

blog.sty

Generating HTML Quickly with T_EX*

Uwe Lück[†]

October 10, 2011

Abstract

`blog.sty` provides T_EX macros for generating web pages, based on processing text files using the `fifinddo` package. Some L^AT_EX commands are redefined to access their HTML equivalents, other new macro names “quote” the names of HTML elements. The package has evolved in several little steps each aiming at getting pretty-looking “hypertext” **notes** with little effort, where “little effort” also has meant avoiding studying documentation of similar packages already existing. [TODO: list them!] The package “*misuses*” T_EX’s macro language for generating HTML code and entirely *ignores* T_EX’s typesetting capabilities.

Contents

1	Installing and Usage	3
2	Examples	4
2.1	A Very Plain Style	4
2.1.1	Driver File <code>makehtml.tex</code>	4
2.1.2	Source File <code>texmap.tex</code>	5
2.2	A Style with a Navigation Column	5
2.2.1	Driver File <code>makehtml.tex</code>	6
2.2.2	Source File <code>schreibt.tex</code>	7
3	The File <code>blog.sty</code>	8
3.1	Package File Header (Legalize)	8
3.2	Processing	8
3.3	General HTML Matters	10
3.3.1	General Tagging	10

*This document describes version [v0.5](#) of `blog.sty` as of 2011/10/10.

[†]<http://contact-ednotes.sty.de.vu>

3.3.2	Attributes	10
3.3.3	HTML's Special Symbols	12
3.3.4	Head	12
3.3.5	Body	13
3.4	Fonts	13
3.5	Logical Markup	14
3.6	Environments	15
3.7	Links	16
3.7.1	Basic Link Macros	16
3.7.2	Special cases of Basic Link Macros	16
3.7.3	Italic Variants	17
3.7.4	Built Macros for Links to Local Files	17
3.7.5	Built Macros for Links to Remote Files	17
3.8	Characters/Symbols	18
3.8.1	Basic Preliminaries	18
3.8.2	Diacritics	19
3.8.3	Greek	19
3.8.4	Arrows	20
3.8.5	Dashes	20
3.8.6	Spaces	20
3.8.7	Quotes, Apostrophe, Prime	21
3.8.8	Math	22
3.8.9	Other	23
3.9	TeX-related	24
3.9.1	Logos	24
3.9.2	Describing Macros	24
3.10	Tables	25
3.10.1	Indenting	25
3.10.2	Starting/Ending Tables	25
3.10.3	Rows	25
3.10.4	Cells	26
3.10.5	Filling a Row with Dummy Cells	26
3.11	Misc	27
3.12	The End	28
3.13	VERSION HISTORY	28
4	Real Web Pages with Inavicol.sty	30
4.1	blog.sty Required	30
4.2	Switches	30
4.3	Page Style Settings (to be set locally)	30
4.4	Possible Additions to blog.sty	31
4.4.1	Tables	31
4.4.2	Graphics	31
4.4.3	HTTP/Wikipedia tooltips	32
4.5	Page Structure	33
4.5.1	Page Head Row	33

4.5.2	Navigation and Main Row	34
4.5.3	Footer Row	34
4.6	The End and HISTORY	35
5	Beamer Presentations with the <code>blogdot</code> Package	35
5.1	Overview	35
5.2	File Header	37
5.3	<code>blog</code> Required	37
5.4	Size Parameters	37
5.5	(Backbone for) Starting a “Slide”	38
5.6	Finishing a “Slide” and “Restart” (Backbone)	39
5.7	Moving to Next “Slide” (User Level)	40
5.8	Constructs for Type Area	40
5.9	Test for <code>blogdot.cfg</code>	41
5.10	The End	41
5.11	VERSION HISTORY	41

1 Installing and Usage

The file `blog.sty` is provided ready, **installation** only requires putting it somewhere where \TeX finds it (which may need updating the filename data base).¹

User commands are described near their implementation below.

However, we must present an **outline** of the procedure for generating HTML files:

At least one **driver** file and one **source** file are needed.

The **driver** file’s name is stored in `\jobname`. It loads `blog.sty` by

```
\RequirePackage{blog}
```

and uses file handling commands from `blog.sty` and `fifinddo` (cf. `mdoccheat.pdf` from the `nicetext` bundle). It chooses **source** files and the name(s) for the resulting HTML file(s). It may also need to load local settings, such as for the language (`lang-de.fdf`, `lang-en.fdf`), and settings for converting the editor’s text encoding into the encoding that the head of the resulting HTML file advertises (`atari.fdf` in the `nicetext` bundle).

The driver file could be run a terminal dialogue in order to choose source and target files and settings. So far, I rather have programmed a dialogue just for converting UTF-8 into an encoding that my Atari editor `xEDIT` can deal with. I do not present this now because it was conceptually mistaken, I must set up this conversion from scratch some time.

The **source** file(s) should contain user commands defined below to generate the necessary `<head>` section and the `<body>` tags.

¹<http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf>

2 Examples

2.1 A Very Plain Style

My “ \TeX -generated pages”² use a **driver** file `makehtml.tex`. To choose a page to generate, I “uncomment”ed just one of several lines that set the “current conversion job” from a list (for some time). I choose the example of a simple “site map:” `texmap.htm` is generated from **source** file `texmap.tex`.—More recently however, I have started to read the job name and perhaps extra settings from a file `jobname.tex` that is created by a Bash script.

In order to make it easier for the reader to see what is essential, I have moved many `.cfg`-like extra definitions into a file `texblog.fdf`. Some of these definitions may later move into `blog.sty`. You should find `makehtml.tex`, `texmap.tex`, and `texblog.fdf` in a directory `demo/texblog` (or `texblog.fdf` may be together with the `.sty` files), perhaps you can use them as templates.

2.1.1 Driver File `makehtml.tex`

```

1  \def \GenDate {2011/10/09}
   \ProvidesFile{makehtml.tex}[\GenDate\space HTML driver]
   \RequirePackage{blog,texlinks}
   \input{atari.fdf}\input{lang-en.fdf}
5  \input{texblog.fdf}          %% 2011/09/24
   %%%%%%%%%%%
   \input{jobname}              %% write with "echo"
   % \def \htmljob
   %%%%%%%%%%%
10 % {texmap}
   % {heyctan}                  %% \def\pkg{\privpkgnamefmt}
   % {makeshow}
   % {texhax}
   % {aaoe1550}
15 % {jobs}                      %% \BlogAutoPars
   %%%%%%%%%%%
   % {beobacht} \input{lang-de.fdf}
   % {gtd}       \input{lang-de.fdf}
   % {MV45}      \input{mv45.fdf}
20 % {STEUER}    \input{lang-de.fdf}
   % {GALLEY}    \input{lang-de.fdf}
   %%%%%%%%%%%
   \ResultFile{\htmljob\htmakeext}
   \BlogCopyFile[\TextCodes
25       \MakeActiveDef\"{\catchdq}%
       ]{\htmljob.tex}
   \CloseResultFile
   \stop

```

²www.webdesign-bu.de/uwe_lueck/texmap.htm

2.1.2 Source File texmap.tex

```

1  \ProvidesFile{texmap.tex}[2011/09/29 TeX-generated: overview]
    %% <- for blog/myfilist.sty 2011/02/22
    \comment{ 2011/01/25 \string\endash\ -> \string\pardash\ }
    \comment{ 2010/12/05 \string\emdash\ -> \string\endash\ }
5  \head \charset{ISO-8859-1} %%% {utf-8}
    \texrobots
    \texstylesheet
    \title{TeX-generated pages - U. L.}
    \body \textopofpage
10 \heading1{Uwe Lück's \TeX-generated/related pages}
    % \emdash\,I'm playing with a different style of pages here.
    % \hrule\ \ %%% \endgraf
    The present page leads you to:

15 \begin{enumerate} %% '\href' 2011/08/18:
    \item \href{index.html}{\file{index}}\pardash my English main page
    \item \href{schreibt.html}{\file{schreibt}}\pardash my German main page
    \hrule
    \item \Fileref{aaoe1550}\pardash UMTS stick with Linux netbook
20 \item \Fileref{heyctan}\pardash CTAN discoveries
    \item \Fileref{jobs}\pardash coaching %% explained 2010/09/24
    \item \Fileref{makeshow}\pardash \TeX\ almost WYSIWYG \dots
    \item \Fileref{texhax}\pardash studies on texhax postings
    \hrule
25 % \item \Fileref{beobacht}\endash\ \dedqtd{Web-Tagebuch}
    %% <- rm. 2011/09/29
    \item \Fileref{gtd}\endash\ \dedqtd{Getting Things Done}
    \item \Fileref{MV45}\endash\ \HTML-Prsentation von
        \ctanpkgref{nicetext} und \CtanPkgRef{morehype}{blog.sty}
30 \item \Fileref{oeffnotz}\endash\ \dedqtd{öffentliche Notizen} %% 2011/08/16b
\end{enumerate}

    \hrule
    \enlastrev
35 \entotopofpage
    \fivebreaks \fivebreaks
    \finish

```

2.2 A Style with a Navigation Column

A style of web pages looking more professional than `texmap.htm` (while perhaps becoming outdated) has a small navigation column on the left, side by side with a column for the main content. Both columns are spanned by a header section above and a footer section below. The package `lnavicol.sty` provides commands

`\PAGEHEAD`, `\PAGENAVI`, `\PAGEMAIN`, `\PAGEFOOT`, `\PAGEEND` (and some more) for structuring the source so that the code following `\PAGEHEAD` generates the header, the code following `\PAGENAVI` forms the content of the navigation column, etc. Its code is presented in Sec. 4. For real professionalism, somebody must add some fine CSS, and the macros mentioned may need to be redefined to use the `@class` attribute. Also, I am not sure about the table macros in `blog.sty`, so much may change later.

