" from their locations immediately inside the start of each section to compose a table of contents; similarly at the end of the document, all indexable or cross-referenceable components can be revisited, sorted, and reformatted with location references to make the index and bibliography — in the same pass.

LaTeX can do all this too, but it requires multiple passes and external programs, and can only output DVI or PDF. XSL comes in two flavors; XSL Formatting Ob-

jects (XSL:FO) and XSL Transformations (XSLT).

- XSL:FO produces a stream of typesetting-oriented formatting specifications and is usually implemented in conjunction with a PDF generator, so you write your XSL file as a structured specification of what gets put where.

- XSLT can output any text format: notably XHTML so that you can create Web pages, but also any other plain text format, which of course includes LaTeX.

And here's the pay-off. XSL:FO still requires you to build into your XSL file all the code to handle every possible combination of circumstances which may occur in your type of document, largely from scratch. FO processors have scant understanding of what documents are: they simply output PDF code. In the hands of an experienced document engineer with extensive typographic training, this can produce excellent results.

But LaTeX already possesses the acquired knowledge of decades of document production and the experience of millions of users and thousands of programmers and document creators. In addition, it has typographic sophistication far in advance of any FO processor. An XSLT file written to output Web pages can be reused with minimal changes to output a LaTeX file, leaving all the hard bits to the tried and trusted LaTeX packages for handling the formatting, much in the same way that it is possible to leave the formatting of HTML to the browser. XSLT can even be used to overcome some of the harder tasks in LaTeX, such as customizing the semantics of cross-references.[5]

By handing over the details of formatting to LaTeX, which knows what to do, you can keep the quality high, but at the same time reap some additional benefits:

- improved document control;
- greater editorial reliability;
- document persistence (XML is based on the international standard [SGML] so the file format is not going to go away);
- wider document acceptability (for the same international standard reasons, plus the universal reliance on Unicode);

[5] In *Formatting Information* I use this to add locational information to section references so that if a label ("ID" in XML-speak) is in the first or last paragraph of a section, the reference ("IDREF" in XML) can be output as "see the first [or last] paragraph of section X".

- availability of a wide range of software, much of it free;
- ability to output multiple formats from a single master;
- ability to use robust addressing to identify and locate information.

Using XML with XSLT and LaTeX adds another step and another processor (a program to apply the XSLT file to your XML document, such as Saxon), and this is a step you shouldn't take lightly. It also means learning a different syntax (XML) to mark up your documents, and another language (XSLT) to specify the LaTeX output.

But for a document or series of any significant size, the gains in document control, re-use, editability, and consistency make it well worthwhile, and I hope I have shed a little light on why I think it is a productive step to take.

Resources

The Web site of the World Wide Web Consortium (W3C) at `http://www.w3.org/` has links on its front page to separate sections for each of these technologies (except LaTeX, of course). There is an XML Frequently-Asked Questions document (FAQ) at `http://www.ucc.ie/xml/`. Users can discuss XML and ask questions on the mailing list at `http://listserv.heanet.ie/xml-l.html` and the Usenet newsgroup at `news:comp.text.xml` (but read the FAQ first, to avoid re-asking frequently-asked questions in the forums).

Users interested in an example of these developments are invited to inspect the code I used for *Formatting Information* (in `http://www.ctan.org/tex-archive/info/beginlatex/src`), in particular the XML document file `beginlatex.xml`, the XSLT stylesheets `typebook-latex.xslt` and `typebook-html.xslt`, and their respective output in `beginlatex-typebook.tex` and the contents of the `html` directory which is sibling to the `src` directory.