TUG 2011 — program and information

Wednesday October 19

8:30 am	*registration*	
9:30 am	Barbara Beeton, TEX Users Group	*Welcome*
9:35 am	Ross Moore, Macquarie University	*Further advances toward Tagged PDF for mathematics*
10:10 am	Rishi, River Valley Technologies	*Creating magical PDF documents with pdfTEX*
10:45 am	*break*	
11:20 am	CV Radhakrishnan, River Valley Tech.	*TEX4ht — A Swiss army knife for TEX*
12:15 pm	Karel Skoupý, Lingea	*Data structures in ε-TEX*
12:30 pm	*lunch*	
1:40 pm	Kaveh Bazargan, River Valley Tech.	*Why TEX is more relevant now than ever*
2:15 pm	Alan Wetmore, US Army	*e-Readers and LATEX*
2:40 pm	*break*	
3:15 pm	Boris Veytsman and Michael Ware, George Mason Univ. & Brigham Young Univ.	*Ebooks and paper size: Output routine hacking made easy*
3:50 pm	Rishi	*Automated generation of ePub from LATEX*
4:25 pm	q&a	

Thursday October 20

9:35 am	Jean-Michel Hufflen, University of Franche-Comté	*A comparative study of methods for bibliographies*
10:10 am	Brian Housley, GCCS GmbH	*Making a package for flexible letter & page headings*
10:45 am	*break*	
11:20 am	Didier Verna, EPITA R & D Lab.	*Toward LATEX coding standards*
11:55 pm	Frank Mittelbach, LATEX3 Project	*LATEX3 architecture and current work in progress*
12:30 pm	*lunch*	
1:40 pm	Dave Crossland, Wimborne, UK	*Freeing fonts for fun and profit*
2:15 pm	Boris Veytsman & Leyla Akhmadeeva, Bashkir State Medical University	*Towards evidence-based typography: Experiment design*
2:40 pm	*break*	
3:15 pm	Karel Skoupý	*Typesetting fancy multilingual phrase books with LuaTEX*
3:50 pm	Dominik Wujastyk, Austria	*Typesetting Sanskrit in various alphabets: XETLATEX, TEC files, hyphenation, and even XML*
4:25 pm	q&a	

Friday October 21

9:35 am	Pavneet Arora, Canada	*Typesetting with masonry*
10:10 am	Jean-luc Doumont, Principiae	*Integrating TEX and PDF seamlessly in pdfTEX*
10:45 am	*break*	
11:20 am	Sukumar Sankar, S Mahalakshmi and L Ganesh, TNQ Books and Journals	*An XML model of CSS3 as a XILATEX-TEXML-HTML5 stylesheet*
11:55 am	S.K. Venkatesan, TNQ	*On the use of TEX as a general markup language for HTML5*
12:30 pm	*lunch*	
1:40 pm	Stefan Kottwitz, Germany	*Bringing together TEX users online: From Usenet to Web 2.0 and beyond*
2:15 pm	Manjusha Joshi, India	*A dream of computing and LATEXing together: A reality with SageTEX*
2:40 pm	*break*	
3:15 pm	Dominik Wujastyk	*My father's book: Typesetting and publishing a family memoir*
3:50 pm	Petr Sojka, Masaryk University	*Why TEX math search is more relevant now than ever*
≈ 4:30 pm	*end*	

Kaveh Bazargan

Why TEX is more relevant now than ever

TEX is around 30 years old, and was conceived and written before the advent of laser printers, personal computers, PostScript and of course the Internet. At that time the idea of WYSIWYG document editing was just a futuristic idea. When people jumped on the WYSIWYG bandwagon, it was predicted that old technologies such as TEX which used mark-up for text would disappear in time. The advent of the Internet brought mark-up to the attention of the public. Somehow it was acceptable again. The recent move to the semantic web and HTML5 has brought renewed attention to mark-up and the need for clear structure in text. I suggest that we have gone full circle and now realise that mark-up is everything. And TEX, which has the most readable and minimalist mark-up might just be the best tool today for structured documentation.

Dave Crossland

Freeing fonts for fun and profit

Google Web Fonts (`www.google.com/webfonts`) has published hundreds of libre fonts during the past year, at an accelerating pace. Dave Crossland has been driving this through consultancy for Google, and presents his personal opinion about the past, present and future of libre fonts — showcasing the latest designs, designers and tools. (Please note that this talk is entirely the personal opinion of Dave Crossland, and does not represent the views of Google, Inc. in any way.)

CV Radhakrishnan

TEX4ht — A Swiss army knife for TEX

There are several technologies to translate LATEX sources into other markup formats like HTML, XML and MathML. TEX4ht assumes a premier position among them owing to the fact that it makes use of the TEX compiler for translation, which helps to assimilate any complex author macros used in the document. This talk provides an overview of how to configure TEX4ht to output custom markup needed by users.

Jean-luc Doumont

Integrating TEX and PDF seamlessly in pdfTEX

In its ability to generate graphical elements, TEX is basically limited to horizontal and vertical black rules. Extended versions such as pdfTEX add color options and, especially, the possibility to draw more freely on the page by inserting raw code (PDF code in the case of pdfTEX). Still, these two coding environments — TEX and PDF — are too often regarded as disjoint. It would be nice to integrate them seamlessly, for example, to use in PDF code a color or a dimension assigned or calculated in TEX. This presentation points out the challenges of such a consistent and transparent TEX–PDF integration, proposes a set of solutions, and illustrates how these solutions help create graphs flexibly or design pages consistently on a grid.

Brian Housley

Making a package for flexible letter & page headings

There are many packages for making letters and page headings with logos, etc., in LATEX, but many do not take into account user changes to the page dimensions and so the heading may not be in the correct, centred, place. Also, the user should be able to specify different types of letter, including private ones, without having to use another version of the package. We describe how to achieve a package to produce headings and letters which is robust against page layout changes and which permits the user to define all the fields, including the logo, himself. Obviously, to redesign the presented headings one must be a little versed in (LA)TEX but the user specifies his own details very easily. Described will be: 1) How to determining the absolute position of the heading on the page. 2) Support for various language styles. 3) Using class option (`.clo`) files for defining types of letters — standard office, private, signed letters, etc. 4) Using class option files to specify the user information (name, address, etc.) for the various letter types and languages. 5) Producing letters, merge letters (possible signed) and headings. 6) Ensuring the letter to-address fits in a C5 and C5/6 window. 7) Hints on how to change the heading to your own style. The talk should be suitable for general users.

Jean-Michel Hufflen

A comparative study of methods for bibliographies

First, we recall the successive steps of the task performed by a bibliography processor such as BIBTEX. Then we show how this modus operandi has been adapted by tools such as the packages `natbib`, `jurabib`, `biblatex`. We also explain the advantages and drawbacks of using other processors like Biber or MlBIBTEX.

Manjusha Joshi

A dream of computing and LATEXing together: A reality with SageTEX

Researchers search for some computational package for their results. At the time when they have good output, they begin worrying about how to insert it in their LATEX document. They have to keep track of their output, formatting and then insert it at the appropriate places in the document.

The SageTEX package is a blessing in these situations. It calls the powerful open source maths

server Sage, to compute and embed the result into a TEX document.

Stefan Kottwitz

Bringing together TEX users online: From Usenet to Web 2.0 and beyond

It all began in the 1980s with mailing lists such as `texhax`, and Usenet. The online discussion board `comp.text.tex` emerged around 1990, where TEX hackers gathered and still frequent it today.

On the continuously developing Internet, TEX user groups created mailing lists, built home pages and software archives. Web forums turned up and lowered the barrier for beginners and occasional TEX users for getting support.

Today, TEX's friends can also follow blogs, news feeds, and take part in vibrant question and answer sites.

In this talk we will look at present online TEX activities.

Frank Mittelbach

LATEX3 architecture and current work in progress

Overview of the current state of LATEX3.

Ross Moore

Further advances toward Tagged PDF for mathematics

This is the 3rd presentation on on-going efforts to develop the ability to generated Tagged PDF output using pdfTEX, in conjuction with other software tools. In this talk I'll show how recent improvements to Adobe Reader and Adobe Acrobat Pro software have increased the usefulness of Tagged PDF documents, containing a MathML description of the TEX-typeset mathematical content.

In particular, by careful specification of the words to be "Read Out Loud", mathematical content can be conveyed quite effectively to the visually impaired. Also, using Adobe's Acrobat Pro as the PDF browser, the ability to export to XML means that a fully marked-up, with MathML for the mathematics, version of the PDF document's contents can be obtained at will from the same file that displays the high-quality typeset visual appearance.

Examples will be shown of diverse mathematical content, generated automatically from standard LATEX coding, along with suitably generated MathML descriptions.

Rishi

Automated generation of ePub from LATEX

We will give a live demonstration of the following fully automated workflow used at River Valley: LATEX to XML to Deliverable, where "Deliverable" might be PDF or ePub.