With things like these, can `blog.sty` become a part of a “content management system” for \TeX addicts? This idea rather is based on the *German* Wikipedia article.

As an example, I present parts of the source for my “home page”³. As the footer is the same on all pages of this style, it is added in the driver file `makehtml.tex`. `schreibt.tex` is the source file for generating `schreibt.html`. You should find *this* `makehtml.tex`, a cut down version of `schreibt.tex`, and `writings.fdf` with my extra macros for these pages in a directory `demo/writings`, hopefully useful as templates.

2.2.1 Driver File `makehtml.tex`

```

1  \def \GenDate {2011/10/05}
    \ProvidesFile{makehtml.tex}
        [\GenDate\space TeX engine for "writings"]
    \RequirePackage{blog,texlinks,lnavicol}      %% 2011/09/02
5  \input{atari.fdf} \input{lang-en.fdf}
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    \input{jobname}          %% 2011/09/24
    % \def \htmljob
    % {_sitemap}
10  % {index}                  \BlogAutoPars
    % {schreibt} \input{lang-de.fdf} \BlogAutoPars
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % {about}                  \BlogAutoPars
    % {contact}                % \tighttrue
15  % {kontakt} \input{lang-de.fdf} % \tighttrue
    % {tutor} \input{lang-de.fdf} \BlogAutoPars \deepttrue
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % {writings} \BlogAutoPars \deepttrue
    % {repres} \BlogAutoPars \deepttrue
20  % {critedl} \BlogAutoPars \deepttrue
    % {ednworks} \BlogAutoPars
    % {public} \BlogAutoPars \deepttrue
    % {texproj} \BlogAutoPars % \deepttrue
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25  \input{writings.fdf}

```

³www.webdesign-bu.de/uwe_lueck/schreibt.html

```

\ResultFile{\htmljob\htmakeext}
\WriteResult\writdoctype           %% TODO
\BlogCopyFile[\TextCodes
30   \MakeActiveDef\"{\catchdq}%    %% TODO attributes!?
      ]{\htmljob.tex}
\WriteResult{\PAGEFOOT}
\WriteResult{\indentii\rainermaster}
\WriteResult{\indentii\}
35 \WriteResult{\indentii\ueberseeport}
\WriteResult{\PAGEEND}
\ifdeep \WriteResult{\indentii\vspace{280}} \fi
\WriteResult{\finish}
\CloseResultFile
40 \stop

```

2.2.2 Source File schreibt.tex

```

1 \ProvidesFile{schreibt.tex}[2011/08/19 f. schreibt.html]
  \head \charset{ISO-8859-1}
    \writrobots
    \writstylesheets
5 \title{\Uwe\ schreibt} \body \writtopofpage
  \PAGEHEAD
    \headuseskiptitle{%
      \timecontimgref{writings}{0}{Zeit-Logo}{Russells Zeit}%
    }{10}{\Uwe\ \dqt{schreibt}}
10 \PAGENAVI
    \fileitem{writings}{Intervallordnungen (Mathematik~etc.)}
    \fileitem{public}{Publikationen}
    \hrule
    \fileitem{critedltx}{Softwarepakete f\"ur kritische Editionen}
15 \fileitem{texproj}{TeX-Projekte} %%% Makro-Projekte}
    \hrule
    \fileitem{tutor}{Mathe-Tutor}
    \indentii\item\href{texmap.htm}{Notizen}
    \hrule
20 \deFIabout \deFIkontakt
  \PAGEMAIN
    \strong{Wissenschaft:}\enspace Diese Seiten entstanden zuerst
    zur Presentation zweier ETC.

25 \rightpar{\textit{Worms-Pfeditersheim, den 19.~August 2011,\\\Uwe}}
  % \rightpar{\textit{Munchen, den 31.~Juli 2011,\\\Uwe}}
  %% <- TODO VERSION

```

3 The File blog.sty

3.1 Package File Header (Legalize)

```

1  \NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
2  \ProvidesPackage{blog}[2011/10/10 v0.5 simple fast HTML (UL)]
3  %% copyright (C) 2010 2011 Uwe Lueck,
4  %% http://www.contact-ednotes.sty.de.vu
5  %% -- author-maintained in the sense of LPPL below.
6  %%
7  %% This file can be redistributed and/or modified under
8  %% the terms of the LaTeX Project Public License; either
9  %% version 1.3c of the License, or any later version.
10 %% The latest version of this license is in
11 %%      http://www.latex-project.org/lppl.txt
12 %% We did our best to help you, but there is NO WARRANTY.
13 %%
14 %% Please report bugs, problems, and suggestions via
15 %%
16 %%      http://www.contact-ednotes.sty.de.vu
17 %%

```

3.2 Processing

We are building on the `fifinddo` package:

```

18  \RequirePackage{fifinddo}

```

`\CLBrk` is a *code line break* (also saving subsequent comment mark):

```

19  \newcommand*{\CLBrk}{^^J}

```

`\htmakeext` is the extension of the generated file. Typically it should be `.html`, as set here, but my Atari emulator needs `.htm` (see `texblog.fdf`):

```

20  \newcommand*{\htmakeext}{.html}

```

`\BlogCopyFile[<changes>]{<src-file>}` “copies” the T_EX source file *<src-file>* into the file specified by `\ResultFile`. As in T_EX an empty line starts a new paragraph, we “interpret” an empty source line as HTML tag `<p>` for starting a new paragraph. Empty source lines following some first empty source line immediately are ignored (“compression” of empty lines).—However, I am not entirely sure that this won’t have unwanted effects, so it must be required explicitly by `\BlogAutoPars`, or by calling the package with option `[autopars]`. In the latter case, it can be turned off by `\noBlogAutoPars`

```

21  \newif\ifBlogAutoPars
22  \newcommand*{\BlogAutoPars}{\BlogAutoParstrue}
23  \newcommand*{\noBlogAutoPars}{\BlogAutoParsfalse}
24  \DeclareOption{autopars}{\BlogAutoPars}
25  \ProcessOptions

```



```

26 \MakeOther\< \MakeOther\>          %% TODO ...
27 \newcommand*{\BlogCopyFile}[2][{}]{%
28     \typeout{^^J\screenqtd{blog.sty} generating          %% 2011/10/05
29         \screenqtd{\htmljob\htmakeext}}}%
30     \ProcessFileWith[\BlogCodes
31         \let\ProvidesFile\BlogProvidesFile %% 2011/02/24
32         \let\protect\@empty                %% 2011/03/24
33         #1]{#2}{%
34         \IfFDinputEmpty
35             {\IfFDpreviousInputEmpty
36                 \relax
37                 {\WriteResult{\ifBlogAutoPars<p>\fi}}}%
38         \CopyLine
39     }%
40 }

```

For a while, line endings swallowed inter-word spaces, until I found the setting of `\endlinechar` (fifinddo’s default is -1) in `\BlogCodes`:

```

41 \newcommand*{\BlogCodes}{%          %% 2010/09/07
42     \endlinechar'\ \catcode'\~\active \BasicNormalCatCodes}

```

The tilde is active as in Plain TeX too, it is so natural to use it for abbreviating HTML’s ` `;

`\ProvidesFile{<file-name>.tex}[<file-info>]` is supported for use with the `myfilist` package to get a list of source file infos. In generating the HTML file, the file infos are transformed into an HTML comment. Actually it is `\BlogProvidesFile` (for the time being, 2011/02/22):

```

43 \@ifdefinable\BlogProvidesFile{%
44     \def\BlogProvidesFile#1[#2]{%
45         \comment{ generated from\CLBrk\CLBrk
46             \ \ \ \ \ \ \ \ #1, #2,\CLBrk\CLBrk
47             \ \ \ \ \ with blog.sty,
48             \GenDate\ }}          %% TODO predefine?

