LaTeX on the Web

Peter Flynn
Electronic Publishing Unit, University College, Cork, Ireland
pflynn@ucc.ie
http://imbolc.ucc.ie/~pflynn

Abstract

This paper presents some techniques for use when authoring in LaTeX which can be used to minimize the conversion problems where a document is to be converted to HTML for serving to the Web, while continuing to produce the quality of typesetting for PostScript/PDF that users have become accustomed to.

The Web as the new mode of publishing

Between TeX users, distributing source documents via the Internet (whether Web, email, or other facility) is easy because the types of files are known (e.g. `.tex`, `.bib`) and they are easily reprocessed. It is also relatively easy to make the final-quality typeset output available to others as single files using PostScript or PDF.

However, PostScript files can be huge, especially if they contain bitmap graphics, and they require a reader which few outside the graphic arts field have installed (although it's simple to do and freely available). PDF browsers are readily available, and many of them implement simple HTML-style hyperlinking for cross-references and URIs, but they have had other limitations, including font instability, the lack of inward addressability, and some typographic alignment problems.

Large or complex documents benefit from being split into chunks for serving, and from being served faster than PostScript or PDF by using HTML or XML and CSS. But HTML editors are notorious for their lack of structural, typographic, and document management facilities — which LaTeX users are accustomed to having at their disposal. For all the hype, XML editors have failed miserably to live up to or exceed these expectations.

Conversion from LaTeX to HTML is widely available, but unavoidably suffers from the inherent mismatch between feature-sets, and from the inherent reprogrammability of LaTeX. Authoring in XML, with conversion both to HTML/CSS and to PDF-via-LaTeX is one option, but has its own drawbacks in the learning curve and the early quality of some software.

Between consenting TeX users ...

For TeX files, the file format is known and expected, and the software to handle it already exists and is installed. TeX files are small by comparison with other systems, and can be transferred by HTTP, FTP, email, etc., without licensing or copyright issues, and there are no known security problems (viruses, worms, etc.).

In fact, it's not necessarily so easy. The user's browser, client, or operating system may not know what to do with the file types, even if a TeX system is installed. The sender's server or operating system may not know what to do with the file types either, and may serve them with inappropriate or misleading labels, i.e., MIME Content-Types.[1]

To be sure the files can be used properly, we need to know what ancillary files are needed, and whether non-default fonts need to be sent as well. Even where the user's system recognises the file types, what do you do actually want to do with `.tex` files when you click on them in a browser or FTP client window? Display them? Edit them? Process them? Save them? Rename them to something else?

Serving non-source files

For many applications it is sufficient to serve the output, rather than the source. While it is possible to serve DVI files, it's unusual because there can be problems if there is any use of fonts outside the default set supplied with a standard installation of TeX or LaTeX, and there can also be problems with the inclusion of images, which need to be provided separately. It will almost certainly be self-defeating if the objective is co-operative editing.

PDF For final-format documents, it's important to remember that most users nowadays have never seen or heard of a PostScript file: as explained earlier,

[1] There was a proposal some years ago to register `.tex` etc as known media types, but we are still using `application/x-latex`. TUG needs to do something about this or someone else will register them as something else.

PDF is ubiquitous, there is a choice of viewers, it has become the *de facto* standard, and it provides for hyperlinks.

Against it count the poor handling of bitmap fonts in some Adobe readers, and the problems experienced with the feature for shrinking or expanding the page-image to fit the paper. This is probably the cause of most grief: if you have carefully set LaTeX to produce an exact text width and height, with margins to fit your size of paper, it can be surprising to get email back from a user complaining either that it doesn't print properly, or that the dimensions are not what you claim they are. In these cases they have almost certainly allowed Acrobat Reader to scale the page up or down for their local paper size.

HTML To take full advantage of the Web today means serving HTML (or, increasingly, XHTML, the more rigorous XML version of HTML). From a LaTeX source this means using a conversion program, of which there are several available: the two most common are LaTeX2HTML and TeX4ht.

However, many browsers still lack rendering ability and font control. The use of home-brew macros will often defeat convertability by making it virtually impossible for a converter to figure out how the output is to appear. Generally, the converters do an excellent job, but they cannot be 100% error-free. Differences between browsers can cause user community problems if not everyone uses the exact same version and build (usually only achievable in tightly-controlled corporate environments). But the fundamental reason for differences in rendering is simply the feature-set mismatch between TeX and HTML: they are intended to perform different tasks, so it is perhaps unreasonable to expect them to be completely congruent.

XML The long-term solution to many of these problems may be to author in XML and convert to HTML for interactive browsing and to LaTeX or ConTeXt for generating PDF.

XML has the advantages that it is very controllable, conversion is robust, it is a *de facto* standard, and that open source solutions are available throughout the process.

Against it stand a steep learning curve for authors unused to structured document markup (LaTeX users have a head start here), the poor quality of much XML editing software, and the need for the additional steps to get to HTML or PDF.

Overall, the fact that an XML document can be reused much more successfully than a LaTeX document tends to indicate that this is the direction for authoring, provided the problems with editors can be overcome.

Making more use of your LaTeX

For successful conversion to HTML there are some key steps you can take:

- Keep your source code neat;
- Make it predictable and recognizable;
- Use white-space above and below all stand-alone control sequences;
- De-abbreviate any home-brew shortcut macros before conversion;
- Re-use equivalent LaTeX environment and command names rather than inventing your own;
- Above all *be consistent*.

An example of this put into practice might be something like this:

```
\section{Making more use of your \LaTeX}

For successful conversion ...

\begin{itemize}

\item Keep your source code neat;

\item Make it predictable and recognizable;

...

\item Above all, \emph{be consistent}.

\end{itemize}
```

Seen and heard ...

During the course of the Practical TeX conference, a number of people had been discussing these topics, and I noted down a few of the opinions:

- Author in HTML and make images for formulae;
- Author in XML and use XMLTeX;
- Author in Word using named styles, convert to XML with *DynaTag*, then to LaTeX;
- Author in Word and use the inbuilt 'Save As... XML', then convert to LaTeX;
- Author in *AbiWord* and 'Save As...' both LaTeX *and* XML;
- Write a version of LaTeX that outputs HTML (after all, it works for PDF...).

As always, there is an infinity of solutions to choose from, and getting your own document onto the Web may involve pieces from more than one of the pathways I have described.