Rishi

Creating magical PDF documents with pdfTEX

PDF has a rich specification. But Adobe Distiller does not exploit all these specifications. We'll demonstrate how pdfTEX can create useful PDF files that are difficult or impossible to create using other technologies. Examples are: PDFs showing differences in two TEX source files; PDFs with useful pop-up tools; and a simple but useful composite PDF for comparing two nearly identical PDF files.

Sukumar Sankar, S Mahalakshmi and L Ganesh

An XML model of CSS3 as a XLATEX-TEXML-HTML5 stylesheet

HTML5 and CSS3 are becoming the popular language of choice in the web. However, quite like HTML, CSS is prone to errors and difficult to port, so we propose an XML version of CSS that can be used as a standard for creating stylesheets and templates across different platforms and pagination systems. XLATEX and TEXML are some examples of XML that are close in spirit to TEX that can benefit from such an approach. Modern TEX systems like XETEX and LuaTEX use simplified `fontspec` macros to create style sheets and templates. We use XSLT to create mappings from this XML-style sheet language to `fontspec`-based TEX templates and also to CSS3. We also provide user friendly interfaces for the creation of such an XML style sheet.

Karel Skoupý

Typesetting fancy multilingual phrase books with LuaTEX

We used TEX for typesetting a series of phrase books with a fancy graphical design. Each book contained the same content for a different language pair. There were several dozens of them semi-automatically generated and thanks to the way how the language data were organized and thanks to TEX as a typesetting engine this process was very time and cost effective.

We have developed interesting TEX macro modules and used many advanced features of pdfTEX and LuaTEX to meet the challenges raised by the graphical design and by some non-Latin script languages. We will show the general structure and discuss some interesting problems and their pdfTEX/LuaTEX solutions.

Karel Skoupý

Data structures in ε-TEX

For the construction of macro packages, TEX is used as a programming language. Unlike general programming languages it lacks complex data structures. We present the experience of providing record and array data structures and the supporting

operations using ε-TEX features. They were successfully applied in real projects for parametrization and as a base for special table module involving complex dimension calculations. We will show how the abstraction level provided by more powerful data-structures can simplify and unify TEX low-level code.

Petr Sojka
Why TEX math search is more relevant now than ever

TEX is around 30 years old, and was conceived and written before the advent of MathML, not to mention the Internet. At that time the idea of indexing and searching mathematics was just a futuristic idea. When people jumped on the Google bandwagon, it was predicted that old technologies such as TEX mark-up for math would disappear in time (it is not used for tokenization and indexing properly). The advent of the Internet and W3C brought mark-up and global search to the attention of the public. Somehow it was acceptable again. The recent move to the semantic search and MathML has brought renewed attention to the need of unambiguous canonical math representation in texts.

As part of the project of building the European Digital Mathematics Library (`http://www.eudml.eu`) we have designed and implemented a math search engine, MIaS (`http://nlp.fi.muni.cz/projekty/eudml/mias`). It currently indexes and searches in more than 160,000,000 formulae originally written by authors in TEX in their scientific papers. We will present the system and will discuss the ways towards global math search engine based on the TEX math notation.

S.K. Venkatesan
On the use of TEX as a general markup language for HTML5

The TEX syntax has been fairly successful at marking-up a variety of scientific and technical literature, making it an ideal authoring syntax. The brevity of the TEX syntax makes it difficult to create overlapping structures, which in the case of HTML has made life so difficult for XML purists. We discuss S-expressions, the TEX syntax and how it can help reduce the nightmare that the HTML5 markup is going to create. Apart from this we implement a new syntax for marking-up semantic information (microdata) in TEX.

Didier Verna
Toward LATEX coding standards

Because LATEX (and ultimately TEX) is only a macro-expansion system, the language does not impose any kind of good software engineering practice, program structure or coding style whatsoever on you. As a consequence, writing beautiful code (for some definition of "beautiful") requires a lot of self-discipline from the programmer.

Maybe because in the LATEX world, collaboration is not so widespread (most packages are single-authored), the idea of some LATEX Coding Standards is not so pressing as with other programming languages. Some people may, and probably have, developed their own programming habits, but when it comes to the LATEX world as a whole, the situation is close to anarchy.

Over the years, the permanent flow of personal development experiences contributed to shape my own taste in terms of coding style. The issues involved are numerous and their spectrum is very large: they range from simple code layout (formatting, indentation, naming schemes etc.), mid-level concerns such as modularity and encapsulation, to very high-level concerns like package interaction/conflict management and even some rules for proper social behavior.

In this talk, I will report on all these experiences and describe what I think are good (or at least better) programming practices. I believe that such practices do help in terms of code readability, maintainability and extensibility, all key factors in software evolution. They help me, perhaps they will help you too.

Boris Veytsman and Michael Ware
Ebooks and paper size: Output routine hacking made easy

It might be surprising for a casual TEX user, but the classical TEX engine does not know anything about the physical dimensions of the paper the text is printed upon. One may argue that it imitates the traditional compositor, who prepared the matrix, but was not involved in the actual printing. Another explanation would be that a printer (a device, not a person) usually has a stock of pre-cut paper sheets, and there is not much use in trying change the paper size from the typesetting program.

While both dvips and pdfTEX provide the possibility to set the paper dimensions using either PostScript or PDF commands embedded in the text, it is commonly done only once per document. This practice assumes uniform paper size for all pages.

The situation with ebooks is different. "Paper" for them is virtual, and we are not constrained by the constant page size of traditional books. On the other hand, typesetting for the screen *requires* paper size tuning.

Since the page size is determined during `\shipout`, we need to deal with the output routine — the task often considered to be too arcane for unexperienced TEXnicians.

In this talk we show that playing with the output routines does not necessarily involve

high wizardry. We show how to use the output routine to determine the required paper size and to communicate it to the driver.

The talk is intended for a beginning to intermediate TEX hacker.

Alan Wetmore
e-Readers and LATEX

2011 has seen many e-readers arrive on store shelves; a new generation of "touch screen" devices including the Nook Simple Touch, Kobo eReader Touch, and i-River Story HD. They all have a capability of loading user created content, so the question arises; how well can they support my legacy documents? The answer might be surprisingly well. After we understand the capabilities and some of the limitations we will explore how we can re-purpose older documents and prepare new LATEX documents for use with these e-readers.

Dominik Wujastyk
My father's book: Typesetting and publishing a family memoir

In 2010, I typeset a 650-page book of memoirs, political essays, and biographical sketches written by my 97-year-old father. The book is in the Polish language, and was published by the University of Lublin. For the design and typsetting I made choices that stylistically echoed my father's life-long links with Malta and Poland. Due to financial restrictions at the University of Lublin, I worked out a cost-effective pathway for printing and distribution using an American web-based printing and distribution service. The final result is of a high standard, and has been gratifyingly well received by all parties. Some niggles remain, however, regarding publicity and distribution. In this paper, I shall describe my choices and discoveries in producing my father's book.

Dominik Wujastyk
Typesetting Sanskrit in various alphabets: XELATEX, TEC files, hyphenation, and even XML

The XETEX extended TEX engine provides a wealth of sophisticated features, and meets many of the long-felt needs of people working with multilingual or multi-script texts. I shall describe the use of XELATEX for typesetting Sanskrit, with both Roman- and Devānagarī-script inputs, and Roman- and Devānagarī-script outputs. I shall describe the complexities of getting differently hyphenated Sanskrit in different scripts. Finally, I shall offer an example of a free IBM XML tool that uses a XELATEX TEC file to auto-convert Sanskrit between Roman and Devānagarī for screen display via HTML. If all this sounds a bit messy, it is. But the results are sometimes quite amazing, and open up exciting possibilities for the beautiful printing of Indian texts.

A class and style for producing flexible letters and page headings

Brian Housley

19th. August, 2011

Abstract

A package is presented which permits the user to specify easily, with the aid of self defined key-words, letters (with a logo and private) and headings. The *heading* may include a footer and the letter provides commands to include a scanned signature, two signees and works with the merge package. It illustrates using zero width boxes and converting lengths into counts.

1 Introduction

Your first thoughts are probably "Not another LaTeX letter package" but, maybe, this package does offer something extra and useful. The idea was at first to produce headers for various department, committees, etc., and the letter was an easy extension. The main ideas for the package are:

- Permit the user to specify key-words which, together with the default or specified language, invoke various styles of the heading.
- With letters one may define an option to produce a private letter, i.e., one with no logo but a from-address.
- The header is always centred, at the top of A4 paper.
- Ensure the to-address is centred in a C5/C6[1] window envelope.
- Use a style file to produce headings as for letters with a horizontal rule underneath.
- The text for the heading together with the footer is produced by key-words dependent on a user defined option.
- A command `\closingtwo` may be used to produce letters with two signees.

[1] I would have supported the North American stationery sizes but I have no access to such envelopes, etc.

- The *merge* package by Graeme McKinstry[3] works.
- A scanned signature may be used — especially useful with *merge* letters.

2 The general design

The files used are shown in figure 1 where the shaded files should be provided by the user. The package loads the packages *graphicx* and *ifthen*.