```

([TODO](#): customizable style.)—Due to the limitations of the approach reading the source file line by line, the “optional argument” `[<file-info>]` of `\ProvidesFile` must appear in the same line as the closing brace of its mandatory argument. The feature may require inserting

```
\let\ProvidesFile\BlogProvidesFile
```

somewhere, e.g., in `\BlogCopyFile`.

3.3 General HTML Matters

The following stuff is required for any web page (or hardly evitable).

3.3.1 General Tagging

```
\TagSurr{<el-name>}{<attr>}{<content>}
```

(I hoped this way code would be more readable than with `\TagSurround ...`)
and

```
\SimpleTagSurr{<el-name>}{<content>}
```

are used to avoid repeating element names `<el-name>` in definitions of \TeX macros that refer to “entire” elements—as opposed to elements whose content often spans lines (as readable HTML code). We will handle the latter kind of elements using L^AT_EX’s idea of “environments.” `\TagSurr` also inserts specifications of element **attributes**, [TODO: `wiki.sty` syntax would be so nice here] while `\SimpleTagSurr` is for elements used without specifying attributes. `\STS` is an abbreviation for `\SimpleTagSurr` that is useful as the `\SimpleTagSurr` function occurs so frequently:

```
49 \newcommand*{\SimpleTagSurr}[2]{<#1>#2</#1>}
50 \newcommand*{\STS}{\let\STS\SimpleTagSurr %% 2010/05/23
51 \newcommand*{\TagSurr}[3]{<#1 #2>#3</#1>}
```

3.3.2 Attributes

Inspired by the common way to use `@` for referring to element attributes—i.e., `@<attr>` refers to attribute `<attr>`—in HTML/XML documentation, we often use

`\@<attr>{<value>}` to “abbreviate” `<attr>="<value>"`

within the starting tag of an HTML element. This does not really make typing easier or improve readability, it rather saves \TeX ’s memory by using a single token for referring to an attribute. This “abbreviation” is declared by `\declareHTMLattrib{<attr>}`, even with a check whether `\@<attr>` has been defined before:

```
52 \newcommand*{\declareHTMLattrib}[1]{%
53   \def\reserved@a{#1}%
54   \ifundefined{#1}%
55     {\@namedef{#1}##1{#1="##1"}}%
56     \@notdefinable}
```

So after `\declareHTMLattrib{<attr>}`, `\@<attr>` is a \TeX macro expecting one parameter for the specification.

A few frequent attributes are declared this way here. `@class`, `@id`, `@style`, `@title`, `@lang`, and `@dir` are the ones named on *Wikipedia*:

```

57 \let\@class\relax      %% for tab/arr in latex.ltx
58 \declareHTMLattrib{class}
59 \declareHTMLattrib{id}
60 \declareHTMLattrib{style}
61 \let\@title\relax      %% for \title in latex.ltx
62 \declareHTMLattrib{title}          %% 2011/04/26
63 \declareHTMLattrib{lang}
64 \declareHTMLattrib{dir}

```

`@type` is quite frequent too:

```

65 \declareHTMLattrib{type}

```

`@href` is most important for that “hyper-text.”

```

66 \declareHTMLattrib{href}

```

... and `@name` (among other uses) is needed for `@name` for hyper-text anchors:

```

67 \declareHTMLattrib{name}          %% 2010/11/06

```

`@bgcolor` is used in tables as well as for the appearance of the entire page:

```

68 \declareHTMLattrib{bgcolor}

```

Of course, conflicts may occur, as the form `\@(<ASCII-chars>)` of macro names is used for internal (La)TeX macros. Indeed, `\@width` that we want to have for the `@width` attribute already “abbreviates” TeX’s “keyword” (TeXbook p. 61) `width` in L^AT_EX (for specifying the width of a `\hrule` or `\vrule` from TeX; again just saving TeX tokens rather than for readability).

```

69 \PackageWarning{blog}{Redefining \protect\@width}
70 \let\@width\relax
71 \declareHTMLattrib{width}

```

Same with `@height`:

```

72 \PackageWarning{blog}{Redefining \protect\@height}
73 \let\@height\relax
74 \declareHTMLattrib{height}        %% 2010/07/24

```

We can enumerate the specifications allowed for `@align`:

```

75 \newcommand*{\@align@c}{\@align{center}}
76 \newcommand*{\@align@l}{\@align{left}}
77 \newcommand*{\@align@r}{\@align{right}}
78 \newcommand*{\@align}[1]{align="#1"}

```

`@valign@t`:

```

79 \newcommand*{\@valign@t}{\v@align{top}} %% 2011/04/24

```

Some other uses of `\declareHTMLattrib` essential for *tables*:

```

80 \declareHTMLattrib{border}          %% 2011/04/24
81 \declareHTMLattrib{cellpadding}    %% 2010/07/18
82 \declareHTMLattrib{cellspacing}    %% 2010/07/18
83 \declareHTMLattrib{colspan}        %% 2010/07/17
84 \declareHTMLattrib{frame}          %% 2010/07/24

```

Another problem with this namespace idea is that *either* this reference to attributes cannot be used in “author” source files for generating HTML—or `@` cannot be used for “private” (internal) macros. Cf. `\ContentAtt` for `<meta>` tags ... well, not so bad, as the main purpose of this namespace is saving tokens *in macros*.

3.3.3 HTML’s Special Symbols

`#` is needed for numerical specifications in HTML, especially colors and Unicode symbols, while it plays a different (essential) role in our definitions of `TEX` macros here. We redefine `LATEX`’s `\#` for a kind of “quoting” `#` in macro definitions in order to refer to their HTML meaning.—I **wonder** what I had in mind with the `&` things here. I cannot find any use of `\AmpMark` in my code (including my web pages). There is no real problem with calling special HTML symbols, `&` is simply made **other** already here for macros calling those symbols (below), and in processing source files, it is as well **other** by default. The symbols section, however, redefines `\&` for calling HTML’s ampersand symbol.

```

85 {\catcode'\&=12 \catcode'\#=12
86 \gdef\AmpMark{&} \gdef\#{#}}

... \CompWordMark etc.?

```

3.3.4 Head

`\head` produces the first two tags that an HTML file must start:

```

87 \newcommand*{\head}{<html><head>} %% ^^J rm 2010/10/10

```

`\MetaTag{inside}` and `\ContentAtt{value}` are internal shortcuts:

```

88 \newcommand*{\MetaTag}[1]{\space\space<meta #1>}
89 \newcommand*{\ContentAtt}[1]{content="#1"}

```

`\charset{code-page}`

```

90 \newcommand*{\charset}[1]{%
91 \MetaTag{http-equiv="Content-Type" \ContentAtt{text/html; #1}}

```

`\description{text}` for web searches (I think):

```

92 % \newcommand*{\description}[1]{%
93 % \MetaTag{\@name{description} \ContentAtt{#1}}

```

... an outright **mistake!** The definition is overridden to get the HTML equivalent to L^AT_EX's `\description` environment. `\newcommand` did not warn here because we don't load any L^AT_EX class for text-processing macros and code generation.— And so some `\MetaDescr` should be defined—and used finally! **TODO**

`\robots{⟨instructions⟩}`:

```
94 \newcommand*{\robots}[1]{%% juergenf: index, follow, noarchive
95   \MetaTag{\@name{robots} \ContentAtt{#1}}}
```

`\norobots` for privacy:

```
96 \newcommand*{\norobots}{\robots{noarchive,nofollow,noindex}}
```

`\stylesheet{⟨media⟩}{⟨css⟩}` uses `⟨css⟩.css` for `media="⟨media⟩"`:

```
97 \newcommand*{\stylesheet}[2]{%
98   \space\space                               %% 2010/09/10
99   <link rel="stylesheet" media="#1"
100         \@type{text/css}                      %% \@type 2011/10/05
101         \@href{#2.css}>>}
```

Alternatively, style declarations may occur in the `<style>` element. It can be accessed by the `{\style}` environment (cf. Sec. 3.6):

```
102 \newenvironment*{\style}[1]
103       {<style \@type{text/css} media="#1">}
104       {</style>}
```

With `\title{⟨text⟩}`, `⟨text⟩` heads the browser window:

```
105 \renewcommand*{\title}[1]{\space\space<title>#1</title>}
```

3.3.5 Body

`\body` separates the `head` element from the `body` element of the page.

```
106 \newcommand*{\body}{</head><body>}
```

`\topofpage` generates an anchor `top-of-page`:

```
107 \newcommand*{\topofpage}{\hanc{top-of-page}{}}
```

`\finish` finishes the page, closing the `body` and `html` elements.

```
108 \newcommand*{\finish}{</body></html>}
```

3.4 Fonts

`\heading{⟨level⟩}{⟨text⟩}` prints `⟨text⟩` with size dependent on `⟨level⟩`. The latter may be one out of 1, 2, 3, 4, 5, 6.

```
109 \newcommand*{\heading}[1]{\SimpleTagSurr{h#1}}
```

... I might use `\section` etc. one day, I made `\heading` when I could not control the sizes of the section titles properly and decided first to experiment with the level numbers.

We “re-use” some \LaTeX commands for specifying font attributes, rather than (re)defining macros `\i`, `\b`, `\tt`, ...

`\textit{\langle text \rangle}` just expands to `<i>\langle text \rangle</i>`

```
110 \renewcommand*\textit{\SimpleTagSurr i}
```

etc. for `\textbf`, `\texttt` ...:

```
111 \renewcommand*\textbf{\SimpleTagSurr b}
```

```
112 \renewcommand*\texttt{\SimpleTagSurr tt}      %% 2010/06/07
```

`\textsf{\langle text \rangle}` [TODO](#) (see some `makehtml.tex`)

`\textcolor` is from \LaTeX ’s color package that we won’t load for generating HTML, so it is “new” here, it is just natural to use it for colored text (2010/05/15):

```
113 \newcommand*\textcolor[1]{\TagSurr{font}{color="#1"}}
```

3.5 Logical Markup

`\code{\langle text \rangle}` marks `\langle text \rangle` as “code,” just accessing the `<code>` element, while standard \LaTeX does not provide a `\code` command:

```
114 \newcommand*\code{\SimpleTagSurr{code}}      %% 2010/04/27
```

`\emph{\langle text \rangle}` is \LaTeX ’s command again, but somewhat abused, expanding to `\langle text \rangle`:

```
115 \renewcommand*\emph{\SimpleTagSurr{em}}
```

... Note that \LaTeX ’s `\emph` feature of switching to up when `\emph` appears in an italic context doesn’t work here ...

`\strong{\langle text \rangle}` again just calls an HTML element. It may behave like `\textbf{\langle text \rangle}`, or ... I don’t know ...

```
116 \newcommand*\strong{\SimpleTagSurr{strong}}
```

`\var{\langle symbol(s) \rangle}` accesses the `<var>` element:

```
117 \newcommand*\var{\SimpleTagSurr{var}}
```

`\acronym{\langle chars \rangle}` ... there is `\acro` in the TUGboat macros, while HTML provides an `<acronym>` environment ...

```
118 \newcommand*\acronym{\SimpleTagSurr{acronym}}
```

3.6 Environments

We reduce L^AT_EX's `\begin` and `\end` to their most primitive core.

`\begin{<command>}` just executes the macro `\<command>`, and

`\end{<command>}` just executes the macro `\end{<command>}`.

They don't constitute a group with local settings. Indeed, the present (2010/11/07) version of `blog.sty` does not allow any assignments while “copying” the T_EX source into the `.htm`. There even is no check for proper nesting. `\begin` and `\end` just represent HTML elements (their starting/ending tags) that typically have “long” content. (We might “intercept” `\begin` and `\end` before copying for executing some assignments in a future version.)

```
119 \let\begin\@nameuse
120 \def\end#1{\csname end#1\endcsname}
```

... moving `{english}` to `xmlprint.cfg` 2010/05/22 ...

As formerly with “fonts,” we have *two* policies for **choosing macro names**: (i) using an *existing* HTML element name, (ii) using a L^AT_EX command name for accessing a somewhat similar HTML element having a *different* name. [2011/10/05: so what? [TODO](#)]

New 2011/10/05: With `\useHTMLelement{<ltx-env>}{<html-el>}`, you can access the `<html-el>` element by the `<ltx-env>` environment. The “starred” form is for “list” environments where I observed around 2011/10/01 that certain links (with Mozilla Firefox) need ``:

```
121 \newcommand*{\useHTMLelement}{%
122   \ifstar{\@useHTMLelement[</li>]}{\@useHTMLelement}}
123 \newcommand*{\@useHTMLelement}[3][{}]{%
124   \@namedef{#2}{<#3>}%
125   \@namedef{end#2}{#1</#3>}}
```

Applications:

CARE: `{small}` is an environment here, it is not in L^AT_EX:

```
126 \useHTMLelement{small}{small}
```

`{center}`:

```
127 % \renewenvironment*{center}{<p align="center">}{</p>}
128 % \renewenvironment*{center}{<p \@align@c>}{</p>}
129 \useHTMLelement{center}{center}
```

The next definitions for `{enumerate}`, `{itemize}`, `{verbatim}` follow policy (ii):

```
130 \useHTMLelement*{enumerate}{ol}
131 \useHTMLelement*{itemize}{ul}
```

With `blog.sty`, `\verbatim` really doesn't work much like its original L^AT_EX variant. T_EX macros inside still are expanded, and you must care yourself for wanted “quoting”:

```
132 \useHTMLElement{verbatim}{pre}
```

```
{quote}:
```

```
133 \useHTMLElement{quote}{blockquote}
```

For list `\item`s, I tried to get readable HTML code using `\indenti`. This fails with nested lists. The indent could be increased for nested lists if we supported assignments with `\begin` and `\end`. 2011/10/04 including ``, repairs more links in DANTE talk:

```
134 \renewcommand*{\item}{\indenti<li>}
```

L^AT_EX's `\description` environment redefines the label format for the optional argument of `\item`. Again, *we* cannot do this here (we even cannot use optional arguments, at least not easily). Instead we define a different `\ditem{term}` having a *mandatory* argument (TODO star?).

```
135 \useHTMLElement{description}{dl}
```

```
136 \newcommand*{\ditem}[1]{\indenti<dt>\strong{#1}<dd>}
```

3.7 Links

3.7.1 Basic Link Macros

`\hanc{<id>}{<text>}` makes `<text>` an anchor with HTML label `<id>` (like `hyperref`'s `\hypertarget`):

```
137 \newcommand*{\hanc}[1]{\TagSurr a{\@name{#1}}}
```

`\hancref{<id>}{<url>}{<text>}` makes `<text>` an anchor with HTML label `<id>` and at the same time a link to `<url>`:

```
138 \newcommand*{\hancref}[2]{\TagSurr a{\@name{#1}} \@href{#2}}
```

`\href{<id>}{<text>}` makes `<text>` a link to `<url>`:

```
139 \newcommand*{\href}[1]{\TagSurr a{\@href{#1}}}
```

3.7.2 Special cases of Basic Link Macros

`\autanc{<text>}` creates an anchor where `<text>` is the text and the internal label at the same time:

```
140 \newcommand*{\autanc}[1]{\hanc{#1}{#1}}           %% 2010/07/04
```

`\ancref{<id>}{<text>}` makes `<text>` a link to an anchor `<id>` on the same web page. This is especially useful for a “table of contents”—a list of links to sections of the page.

```
141 \newcommand*{\ancref}[1]{\href{\@#1}}
```

`\autref{<text>}` makes `<text>` a link to an anchor named `<text>` itself:

```
142 \newcommand*{\autref}[1]{\ancref{#1}{#1}}         %% 2010/07/04
```


3.7.3 Italic Variants

Some of the link macros get “emphasized” or “italic” variants. Originally I used “emphasized,” later I decided to replace it by “italic,” as I found that I had used italics for another reason than emphasizing. E.g., $\langle text \rangle$ may be ‘bug,’ and I am not referring to some bug, but to the Wikipedia article *Bug*. This has been inspired by some Wikipedia typography convention about referring to titles of books or movies. (The `em` \rightarrow `it` replacement has not been completed yet.)

```

143 % \newcommand*\emhref}[2]{\href{#1}{\emph{#2}}}
144 \newcommand*\ithref}[2]{\href{#1}{\textit{#2}}}
145 \newcommand*\itancref}[2]{\ancref{#1}{\textit{#2}}}% 2010/05/30
146 \newcommand*\emancref}[2]{\ancref{#1}{\emph{#2}}}
```

3.7.4 Built Macros for Links to Local Files

Originally, I wanted to refer to my web pages only, using

`\fileref{ $\langle filename-base \rangle$ }`.