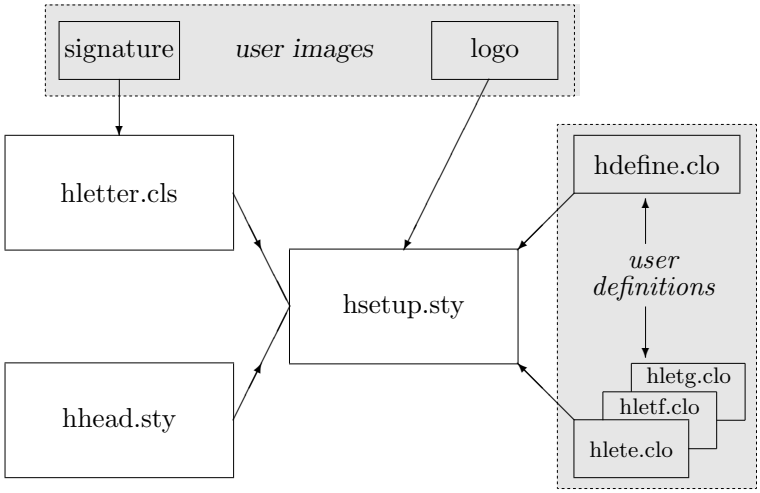


Figure 1: Files used in producing letters and headings

The function of the files are:

`hletter.cls` The class definition file based upon the standard LaTeX letter class[2]. It redefines various commands and defines new ones (see later).

`hhead.sty` The package for producing the headings on top of a page. Call with `\usepackage{hhead}` and the command `\heading` is defined to produce the heading(s).

`hdefine.clo` The user file which defines key-words for the various headings.

`hlet<`***lng***`>.clo` The user file which defines the fields for the heading where *lng* is the letter e, f or g for the languages English (actually British), French and German.

`logo` the image file to produce the logo.

`signature` A scanned signature which may be used in the letter(s).

`hsetup.sty` The file which does most of the work and defines the command to produce the headings and which reads in the files `hdefine.clo` and `hlet`<*lng*>`.clo` where *lng* is specified in the class or style options (default is English).

3 Fields used in the header

Figure 2 shows the commands which define the text where the command is shown. Also there is a command `\centrepos{`*n*`}` where *n* is a length specifying the offset of the centre text from the middle of the paper. The default is 10 mm and it may be negative.

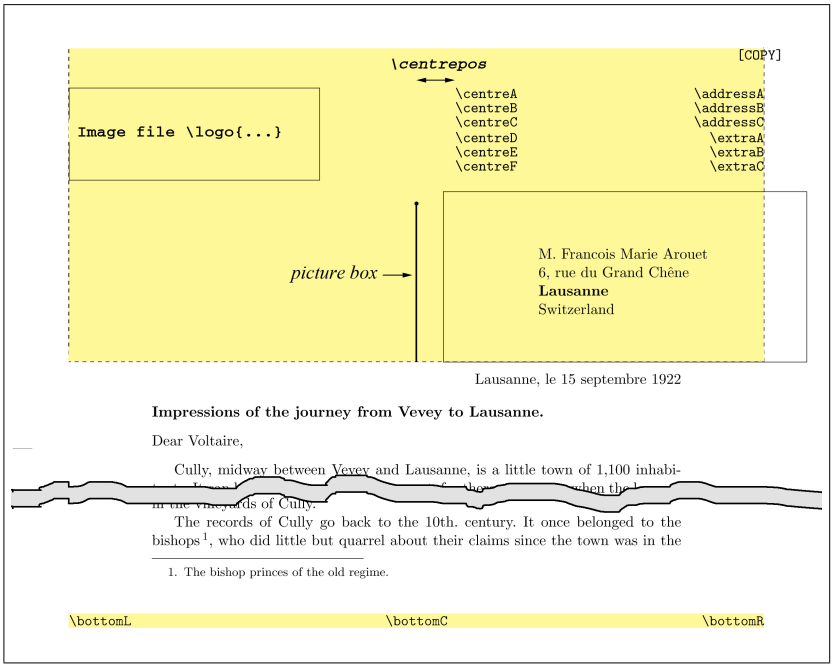


Figure 2: How most of the fields are defined

If a header alone is being produced then it will have a horizontal rule below it of a default width of 180 mm. With the command `\barlength` one may change this length even making it 0 mm. If the logo is very high then the header will be increased accordingly.

4 The layout of the header

Obviously the header for a letter is different from a simple header but both are produced using the *picture* environment and in both cases the origin of

the picture must be the same.

The header must be in the centre of the paper and the offset from the beginning of the text is calculated when the heading is produced. Thus any dimension changes the user may make are taken into account.

4.1 Horizontal positioning

The solution is to space horizontally and then make a LaTeX *picture* of zero width as shown in figure 3.

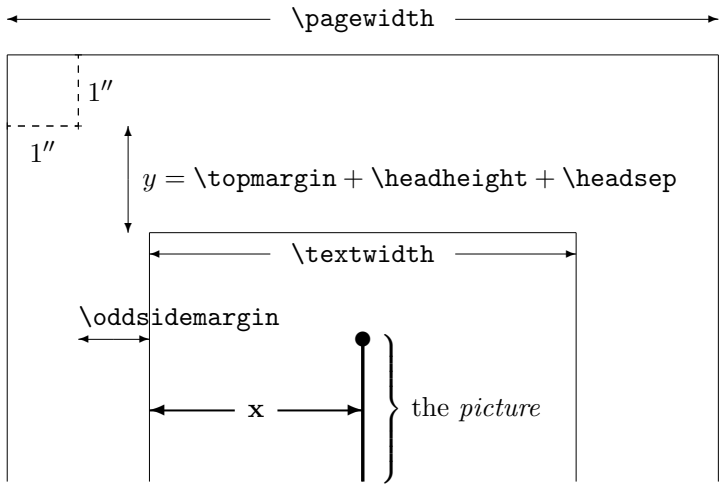


Figure 3: Obviously $\mathbf{x} = .5\texttt{\textbackslash pagewidth} - 1'' - \texttt{\textbackslash oddsidemargin}$

4.2 Vertical positioning

For letters the header stretches to the bottom of the *to-address* box (for a C5/6 envelope) and is 91 mm from the top of the paper. For the simple header (using the package *hhead.sty*) the bottom of the header is 41 mm from the top of the paper but this may be increased if the logo is large.

4.2.1 The letter

As seen in figure 3 we need to calculate $h = 91mm - 1'' - y$ and if this value is negative then a warning "top margin seems to be too large" is issued. This can only happen if the text area is lower than the *to-address* box.

The variable h is a length variable and is stored as scaled points but for the picture we need a counter which depends on `\unitlength`. Thank goodness, TeX is very accommodating and we set a *counter* to the length h and then divide by `\unitlength`. The value is truncated but I think

a header to within 1 mm is sufficiently accurate but one could modify the package to use a *unitlength* of 0.1 mm if one wishes more accuracy.

The command `\begin{picture}(0,h)(0,-41)` is used to produces the picture which contains the header.

4.2.2 A simple header

Here the value calculated is $h = 46mm - 1'' - y$ and again we divide by *unitlength*. If the height of the logo is large then the value of the offset of the rule under the header is increased and the picture must be higher and the lower left of the picture is set to a negative value.

If the document is in *twocolumn* format then the command `\twocolumn` is used to ensure that the header spans the two columns.

5 The user files

`hdefine.clo` Defines the names which the user wants to select the various type of heading together with a sequentially increasing integer. An example is:

```
\logo{GCCS}
\newoption{private}{1}
\newoption{signit}{2}
\newoption{bruni}{3}
\newoption{test}{4}
```

Note that the logo may also be specified in this file to provide a default which may be changed the *hlet* files. the file `hsetup.sty` simply defines a new option which, if used, sets a global counter:

```
\newcommand*{\newoption}[2]{\DeclareOption{#1}%
  {\global\hltype=#2}\typeout{*** Option #2 has name #1}}
```

and types out the option and value in the log file.

`hlet<`***lng***`>.clo` For each of the languages English, French and German which are used (one could add other ones) the user must provide a file which defines the fields for the option specified in `hdefine.clo`. The structure is shown in figure 4.

the logo The command `\logo[ht]{file}` sets the logo file and if the optional height is not specified then 24 mm is used. This command may be used in the definition file and/or in the *hlet* file(s).

signature file A scanned signature may be inserted — particulary useful for form/merge letters. Define the file with the command `\sign[ht]{file}` and if *ht* is not specified then it will be 15 mm high.

```
% Letter options for English
\ifcase\hltype
% case = 0 (no option - GCCS default)
 definitions for default case
\or
% case = 1 (private)
  \address{...
   defining an address give a private letter
  ...}
\or
% case = 2 (signit)
 definitions for signit option
\or
% case = 3 (bruni)
 definitions for bruni option
\else
% all other cases (should never be used)
  \addressA{?} \addressB{?} \addressC{?}
  \extraA{Telephone: ?} \extraB{Telefax: ?} \extraC{eMail: ?}
\fi
```

Figure 4: Structure of definitions file for English in `hlete.clo`

6 Creating a letter

Assuming that the define file and the *hlet* files have been created one creates a letter in the usual LaTeX way but with a few additional commands. The class *hletter* is used with options point size, language (default English) and maybe one of the user options defines in `hdefine.clo` which the selects the required letter type.