I have used extension `.htm` to avoid disturbing my Atari editor `xEDIT` or the Atari emulator (Hatari). The extension I actually use is stored as macro `\htext` in a more local file (e.g., `.cfg`).—Later I realized that I may want to refer to local files other than web pages, and therefore I introduced a more general `\FileRef{ $\langle filename \rangle$ }`, overlooking that it was the same as `\href`.

```

147 % \newcommand*\FileRef}[1]{\TagSurr a{\@href{#1}}}
148 \newcommand*\htext{.htm}                                %% 2011/10/05
149 \newcommand*\fileref}[1]{\href{#1\htext}}
150 % \newcommand*\emfileref}[2]{\fileref{#1}{\emph{#2}}}
151 \newcommand*\itfileref}[2]{\fileref{#1}{\textit{#2}}}
```

`\fileancref{ $\langle file \rangle$ }{ $\langle anchor \rangle$ }{ $\langle text \rangle$ }` links to anchor $\langle anchor \rangle$ on web page $\langle file \rangle$:

```

152 \newcommand*\fileancref}[2]{%
153   \TagSurr a{\@href{#1\htext\##2}}}
154 % \newcommand*\emfileancref}[3]{\fileancref{#1}{#2}{\emph{#3}}}
```

← 2010/05/31 →

```

155 \newcommand*\itfileancref}[3]{\fileancref{#1}{#2}{\textit{#3}}}
```

3.7.5 Built Macros for Links to Remote Files

`blog.sty` currently (even 2011/01/24) implements my style *not* to open a new browser window or tab for *local* files but to open a new one for *remote* files, i.e., when a file is addressed by a full URL. For the latter case, there is

`\httpref{ $\langle host-path[\#frag] \rangle$ }{ $\langle text \rangle$ }`

making $\langle text \rangle$ a link to `http:// $\langle host-path[\#frag] \rangle$` :

```

156 % \newcommand*{\httpref}[1]{\href{http://#1}}
157 \newcommand*{\httpsref}[1]{%           %% 2010/04/11
158   \TagSurr a{\@href{http://#1" target="_blank}}}
```

With v0.4, macros based on `\httpref` are moved to `texlinks.sty`:

```
159 \RequirePackage[blog]{texlinks}[2011/02/10]
```

Former `\urlref` appears as `\urlhttpref` there ...

```
160 \newcommand \urlref {}          \let\urlref\urlhttpref
```

... and `\ctanref` has been replaced by `\tugctanref`. Let's go on playing with the difference ...

```
161 \newcommand*{\ctanref}[1]{\httpref{ctan.org/tex-archive/#1}}
```

`texlinks` sometimes uses a “permanent alias” `\NormalHTTPref` of `\httpref`:

```
162 \@ifdefinable \NormalHTTPref {\let\NormalHTTPref\httpref}
```

`\httpsref` is the analogue of `\httpref` for `https://`:

```

163 \newcommand*{\httpsref}[1]{%           %% 2011/06/27
164   \TagSurr a{\@href{https://#1" target="_blank}}}
```

3.8 Characters/Symbols

3.8.1 Basic Preliminaries

`&` is made `other` for using it to call HTML's “character entities.”

```
165 \@makeother\&
```

Again we have the two policies about choosing macro names and respectively two new definition commands. `\declareHTMLsymbol{<name>}` defines a macro `\<name>` expanding to `&<name>`; . Checking for prior definedness hasn't been implemented yet. (TODO; but sometimes redefining ...)

```
166 \newcommand*{\declareHTMLsymbol}[1]{\@namedef{#1}{&#1;}}
```

`\renderHTMLsymbol{<macro>}{<name>}` redefines macro `<macro>` to expand to `&<name>`;;

```
167 \newcommand*{\renderHTMLsymbol}[2]{\renewcommand*{#1}{&#2;}}
```

Redefinitions of `\&` and `\%` (well, `\PercentChar` is `fifinddo`'s version of L^AT_EX's `\@percentchar`):

```

168 \renderHTMLsymbol{\&}{amp}
169 \let%\PercentChar
```

3.8.2 Diacritics

For the difference between “diacritic” and “accent,” see Wikipedia.

`\ccedil`:

```
170 \declareHTMLsymbol{ccedil}
```

HTML entities `é`, `ô` etc. can be accessed by T_EX’s accent commands `\’`, `\^`, `\’`, `\"`:

```
171 % \declareHTMLsymbol{eacute}
172 % \declareHTMLsymbol{ocirc}
173 \renewcommand*{\'}[1]{&#1acute;}
174 \renewcommand*{\^}[1]{&#1circ;}
175 \renewcommand*{\'}[1]{&#1grave;}
176 \renewcommand*{\"}[1]{&#1uml;}
```

`\uml{<char>}` may have been overestimated:

```
177 % \newcommand*      {\uml}[1]  {&#1uml;}      %% 2010/08/24
```

3.8.3 Greek

```
178 \declareHTMLsymbol{Alpha}
179 \declareHTMLsymbol{alpha}
180 \declareHTMLsymbol{Beta}
181 \declareHTMLsymbol{beta}
182 \declareHTMLsymbol{Gamma}
183 \declareHTMLsymbol{gamma}
184 \declareHTMLsymbol{Delta}
185 \declareHTMLsymbol{delta}
186 \declareHTMLsymbol{Epsilon}
187 \declareHTMLsymbol{epsilon}
188 \declareHTMLsymbol{Zeta}
189 \declareHTMLsymbol{zeta}
190 \declareHTMLsymbol{Eta}
191 \declareHTMLsymbol{eta}
192 \declareHTMLsymbol{Theta}
193 \declareHTMLsymbol{theta}
194 \declareHTMLsymbol{Iota}
195 \declareHTMLsymbol{iota}
196 \declareHTMLsymbol{Kappa}
197 \declareHTMLsymbol{kappa}
198 \declareHTMLsymbol{Lambda}
199 \declareHTMLsymbol{lambda}
200 \declareHTMLsymbol{My}
201 \declareHTMLsymbol{my}
202 \declareHTMLsymbol{Ny}
203 \declareHTMLsymbol{ny}
204 \declareHTMLsymbol{Xi}
205 \declareHTMLsymbol{xi}
```

```

206 \declareHTMLsymbol{Omikron}
207 \declareHTMLsymbol{omikron}
208 \declareHTMLsymbol{Pi}
209 \declareHTMLsymbol{pi}
210 \declareHTMLsymbol{Rho}
211 \declareHTMLsymbol{rho}
212 \declareHTMLsymbol{Sigma}
213 \declareHTMLsymbol{sigma}
214 \declareHTMLsymbol{sigmaf}
215 \declareHTMLsymbol{Tau}
216 \declareHTMLsymbol{tau}
217 \declareHTMLsymbol{Upsilon}
218 \declareHTMLsymbol{upsilon}
219 \declareHTMLsymbol{Phi}
220 \declareHTMLsymbol{phi}
221 \declareHTMLsymbol{Chi}
222 \declareHTMLsymbol{chi}
223 \declareHTMLsymbol{Psi}
224 \declareHTMLsymbol{psi}
225 \declareHTMLsymbol{Omega}          %% render -> declare 2011/02/26
226 \declareHTMLsymbol{omega}
227 \declareHTMLsymbol{thetasym}
228 \declareHTMLsymbol{upsih}
229 \declareHTMLsymbol{piv}

```

3.8.4 Arrows

Arrows: `\gets`, `\to`, `\uparrow`, `\downarrow` ...

```

230 \renderHTMLsymbol {\gets}      {\larr}
231 \renderHTMLsymbol {\to}        {\rarr}
232 \renderHTMLsymbol {\uparrow}   {\uarr}    %% 2010/09/15
233 \renderHTMLsymbol {\downarrow}{\darr}    %% 2010/09/15

```

3.8.5 Dashes

The ligatures -- and --- for en dash and em dash don’t work in our expanding mode. Now, HTML’s policy for choosing names often prefers shorter names than are recommended for (La)T_EX, so here I adopt a *third* police besides (i) and (ii) earlier; cf. L^AT_EX’s `\textemdash` and `\textendash`.—`\newcommand` does not accept macros whose names start with `end`, so: `\endash`, `\emdash` ...

```

234 \def          \endash  {\&ndash;}          %% \end... illegal
235 \newcommand*{\emdash} {\&mdash;}

```

3.8.6 Spaces

“Math” (not only!) spaces `\,`, `\enspace`, `\quad`, `\qquad`:

```

236 \renderHTMLsymbol{\enspace}{ensp}

```

```

237 \renderHTMlsymbol{\quad} {emsp}
238 \renewcommand* {\qqquad} {\quad\quad}

```

2011/07/22: ` ` allows line breaks, so we introduce `\thinsp` to access ` `, while `\thinspace` and `\,` use Unicode “Narrow No-Break Space” (U+202F, see *Wikipedia Space (punctuation)*; browser support?):

```

239 % \renderHTMlsymbol{\thinspace}{thinsp}
240 % \renderHTMlsymbol{\,,} {thinsp}
241 \declareHTMlsymbol{thinsp}
242 \renderHTMlsymbol{\thinspace}{\#8239}
243 \renderHTMlsymbol{\,,} {\#8239}

```

`\figurespace` (U+2007, cf. *Wikipedia*):

```

244 \newcommand*{\figurespace}{&\#8199;}

```

3.8.7 Quotes, Apostrophe, Prime

`\lq`, `\rq`

```

245 \renderHTMlsymbol{\lq} {lsquo}
246 \renderHTMlsymbol{\rq} {rsquo}

```

In order to use the right single quote for the HTML apostrophe, we must save other uses before. `\screentqd{<text>}` is used for screen messages, and `\urlapostr` is the version of the right single quote for URLs of Wikipedia articles:

```

247 \newcommand*{\screentqd}[1]{‘#1’}
248 \newcommand*{\urlapostr} {’} %% 2010/09/10

```

Here finally is the change of `’`:

```

249 \MakeActiveDef\’{\&rsquo;}

```

... **TODO** `\MakeActiveLet\’\rq!` And this might better be in `\BlogCodes!` would save `\screentqd!` Tilde likewise!? ... **TODO** change `\catcode\’!` 2010/04/26

`\bdquo`, `\ldquo`, `\rdquo`, `\sbquo`, `\prime`, `\Prime` ...

```

250 \declareHTMlsymbol{bdquo} %% 2011/09/23
251 \declareHTMlsymbol{ldquo}
252 \declareHTMlsymbol{rdquo}
253 \declareHTMlsymbol{sbquo} %% 2010/07/01
254 \renewcommand*{\prime}{&prime;}
255 \declareHTMlsymbol{Prime}
256 % \newcommand*{\Prime}{&Prime;}

```

`\endqtd{<text>}` quotes in the English style using double quote marks, `\enqtd{<text>}` uses single quote marks instead, and `\dedqtd{<text>}` quotes in German style:

```

257 \def\endqtd#1{\ldquo#1\rdquo}      %% \newcommand: ‘‘\end”
258 \newcommand*\enqtd[1]{\lq#1\rq}    %% 2010/09/08, \new... 2010/11/08
259 \newcommand*\dedqtd[1]{\bdquo#1\ldquo}
260 \newcommand*\deqtd[1]{\&sbquo;#1\lsquo;}  %% corr. 2011/05/14

```

TODO `\glqq` from `german.sty` etc.

3.8.8 Math

Because `<` and `>` are used for HTML’s element notation, we provide aliases `\gt`, `\lt` for mathematical `<` and `>`:

```

261 \declareHTMLsymbol{gt}
262 \declareHTMLsymbol{lt}

```

`\ge`, `\le`, and `\ne` for \geq , \leq , and \neq resp.:

```

263 \declareHTMLsymbol{ge}
264 \declareHTMLsymbol{le}
265 \declareHTMLsymbol{ne}

```

We also provide their \TeX aliases `\geq`, `\leq`, `\neq`:

```

266 \let\geq\ge
267 \let\leq\le
268 \let\neq\ne

```

Angle braces `\langle` and `\rangle`:

```

269 \renderHTMLsymbol{\langle}{lang}
270 \renderHTMLsymbol{\rangle}{rang}

```

The one-argument macro `\angled{⟨angled⟩}` allows better readable code (should be in a more general package):

```

271 \newcommand*\angled[1]{\langle#1\rangle}

```

Curly braces `\{` and `\}` ...:

```

272 \begingroup
273   \Delimiters\[\] \gdef\{[\] \gdef\}[{\]
274 \endgroup

```

\TeX ’s `\ast` corresponds to the “lower” version of the asterisk:

```

275 \renderHTMLsymbol{\ast}{lowast}      %% 2011/03/29

```

Besides \TeX ’s `\subset` and `\subseteq`, we provide short versions `\sub` and `\sube` inspired by HTML:

```

276 \declareHTMLsymbol{sub}              %% 2011/04/04
277 \let\subset\sub                      %% 2011/05/08
278 \declareHTMLsymbol{sube}            %% 2011/03/29
279 \let\subseteq\sube                  %% 2011/05/08

```

TeX and HTML agree on `\cap`, `\cup`, and `\times`:

```
280 \declareHTLsymbol{cap}                %% 2011/04/04
281 \declareHTLsymbol{cup}                %% 2011/04/04
282 \declareHTLsymbol{times}              %% 2011/04/04
```

We stick to TeX’s `\emptyset`

```
283 \renderHTLsymbol{\emptyset}{empty}    %% 2011/04/14
```

We need `\minus` since math mode switching is not supported by blog:

```
284 \declareHTLsymbol{minus}              %% 2011/03/31
```

We override HTML’s ‘ˆ’ to get TeX’s `\circ` (i.e., `\circ`; **but I cannot see it on my own pages!?**):

```
285 \renderHTLsymbol{\circ}{\#x2218}      %% 2011/04/28
286 \renderHTLsymbol{\cdot}{middot}       %% 2011/05/07
```

`\sdot` generates `&sdot`, a variant of `·`; reserved for the dot product according to the German *Wikipedia*

```
287 \declareHTLsymbol{sdot}               %% 2011/05/08
```

I provide `\degrees` for the degree symbol. L^AT_EX already has `\deg` as an operator, therefore I do not want to use `\declareHTLsymbol` here.

```
288 \newcommand*{\degrees}{\&deg;}
```

3.8.9 Other

The tilde `~` is used for its wonderful purpose, by analogy to TeX:

```
289 \renderHTLsymbol{~}{nbsp}
```

But now we need a replacement `\tilde` for URLs involving home directories of institution members (should better be `\tildechar` or `\TildeChar`, cf. `fifinddo`):

```
290 {\@makeother~ \gdef\tilde{~} \gdef\tildechar{~}}
```

Horizontal ellipsis: `\dots` ...

```
291 \renderHTLsymbol {\dots} {hellip}
```

`\copyright`:

```
292 \renderHTLsymbol{\copyright}{copy}
```

`\bullet`

```
293 \renderHTLsymbol{\bullet}{bull}
```

`\euro`:

```
294 \declareHTLsymbol{euro}
```

L^AT_EX's `\S` prints the “section sign” ‘§’. In HTML, the latter accessed by `§`, we “redirect” `\S` to this:

```
295 \renderHTMLsymbol{\S}{sect}

\ dagger, \ddagger:

296 \renderHTMLsymbol{\dagger}{dagger}
297 \renderHTMLsymbol{\ddagger}{Dagger}
```

3.9 T_EX-related

Somebody actually using `blog.sty` must have a need to put down notes about T_EX for her own private purposes at least—I expect.

3.9.1 Logos

“Program” names might be typeset in a special font, I once thought, and started tagging program names with `\prg`. It could be `\texttt` or `\textsf` like in documentations of L^AT_EX packages. However, sans-serif is of doubtful usefulness on web pages, and typewriter imitations usually look terrible on web pages. So I am waiting for a better idea and let `\prg` just remove the braces.

```
298 \newcommand*{\prg}[1]{} \let\prg\@firstofone
299 \newcommand*{\BibTeX}{\prg{BibTeX}} %% 2010/09/13
300 \renewcommand*{\TeX}{\prg{TeX}}
301 \renewcommand*{\LaTeX}{\prg{LaTeX}}
302 \newcommand*{\allTeX}{\prg{(La)TeX}}%% 2010/10/05
303 \newcommand*{\LuaTeX}{\prg{LuaTeX}}
304 \newcommand*{\pdfTeX}{\prg{pdfTeX}}
305 \newcommand*{\XeTeX}{\prg{XeTeX}} %% 2010/10/09
306 \newcommand*{\TeXbook}{\TeXbook} %% 2010/09/13
```

3.9.2 Describing Macros

With v0.4, T_EX-related *links* are moved to `texlinks.sty`.

`\texcs{\<tex-cmd-name>}` or `\texcs\<tex-cmd-name>` (care for spacing yourself):

```
307 \newcommand*{\texcs}[1]{\code{\string#1}} %% 2010/11/13
```

Good old `\cs{\<tex-cmd-name>}` may be preferable:

```
308 \def\cs#1{\code{\BackslashChar#1}} %% 2011/03/06
```

`\metavar{\<name>}`:

```
309 \newcommand*{\metavar}[1]{\angled{\meta{#1}}}
```


3.10 Tables

3.10.1 Indenting

There are three levels of indenting:

`\indenti`, `\indentii`, and `\indentiii`.