6.1 A short summary of the letter commands

`\signature` The single argument is the name under the closing signature. Terminal multiple lines with `\\`.

`\address` The from-address and, when used, makes a private letter without a logo. Terminal multiple lines with `\\`.

`\reference` If used the argument is set centred under the opening for English and above, left justified, otherwise.

`letter` **environment** Starts the letter and the argument is the *to-address*.

`\date` Set the date to be printed under the header.

`\opening` This command has an optional argument which, when used, is placed in typewriter font at the top right of the letter, e.g.,
`\opening[{[COPY]}]{Dear Voltaire,}`.

`\closing` The argument is the closing text above the signature. Terminal multiple lines with `\\`.

`\closingtwo` Supplies the closing which is centred above two signatures. The `\signature` command should contain two names, each line separated with an '&' as in tabular (which it is), e.g.:

```
\signature{Dr.~A. Nother & Mr.~B. Bitt \\ CEO & CIO}
\closingtwo{Yours Faithfully,}
```

`\encl` A list of enclosures; multiple lines separated with `\\`.

`\cc` A list of persons who are to receives copies of the letter; multiple lines separated with `\\`.

7 Creating simple headings

In the document prologue one loads the package *hhead* with any optional argument such as language. A header is produce with the command `\heading` which has an optional argument which if used will be printed top right of the page. If *heading* is used more than once in a document then a *cleardoublepage* is issued and the page count is reset.

8 Form or merge letters

The package *merge* from Graeme McKinstry works well with this letter package. It reads a file of {*to-address, opening*} pairs which are used to create a letter which is addressed to many recipients. When TeX reads from an external file it honours grouped lines, i.e., to enter the address over many lines in the merge file (new lines terminating with `\\`) enclose the address in `{...}`. The packages uses *tabular* to set the to-address so these brackets, if present, must be removed. Fortunately the TeX-Book[1] (as usual) provides the answer and the to-address is produced with the, at first look, rather strange commands:

```
\def\dotoaddress#1{\setbox0\hbox{\expandafter\cmda#1}
    \ifnum\myc=1\settoaddress{#1}\else
    \expandafter\settoaddress#1\relax\fi}
\def\settoaddress#1{\global\setbox\addrbox
    \hbox{\begin{tabular}{@{}l@{}}#1\end{tabular}}}
%
\newcount\myc
\def\cmda#1{\global\myc=0 \cmdb#1\end}
\def\cmdb#1{\ifx#1\end \let\next=\relax
   \else \global\advance\myc by1 \let\next=\cmdb\fi \next}
```

Thus the creation of the address file is very easy and readable.

To make it a little easier, a small modification to `merge.sty` has been made so that *after* the first address pair one can insert a % as the first character of a line. The modified version is called `mergeh.sty`.

9 Examples

In the examples the extent of the contents of the picture are shown together with its origin to illustrate what is happening. The file `hdefine.clo` was as shown in section 5.

1. The LaTeX file contained:

```
\documentclass[11pt,english]{hletter}
\begin{document}
\signature{Sir Frederick Treves\\
    Sergeant-Surgeon to His Majesty the King}
\reference{Impressions of the journey from
    Vevey to Lausanne}
\date{Lausanne, le 15 septembre 1922}
\begin{letter}{M. Francois Marie Arouet \\
               6, rue du Grand Ch\^{e}ne \\
               \textbf{Lausanne} \\
               Switzerland}
\opening[{[COPY]}]{Dear Voltaire,}
\closing{I remain, Sir,\\yours Truly,}
\vfill
\cc{All Smiths in London\\ Mademoiselle S. Curchod}
\encl{Tourist guide to Switzerland.\\ Plan of Cully.}
\end{letter}
\end{document}
```

and the default (value=0) in the file `hlete.clo` specified:

```
\addressA{Largitzenstrasse 15}
\addressB{CH--4056 Basle}
\addressC{Switzerland}
\extraA{Telephone: +41 (61) 345 78 90}
\extraB{Telefax: +41 (61) 345 78 92}
\extraC{eMail: info@gccs.com}
\bottomL{Bank: VCT Unterwil, CH--4220 Unterwil/BL}
\bottomR{Account: 322--956123.02R}
```

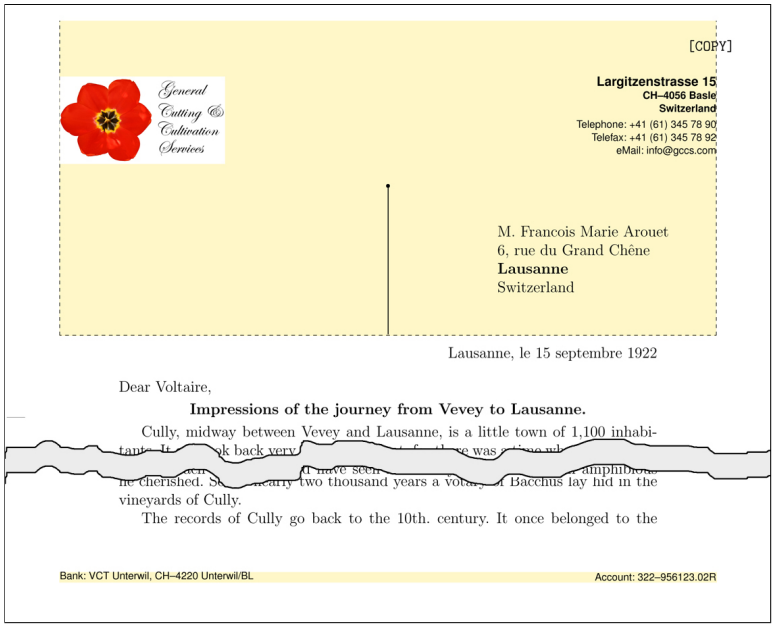


Figure 5: The letter using the defaults.

The truncated output is show in figure 5. The example would be improved if the logo was somewhat larger.

2. Here the commands used were:

```
\documentclass[11pt,german,bruni]{hletter}
\begin{document}
\signature{Dr.~C. Featherstonehaugh &
   Dr.~A. Beauchamp \\ CEO & CIO}
\reference{Impressions of Lausanne}
\date{Lausanne, le 15 septembre 2008}
\begin{letter}{Sir F. Treves, Bart.,\\
               \textbf{Vevey.}\\
               Switzerland}
\opening[\textsc{[draft]}]{Sir,}
...
\closingtwo{Yours Faithfully,}
\vspace{2cm}
\cc{All Smiths in London\\ Mademoiselle S. Curchod}
\encl{Tourist guide to Switzerland.\\ Plan of Cully.}
\vfill
\end{letter}
\end{document}
```

The file `hletg.clo` for the option *bruni* contains:

```
% case = 3 (bruni)
\addressA{Der Glockenturm}
\addressB{Hauptstrasse 54}
\addressC{Upper Throgmortondale}
\extraA{Telefon: +44 187 3546}
\extraB{Telefax: +44 187 3547}
\extraC{email: bruni@songs.flat.ac.uk}
\centreA{Songs written \& sung}
\centreB{Loudness no problem}
\centreC{Flats \& sharps used}
\centreD{\rule[.5ex]{16mm}{1pt}}   % a rule
\centreE{Notes sometimes used}
\centreF{Spears may be hurled}
\centrepos{-10mm}
\bottomL{$\ast\ast\ast\ast\ast$}   % a fancy footer
\bottomC{Lullabies for children aged ... our speciality}
\bottomR{$\ast\ast\ast\ast\ast$}
\sign[10mm]{signat}
\logo[50mm]{Bruennhilde}
\DeclareFixedFont{\newfa}{OT1}{phv}{m}{n}{12pt}
\DeclareFixedFont{\newfc}{OT1}{phv}{m}{sl}{10pt}\or
```

This contained a larger logo, two signees, a rather special footer and it also changed the default fonts `\newfa` and `\newfc`. The font `\newfa` is used for `\addressA` and `\centreA`; `\newfb` is used for address and centre B and C; all the other fields use `\newfc`.

The output is shown in figure 6. The `\sign` command is ignored for two signees.

3. This example is a simple heading for a two column document. The *bruni* option is used again and the document used the commands:

```
\documentclass[11pt,a4paper,twocolumn]{article}
\usepackage[german,bruni]{hhead}
\begin{document}
\setlength{\columnseprule}{.4pt}
\barlength{\textwidth}

\heading[\textsc{confidential}]
```

Note that the commands to specify the header may be placed in the definition file, the *hlet* file or in the document itself. The result is shown in figure 7.