The intention for these was to get readable HTML code. Not sure ...

```
310 {\catcode'\ =12%% 2010/05/19
311 \gdef\indenti{ }\gdef\indentii{ }\gdef\indentiii{ }}
```

3.10.2 Starting/Ending Tables

2010/07/17:

```
312 \newcommand*\startTable}[1]{<table #1>}
313 \def\endTable{</table>}
314 \newcommand*\@frame@box{\@frame{box}}
315 \newcommand*\@frame@groups{\@frame{groups}}
316 \newenvironment{allrulestable}[2]
317   {\startTable{\@cellpadding{#1} \@width{#2}
318               \@frame@box rules="all"}\CLBrk
319   \indenti<tbody>}
320   {\indenti</tbody>\CLBrk\endTable}
```

3.10.3 Rows

I first thought it would be good for readability if some HTML comments explain nesting or briefly describe the content of some column, row, or cell. But this is troublesome when you want to comment out an entire table ...

```
321 \newenvironment*{TableRow}[2]{%% lesser indentation 2011/04/25
322   \ \comment{ #1 }\CLBrk
323   \indenti<tr #2>%
324   }{%
325   \indenti</tr>}
326 \newenvironment{tablecoloredrow}[2]
327   {\TableRow{#1}{\@bgcolor{#2}}}
328   {\endTableRow}
```

“top” 2010/05/18:

```
329 \newenvironment{tablerow}[1]{\TableRow{#1}{\@valign@t}}
330   {\endTableRow}
```

2010/07/18:

```
331 \newcommand*\starttr{<tr>}
332 \def\endtr{</tr>}
```

3.10.4 Cells

```

333 \newcommand*\simplecell{\SimpleTagSurr{td}} %% 2010/07/18
334 % \newcommand*\TableCell}[2]{\indentiii<td #1>#2</td>}
335 % \newcommand*\TableCell}[2]{\indentiii\TagSurr{td}{#1}{#2}}
336 %% <- 2010/07/18 ->
337 \newcommand*\TableCell}[2]{\indentiii\startTd{#1}#2\endTd}

```

2010/06/15:

```

338 \newcommand*\colorwidthcell}[2]{\TableCell{\@bgcolor{#1}\@width{#2}}}
339 \newcommand*\tablewidthcell}[1]{\TableCell{\@width{#1}}}
340 \newcommand*\tablecell{\TableCell{}}
341 \newcommand*\tableCell{\TableCell{\@align@c}}

```

Idea: use closing star for environment variants!?

```

342 % %% 2010/05/19:
343 \newenvironment{bigtablecell}[1]{\BigTableCell{#1}{}}
344 {\endBigTableCell}
345 % {\ifx\#1\\% %% 2010/05/30
346 % \indentii\ \comment{#1}\CLBrk
347 % \fi
348 % \indentiii<td>}
349 % {\indentii</td>} %% !? 2010/05/23

```

2010/06/05:

```

350 \newenvironment{BigTableCell}[2]
351 {\ifx\#1\\% \comment{#1}\CLBrk\fi
352 \indentiii\startTd{#2}}
353 {\indentii\endTd} %% TODO indent? 2010/07/18

```

2010/07/18:

```

354 \newcommand*\startTd}[1]{<td #1>}
355 \def\endTd{</td>}
356 \newcommand*\emptycell{<td />} %% 2011/10/07

```

3.10.5 Filling a Row with Dummy Cells

Generalization 2010/06/28:

```

357 % \newcommand*\FillRow}[2]{% %% broke line 2011/01/24
358 % \indentiii\TagSurr{td}{\@colspan{#1} #2}{}}
359 %% <- 2010/07/18 ->
360 \newcommand*\FillRow}[2]{\indentiii\startTd{\@colspan{#1} #2}\endTd}
361 \newcommand*\fillrow}[1]{\FillRow{#1}{}}
362 \newcommand*\fillrowcolor}[2]{\FillRow{#1}{\@bgcolor{#2}}}

```

3.11 Misc

`\comment{comment}` produces a one-line HTML comment. By contrast, there is an environment `{\commentlines}{comment}` for multi-line comments. It is convenient for “commenting out” code (unless the latter contains other HTML comments ...) where *comment* is a *comment* for explaining what is commented out.

```

363 \newcommand*\comment}[1]{<!--#1-->}
364 % \newcommand*\commentlines[1]{\comment{^^J#1^^J}} %% 2010/05/07
365 % %% <- TODO bzw. \endlinechar='^^J 2010/05/09 back 2010/05/10
366 \newenvironment{commentlines}[1] %% 2010/05/17
367 {<!--#1}
368 {-->}

```

TeX’s `\hrule` (rather deprecated in L^AT_EX) is redefined to produce an HTML horizontal line:

```

369 \renewcommand*\hrule{<hr>}

```

Redefining `_` to be the same as `\space` may be helpful for manual indenting or spacing of HTML code. Or better (just now remembering): I used it for making “ASCII trees” with the `<pre>` element (redefined `verbatim`).

```

370 \let\_ \space

```

I couldn’t find a perfect way to generate `<p>`. Actually I started completing the present documentation when I had decided to implement automatic generation of `<p>` from empty lines.

```

371 % \def\par{<p>} %% + empty lines !? 2010/04/26
    ← difficult with \stop; 2010/09/10: \endgraf produces </p>!?
372 \renewcommand*\endgraf{</p>}

```

2010/04/28: `
` can be generated either by `\newline` or by `_`:

```

373 \renewcommand*\newline{<br>}
374 \let\_ \newline

```

`\rightpar{text}` places *text* flush right. I have used this for ‘Last revised ...’ and for placing navigation marks.

```

375 \newcommand*\rightpar{\TagSurr p\align@r} %% 2010/06/17

```

Often I use `\rightpar` with *italics*, now there is `\rightitpar{text}` for this purpose:

```

376 \newcommand*\rightitpar[1]{\rightpar{\textit{#1}}}

```

For references, there were

```

377 % \catcode'\^=\active
378 % \def^#1{\SimpleTagSurr{sup}{#1}}

```

and

```

379 % \newcommand*\src{[1]{\SimpleTagSurr{sup}{[#1]}}

```

as of 2010/05/01, inspired by the <ref> element of MediaWiki; moved to
xmlprint.tex 2010/06/02.

3.12 The End

```

380 \endinput

```

3.13 VERSION HISTORY

```

381 v0.1      2010/08/20  final version for DFG
382 v0.2      2010/11/08  final documentation version before
383                      moving some functionality to 'fifinddo'
384 v0.3      2010/11/10  removed ^^J from \head
385                      2010/11/11  moving stuff to fifinddo.sty; \BlogCopyFile
386                      2010/11/12  date updated; broke too long code lines etc.;
387                      \CatCode replaced (implemented in niceverb only);
388                      \ifBlogAutoPars etc.
389                      2010/11/13  doc: \uml useful in ...; \texcs
390                      2010/11/14  doc: argument for {commentlines},
391                      referring to environments with curly braces,
392                      more on \ditem
393                      2010/11/15  TODO: usage, templates
394                      2010/11/16  note on {verbatim}
395                      2010/11/23  doc. corr. on \CtanPkgRef
396                      2010/11/27  "keyword"; \CopyLine without 'fd'
397                      2010/12/03  \emhttpref -> \ithttpref
398                      2010/12/23  '%' added to \texhaxpref
399                      2011/01/23  more in \Provides...
400                      2011/01/24  updated copyright; resolving 'td' ("today")
401                      JUST STORED as final version before texlinks.sty
402 v0.4      2011/01/24  moving links to texlinks.sty
403 v0.41     2011/02/07  \NormalHTTPref
404                      2011/02/10  refined call of 'texlinks'
405 part of MOREHYPE RELEASE r0.3
406 v0.5      2011/02/22  \BlogProvidesFile
407                      2011/02/24  ... in \BlogCopyFile
408                      2011/02/25  ordering symbols
409                      2011/02/26  subsection Greek; note on \declareHTMLsymbol
410                      2011/03/04  diacritics
411                      2011/03/06  \cs
412                      2011/03/09  \var
413                      2011/03/16  \robots
414                      2011/03/19  doc. \fileancref arg.s corr.

```

```

415      2011/03/29 \Sigma, ...
416      2011/03/31 \minus
417      2011/04/04 \times, \sub, \delta
418      2011/04/11 Greek completed
419      2011/04/14 \emptyset
420      2011/04/22 \deqtd
421      2011/04/24 doc.: folding, \stylesheet, ordered "tables";
422      @border, @align, @valign
423      2011/04/25 lesser indentation with TableRow
424      2011/04/26 \,, \thinspace, \@title; doc. \@name
425      2011/04/28 [\circ] PROBLEM still
426      2011/04/29 \rightitpar
427      2011/05/07 \cdot
428      2011/05/08 extended doc. on math symbols; \sdot;
429      \ast replaces \lowast; \subset, \subseteq;
430      \angled
431      2011/05/09 \euro
432      2011/05/11 |\geq| etc.; new section "logical markup"
433      2011/05/12 corr. doc. \heading
434      2011/05/14 right mark of \deqtd was rsquo instead of lsquo!
435      2011/05/18 \S and note on \StoreOtherCharAs
436      2011/06/27 \httpsref; doc: \acro
437      2011/07/22 \thinspace vs. \thinsp; 'fifinddo''s
438      2011/07/25 "todo" on \description
439      2011/08/18f. removing \FileRef, 0.42-> 0.5
440      2011/08/31 clarified use of \urlapostr
441 part of MOREHYPE RELEASE r0.4
442 v0.6 2011/09/08 doc. uses \HTML, \lq/\rq with &circ;,
443      doc. fix 'mult-'; \degrees
444      2011/09/21 \acronym
445      2011/09/22 \metavar; TODO \glqq...
446      2011/09/23 \bdquo
447      2011/09/25 doc. 'Characters/Symbols'; \figurespace
448      2011/09/27 "universal" attributes completed, reworked doc.
449      2011/09/30 end lists with </li>
450      2011/10/01 \dagger, \ddagger
451      2011/10/04 \item includes </li>
452      2011/10/05 {style}; doc. \acronym -> \acro, \pagebreak,
453      rm. \description; {center} accesses <center>,
454      \useHTMLenvironment replaces \declareHTMLelement
455      and \renderHTMLelement, message "generating"
456      2011/10/07 \emptycell
457      2011/10/10 doc.: page breaks, $$->\[/\]
458