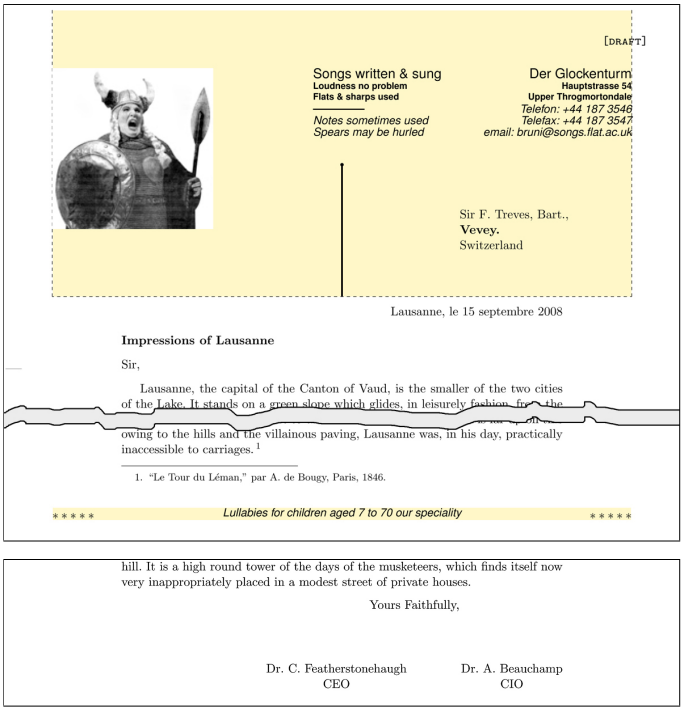


Figure 6: First part of the Bruennhilde letter and the double closing.

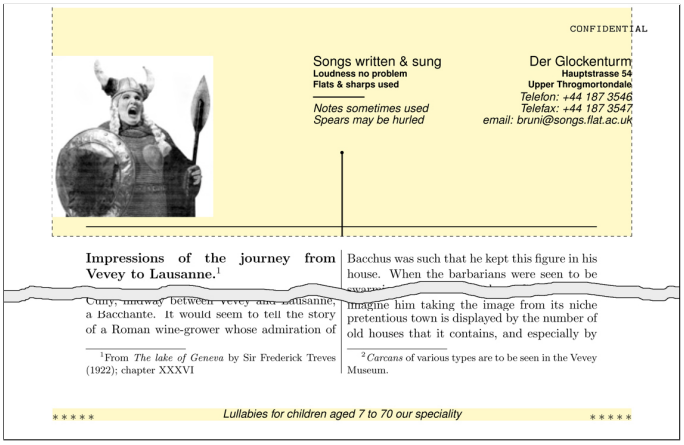


Figure 7: A heading for Bruennhilde.

4. An example of using the slightly modified merge package contains the commands:

```
\documentclass[11pt,english,signit]{hletter}
```

```
\usepackage{mergeh}
\signature{A. Nother\\ Head of Batology Dept.}
\date{Lausanne, le 15 septembre 2008}
\begin{document}
 \reference{Impressions of the journey from
     Vevey to Lausanne}
 \begin{merge}{testmerge.dat}
 between Vevey and Lausanne, is a little town of 1,100
...
 at Cully, but unfortunately the suggestion is unfounded.
 \closing{Yours Sincerely,}

 \vfill
 \cc{All Smiths in London\\ Mademoiselle S. Curchod}
 \encl{Tourist guide to Switzerland.\\ Plan of Cully.}
 \end{merge}
\end{document}
```

and part of the address file is shown below.

```
{Professor Alfred B. Colquhoun\\
   Tittlebat Research Centre\\
   \textbf{Isle of Skye}\\
   Scotland}
Dear Prof.~Colquhoun,
%  old Coony
{Mr.~A. Miller\\
   23a, Council Flats\\
   Park Lane\\
   \textbf{London WC1}}
Dear Archibald,
% first Miller
Dr.~V. M\"{u}ller\\ Langstrasse 15\\ \textbf{3012 Bern}
Dear Vee,
%
%{Mr.~A. Nother\\
%   123 High street\\
%   \textbf{Nether Poppleton}\\
%   Nr. York\\ England}
%Hello Alf,
%% Skip alf today
{Viscountess Elizabeth Featherstonehaught-Cholmondeley\\
   Cathedral Close\\
   \textbf{Winchester}}
```

```
My Dearest Elizabeth,
%
{Sir Archibald Bloggs\\
   Jones Old Yard\\
   Gasworks Lane\\
   \textbf{Throgmortendale}}
Howdy Sir Archie,
%
% NOTE: comments are allowed between the addresses
% but NOT before the first address but NO BLANK LINES!
```

The address of the viscountess will give a *ClassWarning* "** Address too wide for window **".

10 Possible future changes

The first version was called *gletter* (for the company GCCS), *h* was the next letter so maybe a future version will be called *iletter*.

One change which has been suggested is to make the dimensions of the headers easier to specify rather than changing values in the package. Also, the positioning of the text and logo should be more flexible. I really wish to sort out the present confusion in the package between the babel options *english* and *british*. At the moment specifying *english* invokes *british* which is really not correct. The reason for the mix is that english was original used and then it was requested that I also include british — but I was rather lazy!

The support of North American stationery was planned but depends on when and if I acquire samples of the writing materials.

References

[1] Donald E. Knuth, *The tex book*, 15th. ed., Addison-Westley, 1989, ISBN-10: 0201134489.

[2] Leslie Lamport, *Latex: User's guide & reference manual*, 2nd. ed., Addison-Westley, 1994, ISBN-10: 0-201-52983-1.

[3] Graeme McKinstry, *Form letters*, TUGboat **8** (1987), no. 1, 60–61, (Macros revised 6 September 1988).

An XML model of CSS3 as an XeLaTeX-TeXML-HTML5 stylesheet language

S. Sankar, S. Mahalakshmi and L. Ganesh

TNQ Books and Journals
Chennai

Abstract

HTML5[1] and CSS3[2] are popular languages of choice for Web development. However, HTML and CSS are prone to errors and difficult to port, so we propose an XML version of CSS that can be used as a standard for creating stylesheets and templates across different platforms and pagination systems. XeLaTeX[3] and TeXML[4] are some examples of XML that are close in spirit to TeX that can benefit from such an approach. Modern TeX systems like XeTeX and LuaTeX[5] use simplified fontspec macros to create stylesheets and templates. We use XSLT to create mappings from this XML-stylesheet language to fontspec-based TeX templates and also to CSS3. We also provide user-friendly interfaces for the creation of such an XML stylesheet.

Style pattern comparison with Open Office and CSS

Now a days, most of the modern applications have implemented an XML package format that includes an XML implementation of stylesheet: InDesign has its own IDML[6] (InDesign Markup Language) XML package format and MS Word has its own OOXML[7] format, which is another ISO standard format. As they say ironically, the nice thing about standards is that there are plenty of them to choose from. However, instead of creating one more non-standard format, we will be looking to see how we can operate closely with current standards. Below is a sample code derived from OpenOffice document format:

```
<style:style style:name="Heading_20_1"
    style:display-name="Heading 1"
    style:family="paragraph"
    style:parent-style-name="Heading"
    style:next-style-name="Text_20_body"
    style:default-outline-level="1"
    style:class="text">+
<style:font-face style:name="Adobe Caslon Pro Bold"
    svg:font-family="'Adobe Caslon Pro Bold'"
    style:font-family-generic="roman"
    style:font-pitch="variable" />
```

```
<style:text-properties fo:font-size="115%"
    fo:font-weight="bold" style:font-size-asian="115%"
    style:font-weight-asian="bold"
    style:font-size-complex="115%"
    style:font-weight-complex="bold" />
</style:style>
-<style:style style:name="Heading_20_2"
    style:display-name="Heading 2"
    style:family="paragraph"
    style:parent-style-name="Heading"
    style:next-style-name="Text_20_body"
    style:default-outline-level="2"
    style:class="text">
<style:font-face style:name="Arial"
    svg:font-family="Arial"
    style:font-family-generic="swiss"
    style:font-pitch="variable" />
<style:text-properties fo:font-size="14pt"
    fo:font-style="italic" fo:font-weight="bold"
    style:font-size-asian="14pt"
    style:font-style-asian="italic"
    style:font-weight-asian="bold"
    style:font-size-complex="14pt"
    style:font-style-complex="italic"
    style:font-weight-complex="bold" />
</style:style>
```

The equivalent CSS style is listed below:

```
heading1{
        font-family: Adobe Caslon Pro Bold;
        font-size:14pt;
        font-style: normal;
        font-variant: normal;
        font-weight: Bold;
        line-height: 16pt;
        text-align: left;
        color: black;
        background-color: none;
        text-decoration: none;
        text-transform: normal;
        }
```

```
heading2{
        font-family: Arial;
        font-size:14pt;
        font-style: Italic;
        font-variant: normal;
        font-weight: Bold;
        line-height: 16pt;
        text-align: left;
        color: black;
        background-color: none;
        text-decoration: none;
        text-transform: normal;
        }
```

When comparing the above two style coding standards, the Cascading Style Sheet (CSS) is a simple and straightforward formulation without complex namespaces, additional attributes and other details which are not mandatory to form a style.

Advantages of CSS style pattern:
CSS makes it very easy to change the style of a document The useful feature of CSS is that the entire style and layout are abstracted out of the HTML, so the HTML has only the content and not the style aspects. Different stylesheets can be used for different media without even the user knowing it. Stylesheets can be made for different media like the printer and PDA[8]. CSS is considered a clean coding technique, which means search engines won't have to struggle to "read" its content.

Disadvantages of CSS style pattern:
While new additions to CSS3 provide a stronger, more robust feature-set for layout, CSS is still at heart a styling language (for fonts, colours, borders and other decoration), and not a layout language (for blocks with positions, sizes, margins, and so on). These limitations mean that creating fluid layouts generally requires hand-coding of CSS, and has held back the development of a standards-based WYSIWYG[9] editor.

Introducing the Cascading Style Sheet Markup Language (CSSML)

SASS[10] is a meta-language on top of CSS that is used to describe the style of a document cleanly and structurally, with more power than flat CSS. Sass provides a simpler, more elegant syntax for CSS and also implements various features that are useful for creating manageable stylesheets.

SASS (http://sass-lang.com/) is an extension of CSS3 and provides several useful features which can handle nested rules, common variables, etc. However, it is not an XML model and cannot be validated using Schema[11]/DTD[12].

We have introduced the Cascading Style Sheet Markup Language (CSSML), an XML version of CSS that can be used as a standard for creating stylesheets and templates across different platforms and pagination systems. It is also an extension of CSS3 to handle nested rules as in "SASS", and can be validated using Schema/DTD.

Definition of CSSML

The Cascading Style Sheet Markup Language (CSSML) contains the style format details in a well-structured XML format.

The style names used in CSSML are similar to CSS, and the only difference is that the style names are defined as XML tag elements.

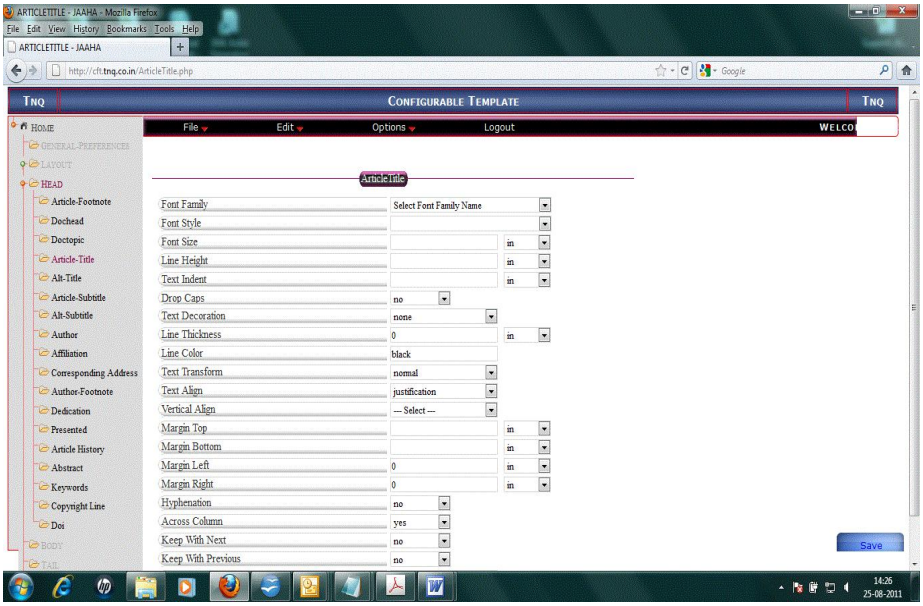
The below example explains the difference between CSS and CSSML style coding:
CSS format: font-family: Adobe Caslon Pro Bold;
CSSML format: <font-family> Adobe Caslon Pro Bold</font-name>

The main advantage of the CSSML tag pattern is that we can validate the CSSML document using XML Schema or XML DTD which is not possible in CSS. We can write our own XML Schema/DTD to validate the CSSML document as follows:

- Elements and attributes that must/may be included, and are permitted in the structure
- The structure as specified by a regular expression syntax
- How character data is to be interpreted, e.g. as a number, a date etc.

However, creating the CSSML document is not as simple as creating a CSS. The CSSML needs XML tagging for all the data, and the user needs to wrap all the details with appropriate XML elements. To avoid such difficulties, we have provided a User Interface to create CSSML automatically with appropriate XML elements.

User Interface to get style specification details:



Sample CSSML coding:

```
<section1>
    <style>
        <font-family>Immono10-regular</font-name>
        <font-url>
            http://www.ctan.org/tex-archive/
            fonts/lm/fonts/opentype/public/lm/
            Immono10-regular
        </font-url>
        <font-size unit="pt">11</font-size>
        <font-style>normal</font-style>
        <font-variant>normal</font-variant>
        <font-weight>Bold</font-weight>
        <font-face>Immono10.OTF</font-name>
        <line_height unit="pt">13.2</line_height>
    </style>
    <section-label id="B12">
        <text-indent unit="pt">6</text-indent>
        <rule-color>black</rule-color>
        <text-transform>normal</text-transform>
        <vertical-align>bottom</vertical-align>
    </section-label>
    <section-title1 id="B13">
        <text-indent unit="in">0</text-indent>
        <text-align>justify</text-align>
        <word-break>hyphenate</word-break>
        <column-span>all</column-span>
        <vertical-align>bottom</vertical-align>
    </section-title1>
</section1>
```

CSSML to CSS3 conversion

CSSML provides a more elegant syntax for CSS and implements various features that are useful for creating stylesheets and LaTeX templates. CSSML allows us to use formatting, nested rules, inline imports, etc., all with CSS compatibility. XSLT[13] is used to transform the CSSML XML to CSS format or LaTeX class file with the use of fontspec package.

Formatting:

```
<paragraph>
 <style>
  <font-name>Times</font-name>
  <font-size unit="pt">11</font-size>
  <line_height unit="pt">13</line_height>
  <text-indent unit="pt">11</text-indent>
 </style>
<paragraph>
```

```
paragraph
{
 font-family: Times;
 font-size: 11pt;
 line-height:13pt;
 text-indent:11pt
}
```

Nesting:

```
<section1>
 <style>
  <font-family>Times</font-name>
  <font-size unit="pt">11</font-size>
  <font-weight>Bold</font-weight>
  <line_height unit="pt">12</line_height>
 </style>
 <section-label id="B12">
   <text-indent unit="pt">6</text-indent>
 </section-label>
 <section-title id="B13">
  <text-align>left</text-align>
 </section-title>
</section1>
```

```
section1
{
 font-family: Times;
 font-size: 11pt;
 line-height:12pt;
 font-weight:bold;
}
section1>section-label
{
 text-indent:6pt;
}
section1>section-title
{
 text-align:left;
}
```

XPath is used to select nodes instead of CSS selectors. A good XPath to CSS mapping is given below:

CSS Selectors	XPath Pattern
h1 p	h1//p (Matches any p element that is a descendant of an h1 element)
h1 > p	h1/p (Matches any p element that is a child of an element h1)
p:first-child	*[1]/self::p (Matches element p when p is the first child of its parent)
h1 + h2	h1/following-sibling::*[1]/self::h2 (Matches any h1 element immediately preceded by an element h2.

CSSML to TEX (FONTSPEC) conversion

We are using "fontspec" package for font definition (fd)[14]. The "fontspec" package allows users of XETEX or LuaTEX to load OpenType fonts in a LATEX document. No font installation is necessary, and font features can be selected and used as desired throughout the document.

XETEX and LuaTEX also allow fonts to be loaded by file name instead of font name. When you have a very large collection of fonts, you will sometimes not wish to have them all installed in your system's font directories. In this case, it is more convenient to load them from a different location on your disk.

Font Declaration:

```
<font-group>
 <font-family>Times</font-family>
 <font-style-1>Times CG</font-style-1>
 <font-style-2>Times-Bold</font-style-2>
 <font-style-3>Times-Italic</font-style-3>
 <font-style-4>Times-BoldItalic</font-style-4>
 <font-style-5>Times-BoldSC</font-style-5>
</font-group>
```

```
\fontspec
[%
BoldFont= Times-Bold.otf ,
ItalicFont= Times-Italic.otf ,
BoldItalicFont = Times-BoldItalic.otf,
SmallCaps=Times-BoldSC.oft,
]{Times.otf}
```

Paragraph Style:

```
<paragraph>
 <style>
  <font-name>Times</font-name>
  <font-size unit="pt">11</font-size>
  <line_height unit="pt">13</line_height>
  <text-indent unit="pt">11</text-indent>
 </style>
<paragraph>
```

```
\def\normalsize{%
\fontsize{11}{13}
\fontspec{Times}
\paraindent=11\p@
}
```

Section Heading Style:

```
<section1>
 <style test="section1">
  <font-family>Times</font-name>
  <font-size unit="pt">11</font-size>
  <line-height unit="pt">13</line-height>
 </style>
 <section-label>
  <style test="section1">
   <font-variant>Bold</font-variant>
  </style>
 </section-label>
 <section-title>
  <style test="section1">
   <text-align>center</text-align>
   <margin-top unit="pt">12</margin-top>
   <margin-bottom unit="pt">6</margin-bottom>
  </style>
 </section-title>
</section1>
```

```
\newcommand\section{\@startsection%
{section}%
{1}%
{\z@}%
{-12\p@ \@plus -2\p@ \@minus -2\p@}
{6\p@}%
{\centering\fontsize{11}{13}%
 \selectfont\bfseries}%
}
```

References

[1] http://www.w3.org/TR/html5/ - A vocabulary and associated APIs for HTML and XHTML.

[2] http://www.css3.info/ - Everything you need to know about CSS3.

[3] http://www.tug.org/mailman/listinfo/xelatex - A DTD/Schema which is very close.

[4] http://getfo.org/texml/ - An XML syntax for TEX.

[5] http://www.luatex.org/ - An extended version of pdfTEXusing Lua as an embedded scripting language.

[6] http://blogs.adobe.com/indesignsdk/2009/03/idml_file_types.html - IDML for representing InDesign content.

[7] http://en.wikipedia.org/wiki/Office_Open_XML - Open Office XML definition.

[8] http://www.webopedia.com/TERM/P/PDA.html - What is PDA? A word definition from the Webopedia.

[9] http://www.webopedia.com/TERM/W/WYSIWYG.html - What is WYSIWYG? – A word definition from the Webopedia.

[10] http://sass-lang.com/ - Syntactically Awesome Stylesheets.

[11] http://www.w3.org/XML/Schema - XML Schemas express shared vocabularies and allow machines to carry out rules made by people.

[12] http://en.wikipedia.org/wiki/Document_Type_Definition - DTD is a set of markup declarations that define a document type for SGML-family markup languages (SGML, XML, HTML).

[13] http://www.w3schools.com/xsl/xsl_intro.asp - XSLT is a language for transforming XML documents into XHTML documents or to other XML documents.

[14] http://www.ctan.org/tex-archive/macros/xetex/latex/fontspec/ - The fontspec package provides an automatic and unified interface for loading fonts in LaTeX.

On the use of TEX as an authoring language for HTML5

S.K.Venkatesan

TNQ Books and Journals
Chennai

Abstract

The TEX syntax has been fairly successful at marking-up a variety of scientific and technical literature, making it an ideal authoring syntax. The brevity of the TEX syntax makes it difficult to create overlapping structures, which in the case of HTML has made life so difficult for XML purists. We discuss S-expressions, the TEX syntax and how it can help reduce the nightmare that the HTML5 markup is going to create. Apart from this we implement a new syntax for marking-up semantic information (microdata) in TEX.

Introduction

The brevity of TEX syntax has made it fairly successful at marking-up a variety of scientific and technical literature. However, modern markup languages such as (X)HTML and XML have verbose syntax which are not only difficult to author but also produce non-tree like structures such as overlapping structures that needs to be checked for well-formedness. However, TEX and its macros are difficult to parse and validate like XML with DTD or Schema. Many XML versions of TEX have been proposed such as TEXML [1] and XELATEX [1] that is intrinsically close to TEX/LATEX. The main advantage with such a system is that one can introduce validator using DTD or Schema that can check the syntax before passing it to the TeX engine. However, XML syntax is difficult to author and in fact is prone to producing overlapping structures that needs to be avoided in order for it to be well-formed, and as a result these XML versions have not become popular.

In this article, we propose something that is quite the reverse, i.e., TeX as a an authoring syntax for both XML and HTML.

TeX, S-expressions and XML

Let us look the following TEX code:

```
\title[lang=en]{Title of a \textit{plain} article}
```

The same code in LISP like S-expression would be:

```
(title (@lang="en") ("Title of a ") (italic "plain") ("article"))
```

The S-expression has the nicety that elements and attributes are treated on par and in fact is an improvement on XML as it allows further nesting within attributes. The corresponding XML code will be:

```
<title lang="en">Title of a <italic>plain</italic> article</title>
```

In both TEX and XML syntax further nesting of structures is not possible within attributes, which make TEX ideal for authoring XML or HTML5.

Overlapping markup in HTML

Since HTML is marked-up by humans, there tends to many situations with overlapping elements and other eccentric markups that do not confirm to a well-formed SGML or XML syntax. Consider the HTML markup:

```
<p>A piece of text with <i>unique <b>and</i> strong
  formatting</b> issues</p>
```

An utility like HTMLTidy [1] or TagSoup [2] can convert it into well formed markup such as:

```
<p>A piece of text with <i>unique </i><b><i>and</i>
    <b> strong formatting</b> issues</p>
```

However it is not clear what should be done with such a non-standard markup. HTML5 specification also defines clearly on how such a non-standard markup should be interpreted [4] but the HTML implementations in browsers currently deal with differently from each other.

W3C has been insisting for some time that the next generation of markup should be XML compliant like XHTML+MathML+SVG profiles, with other intricacies such as namespaces. However, more than 99% of HTML pages in the wild have been invalid according to the HTML4 DTD or Schema. This being the case, W3C gave up on the idea of an XML solution and moved on to HTML5 with added elements and features, such as MathML, SVG and video, audio and addtional microdata formats.

With the experience in HTML4 it can be safely predicted that more features one adds to HTML more the scope for non-standard markup with overlaps and entanglement that can create a great deal of difficulty for different browsers and users.

We will consider here, e.g., Microsoft's own interpretation of MathML in HTML5. Microsoft has been pushing for certain agenda in MathML3 (although much of which has not been accepted by the MathML committee). Based on their own experience with OML, a subset OOXML markup, they would like to add formatting features in

MathML such as bold, italic and paragraph elements inside MathML. Consider the following markup:

```
<math><b><mi>r</mi></b>=<mfenced><mi>x</mi>
  <mi>y</mi></mfenced></math>
```

which could in pure MathML coding would be:

```
<math><mi mathvariant="bold-italic">r</mi></b>=<mfenced>
    <mi>x</mi><mi>y</mi></mfenced></math>
```

Mixing elements from different namespaces is one of the side effects one can expect in HTML5. It is not clear if MathML elements could be included within SVG elements and vice versa. One can expect such new non-standard markups to be created that will be quite difficult for browsers to handle.

New elements such as `<section>` have been introduced, so one can expect more confusion:

```
<section>
  <h2>Section title</h2>
  <section>
    <h1>Another section title</h1>
  </section>
</section>
```

It is not clear from the above markup whether `<h1>` is supposed to have `<h2>` some meaning?

In this article we do not want to convey the impression that everything about HTML5 is wild west, rather, it is a rich arena that needs to be authored carefully, because there are many pit falls. However, HTML5 introduces new features like MathML, SVG, Video and audio features that are quite essential for further enrichment of basic content [3].

TEX as input format for HTML5

In this section we would like introduce LATEX environment for authoring HTML5. Many of these features have been introduced before, say, e.g., in XLATEX and other concepts.

Main structural elements of the document

HTML5 has introduced new content elements that brings it closer to article class. We propose the following TEX macros.

No.	HTML	LaTeX	Description
1	`<article>#1</article>`	`\begin{article} {#1} \end{article}`	Document
			Section Heading
2	`<h1>#1</h1>`	`\Ha{#1}`	– Level One
3	`<h2>#1</h2>`	`\Hb{#1}`	– Level Two
4	`<h3>#1</h3>`	`\Hc{#1}`	– Level Three
4	`<h4>#1</h4>`	`\Hd{#1}`	– Level Four
5	`<p>#1</p>`	`\p{#1}`	Paragraph
6	`<span>#1</span>`	`\s{#1}`	Text Span

Simple formatting elements

We propose the following TEX macros for HTML formatting elements:

No.	HTML	LaTeX	Description
1	`<b>#1</b>`	`\b{#1}`	bold
2	`<i>#1</i>`	`\i{#1}`	italic
3	`<b><i>#1</i></b>`	`\bi{#1}`	bold-italic
4	`<tt>#1</tt>`	`\m{#1}`	monospace
5	`<sup>#1</sup>`	`\sp{#1}`	superscript
6	`<sub>#1</sub>`	`\sb{#1}`	subscript

MathML elements

We propose the following TeX macros for MathML formatting elements:

No.	MathML	LaTeX	Description
1	`<mrow>#1</mrow>`	`{#1}`	grouping
2	`<mi>#1</mi>`	`{#1}`	variables
3	`<mo>#1</mo>`	`{#1}`	operators
4	`<mn>#1</mn>`	`{#1}`	numbers
5	`<mtext>#1</mtext>`	`\mbox{#1}`	monospace
6	`<mfrac>#1#2</mfrac>`	`\frac{#1}{#2}`	fraction
7	`<msup>#1#2</msup>`	`\{#1}^{#2}`	superscript
8	`<msub>#1#2</msub>`	`\{#1}_{#2}`	subscript
9	`<mover>#1#2</mover>`	`\{#1}^{#2}`	over
10	`<munder>#1#2</munder>`	`\{#1}_{#2}`	under

SVG elements

We propose the following TeX macros for SVG formatting elements:

No.	SVG	LaTeX	Description
1	`<circle cx=”#1” cy=”#2” r=”#3” style=”stroke:#4;stroke-width:#5; fill:#6;”/>`	`\circle[x=#1,y=#2,r=#3 s=#4,w=#5,f=#6]`	Circle
2	`<ellipse cx=”#1” cy=”#2” rx=”#3” ry=”#4” style=”stroke:#5; ry=”#4” stroke-width:#6;fill:#7;”/>`	`\ellipse[x=#1,y=#2,rx=#3,ry=#4 s=#5,w=#6,f=#7]`	Ellipse
3	`<rect x=”#1” y=”#2” width=”#3” height=”#4” style=”stroke:#5; stroke-width:#6;fill:#7;”/>`	`\rect[x=#1,y=#2,w=#3,h=#4 s=#5,w=#6,f=#7]`	Rectangle

These elements can be implemented using LaTeX graphics packages such as `TikZ` [4].

Microdata attributes

Since microdata (semantic) attributes can be added to any of the basic HTML elements we need to be able to add attributes to any of the HTML5 TEX macros.

No.	Microdata	LATEX	Description
1	itemscope	`\s[is=on]`	Top element that indicates descendants are in scope
2	itemtype	`\s[it=http://data-vocabulary.org/Person]`	Property URL
3	itemid	`\s[iid=p0312]`	Unique ID of the person
4	itemprop	`\s[ip=name]`	Indicates as the name of the person
5	itemref	`\s[ir=http://www.ctan.org/pub/another-article]`	Refeerence URL

References

[1] Dave Raggett, HTML Tidy, http://tidy.sourceforge.net/

[2] John Cowan, TagSoup: A SAX parser in Java for nasty, ugly HTML, http://home.ccil.org/c̃owan/tagsoup/tagsoup.pdf.

[3] Mark Pilgrim, HTML5: Up and Running, Dive into the Future of Web Development, O'Reilly Media, 2010.

[4] Andrew Mertz and William Slough, Graphics with TikZ, The PracTEX Journal, 2007, No. 1

Towards Evidence-Based Typography: Experiment Design

Boris Veytsman
Computational Materials Science Center, MS 6A2
George Mason University
Fairfax, VA 22030
borisv (at) lk dot net

Leyla Akhmadeeva
Bashkir State Medical University
3 Lenina Str. Ufa, 450000, Russia
la (at) ufaneuro dot org

Typography is both a science and an art with several hundred years of history — or, if we count its ancestor, calligraphy, with several thousand years of history. A beginning typographer faces a large amount of knowledge and rules (see, for example, [1]): for example, that serifed fonts improve readability of body texts, while sans serif is good for advertising and posters; we know the optimal number of words per line and the lines per page, etc. Some of these rules are æsthetic ones, while some are purported reflect the neurophysiology of reading. With respect to the latter ones, we can ask, how do we know what we know?

The situation here may resemble the history of medical science (and art!). Centuries of practical medicine resulted in a vast number of rules and methods of cure. Some of them we now know to be reasonable, like the use of diuretics for lowering blood pressure. Some, like purging, have much narrower applicability than was assumed in the past. Some rules turned out to be ineffective or even harmful, like the unrestrained use of bloodletting. The modern evidence based medicine tries to use a more scientific approach to these rules, putting empirical knowledge in a more formal framework [2].

In this talk we discuss the applicability of evidence based approach to typography. While it is difficult to measure the beauty of the book page, we can measure the readability and the understandability of the text and their dependence on the fonts, type area dimensions and other typographic parameters. This area has been actively developing in the last decade. The modern studies question the widespread notions of the classical typography like the use of serifed fonts [3–5], the mix of minuscule and majuscule letters in body texts [6, 7], text layout [8,9] and other factors [10–12]. This research was stimulated by the challenges presented by new technologies [4, 13–16], the use of type in messages and signage [17–21] and special situations like texts for low vision readers [5, 7, 22], drug information leaflets and other medical data [23–25].

An overwhelming majority of published studies deals with English texts, while there are some works on Arabic [26], Chinese [27,28], Japanese [15,29] and Korean [30] typography. There was no comparable research on Cyrillic scripts and text perception by Russian readers.

Our group works on a a large scale study of the neurophysiology of reading for Russian subjects. We plan to collect a database of readability and understandability as dependent on typographic parameters for Cyrillic texts, In this talk we provide the literature review and discuss the setup of the experiments.

This is a preliminary report of what is envisioned to be an ongoing project. At this stage we need the input of the practitioners of typography and would be grateful for advice and suggestions.

References

[1] Bringhurst, R. *The Elements of Typographic Style*. Hartley & Marks, Publishers, Vancouver, BC, Canada (2004).

[2] Elstein, A. S. *Inflamm. Res.* **53**(Suppl. 2), S184–S189 (2004).

[3] Arditi, A. and Cho, J. *Vision Res.* **45**(23), 2926–2933 (2005).

[4] Bernard, M. L., Chaparro, B. S., Mills, M. M., and Halcomb, C. *Int. J. Hum.-Comput. Stud.* **59**(6), 823–835 (2003).

[5] Russell-Minda, E., Jutai, J. W., Strong, J. G., Campbell, K. A., Gold, D., Pretty, L., and Wilmot, L. *J. Vis. Impair. Blind.* **101**(7), 402–415 (2007).

[6] Sheedy, J. E., Subbaram, M. V., Zimmerman, A. B., and Hayes, J. R. *Hum. Factors* **47**(4), 797–815 (2005).

[7] Arditi, A. and Cho, J. *Vision Res.* **47**(19), 2499–2505 (2007).

[8] Wendt, D. *Vis. Lang.* **16**(1), 88–93 (1982).

[9] dos Santos Lonsdale, M., Dyson, M. C., and Reynolds, L. *J. Res. Read.* **29**(4), 433–453 (2006).

[10] Lewis, C. and Walker, P. *British J. Psychol.* **80**, 241–257 (1989).

[11] Subbaram, M. V., Sheedy, J. E., and Hayes, J. R. *Invest. Ophthalmol. Vis. Sci.* **45**(Suppl. 2), 4354 (2004).

[12] Coronel-Beltran, A. and Alvarez-Borrego, J. *J. Modern Optics* **57**(1), 58–64 (2010).

[13] Dyson, M. C. *Behav. Inf. Technol.* **23**(6), 377–393 (2004).

[14] Lee, D.-S., Shieh, K.-K., Jeng, S.-C., and Shen, I.-H. *Displays* **29**(1), 10–17 (2008).

[15] Hasegawa, S., Fujikake, K., Omori, M., and Miyao, M. *Int. J. Occup. Saf. Ergon.* **14**(3), 293–304 (2008).

[16] Shen, I.-H., Shieh, K.-K., Chao, C.-Y., and Lee, D.-S. *Displays* **30**(2), 53–58 (2009).

[17] Wang, J. H. and Cao, Y. *Int. J. Ind. Eng.-Theory Appl. Pract.* **10**(4), 339–344 (2003).

[18] Garvey, P. M., Chirwa, K. N., Meeker, D. T., Pietrucha, M. T., Zineddin, A. Z., Ghebrial, R. S., and Montalbano, J. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2004*, number 1862 in Transportation Research Record, 1–9. (2004).

[19] Carlson, P. J. and Holick, A. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2005* [31], 26–34.

[20] Ullman, B. R., Ullman, G. L., Dudek, C. L., and Ramirez, E. A. In *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings 2005* [31], 56–62.

[21] Gabbard, J. L., Swan, J. E., and Hix, D. *Presence-Teleoper. Virtual Env.* **15**(1), 16–32 (2006).

[22] Arditi, A. *Ergonomics* **47**(5), 469–482 (2004).

[23] Mackey, M. A. and Metz, M. *Int. J. Consumer Studies* **33**(4), 369–381 (2009).

[24] Chubaty, A., Sadowski, C. A., and Carrie, A. G. *Age & Ageing* **38**(4), 441–447 (2009).

[25] Bix, L., Lockhart, H., Selke, S., Cardoso, F., and Olejnik, M. *Packag. Technol. Sci.* **16**(5), 199–207 (2003).

[26] Al-Harkan, I. M. and Ramadan, M. Z. *Int. J. Ind. Ergon.* **35**(7), 652–664 (2005).

[27] Huang, D.-L., Rau, P.-L. P., and Liu, Y. *Int. J. Ind. Ergonomics* **39**(1), 81–89 (2009).

[28] Cai, D. C., Chi, C. F., and You, M. L. *Int. J. Ind. Ergon.* **27**(1), 9–17 (2001).

[29] Ayama, M., Ujike, H., Iwai, W., Funakawa, M., and Okajima, K. *Opt. Rev.* **14**(1), 48–56 (2007).

[30] Kong, Y.-K., Lee, I., Jung, M.-C., and Song, Y.-W. *Ergonomics* **54**(5), 453–465 (2011).

[31] National Research Council (U. S.). Transportation Research Board and National Research Council (U. S.) Transportation Research Board Meeting. *Traffic Control Devices, Visibility, and Rail-Highway Grade Crossings, 2005*. Number 1918 in Transportation research record. National Academy Press (2005).