```

4 Real Web Pages with Inavicol.sty

This is the code and documentation of the package mentioned in Sec. 2.2.

```

1 \ProvidesPackage{lnavicol}[2011/10/10
2                               left navigation column with blog.sty]
3 %%
4 %% Copyright (C) 2011 Uwe Lueck,
5 %% http://www.contact-ednotes.sty.de.vu
6 %% -- author-maintained in the sense of LPPL below --
7 %%
8 %% This file can be redistributed and/or modified under
9 %% the terms of the LaTeX Project Public License; either
10 %% version 1.3c of the License, or any later version.
11 %% The latest version of this license is in
12 %% http://www.latex-project.org/lppl.txt
13 %% We did our best to help you, but there is NO WARRANTY.
14 %%
15 %% Please report bugs, problems, and suggestions via
16 %%
17 %% http://www.contact-ednotes.sty.de.vu

```

4.1 blog.sty Required

—but what about options ([TODO](#))?

```

18 \RequirePackage{blog}

```

4.2 Switches

There is a “standard” page width and a “tight one” (the latter for contact forms)—`\iftight`:

```

19 \newif\iftight

```

In order to move an anchor to the *top* of the screen when the anchor is near the page end, the page must get some extra length by adding empty space at its bottom—`\ifdeep`:

```

20 \newif\ifdeep

```

4.3 Page Style Settings (to be set locally)

```

21 % \newcommand*{\pagebgcolor}{\#f5f5f5} %% CSS whitesmoke
22 % \newcommand*{\pagespacing}{\@cellpadding{4} \@cellspacing{7}}
23 % \newcommand*{\pagenavicolwidth}{125}
24 % \newcommand*{\pagemaincolwidth}{584}
25 % \newcommand*{\pagewholewidth}{792}

```

4.4 Possible Additions to **blog.sty**

4.4.1 Tables

`\begin{spancolscell}{ $\langle number \rangle$ }{ $\langle style \rangle$ }` opens an environment that contains a row and a single cell that will span $\langle number \rangle$ table cells and have style $\langle style \rangle$:

```
26 \newenvironment{spancolscell}[2]{%
27     \starttr\startTd{\@colspan{#1} #2 %
28         \@width{100\%}}% %% TODO works?
29     }\endTd\endtr}
```

The `{\hiddencells}` environment contains cells that do not align with other cells in the surrounding table. The purpose is using cells for horizontal spacing.

```
30 \newenvironment{hiddencells}
31     {\startTable{}\starttr}
32     {\endtr\endTable}
```

`{\pagehiddencells}` is like `{\hiddencells}` except that the HTML code is indented:

```
33 \newenvironment{pagehiddencells}
34     {\indentii\hiddencells}
35     {\indentii\endhiddencells}
```

`\begin{FixedWidthCell}{ $\langle width \rangle$ }{ $\langle style \rangle$ }` opens the `{FixedWidthCell}` environment. The content will form a cell of width $\langle width \rangle$. $\langle style \rangle$ are additional formatting parameters:

```
36 \newenvironment{FixedWidthCell}[2]
37     {\startTd{#2}\startTable{\@width{#1}}%
38     \starttr\startTd{}}
39     {\endTd\endtr\endTable\endTd}
```

`\tablehspace{ $\langle width \rangle$ }` is a variant of L^AT_EX's `\hspace{ $\langle glue \rangle$ }`. It may appear in a table row:

```
40 \newcommand*{\tablehspace}[1]{\startTd{\@width{#1} /}}
```

4.4.2 Graphics

The command names in this section are inspired by the names in the standard L^AT_EX `graphics` package. (They may need some re-organization [TODO](#).)

`\simpleinclgrf{ $\langle file \rangle$ }` embeds a graphic file $\langle file \rangle$ without the tricks of the remaining commands.

```
41 \newcommand*{\simpleinclgrf}[1]{\IncludeGrf{alt="" \@border{0}}%
42     {#1}}
```

`\IncludeGrf{ $\langle style \rangle$ }{ $\langle file \rangle$ }` embeds a graphic file $\langle file \rangle$ with style settings $\langle style \rangle$:

```

43 \newcommand*{\IncludeGrf}[2]{}

\includegraphic{<width>}{<height>}{<file>}{<border>}{<alt>}{<tooltip>} ...:

44 \newcommand*{\includegraphic}[6]{%
45   \IncludeGrf{%
46     \@width{#1} \@height{#2} %% data; presentation:
47     \@border{#4}
48     alt="#5" \@title{#6}}%
49   {#3}}

\insertgraphic{<wd>}{<ht>}{<f>}{<b>}{<align>}{<hsp>}{<vsp>}{<alt>}{<t>}
adds <hsp> for the @hspace and <vsp> for the @vspace attribute:

50 \newcommand*{\insertgraphic}[9]{%
51   \IncludeGrf{%
52     \@width{#1} \@height{#2} %% data; presentation:
53     \@border{#4}
54     align="#5" hspace="#6" vspace="#8"
55     alt="#8" \@title{#9}}%
56   {#3}}

\includegraphic{<wd>}{<ht>}{<file>}{<anchor>}{<border>}{<alt>}{<tooltip>}
uses an image with \includegraphic parameters as a link to <anchor>:

57 \newcommand*{\inclgrfref}[7]{%
58   \fileref{#4}{\includegraphic{#1}{#2}{#3}%
59     {#5}{#6}{#7}}}

```

4.4.3 HTTP/Wikipedia tooltips

`\httptipref{<tip>}{<www>}{<text>}` works like `\httpref{<www>}{<text>}` except that `<tip>` appears as “tooltip”:

```

60 \newcommand*{\httptipref}[2]{%
61   \TagSurr a{\@title{#1}\@href{http://#2}\@target@blank}}

```

`\@target@blank` abbreviates the `@target` setting for opening the target in a new window or tab:

```

62 \newcommand*{\@target@blank}{target="_blank"}

```

`\wikitipref{<lc>}{<lem>}{<text>}` works like `\wikiref{<lc>}{<lem>}{<text>}` except that “Wikipedia” appears as “tooltip”. `\wikideref` and `\wikienref` are redefined to use it:

```

63 \newcommand*{\wikitipref}[2]{%
64   \httptipref{Wikipedia}{#1.wikipedia.org/wiki/#2}}
65 \renewcommand*{\wikideref}{\wikitipref{de}}
66 \renewcommand*{\wikienref}{\wikitipref{en}}

```


4.5 Page Structure

The body of the page is a table of three rows and two columns.

4.5.1 Page Head Row

`\PAGEHEAD` opens the head row and a single cell that will span the two columns of the second row.

```

67 \newcommand*{\PAGEHEAD}{%
68   \startTable{%
69     \@align@c\
70     \@bgcolor{\pagebgcolor}%
71     \@border{0}%%           %% TODO local
72     \pagespacing
73     \iftight \else \@width\pagewidth \fi
74   }\CLBrk
75   %% omitting <tbody>
76   \comment{ HEAD ROW }\CLBrk
77   \indentii\spancolscell{2}{}%
78 }
79 % \newcommand*{\headgrf}[1]{%           %% rm. 2011/10/09
80 %   \indentiii\simplecell{\simpleinclgrf{#1}}
81 % \newcommand*{\headgrfskiptitle}[3]{%
82 %   \pagehiddencells
83 %   \headgrf{#1}\CLBrk
84 %   \headskip{#2}\CLBrk
85 %   \headtitle1{#3}\CLBrk
86 %   \endpagehiddencells}

```

`\headuseskiptitle{<grf>}{<skip>}{<title>}` first places `<grf>`, then skips horizontally by `<skip>`, and then prints the page title as `<h1>`:

```

87 \newcommand*{\headuseskiptitle}[3]{%
88   \pagehiddencells\CLBrk
89   \indentiii\simplecell{#1}\CLBrk
90   \headskip{#2}\CLBrk
91   \headtitle1{#3}\CLBrk
92   \endpagehiddencells}

```

`\headskip{<skip>}` is like `\tablehspace{<skip>}` except that the HTML code gets an indent.

```

93 \newcommand*{\headskip}{\indentiii\tablehspace}

```

Similarly, `\headtitle{<digit>}{<text>}` is like `\heading{<digit>}{<text>}` apart from an indent and being put into a cell:

```

94 \newcommand*{\headtitle}[2]{\indentiii\simplecell{\heading#1{#2}}}

```

4.5.2 Navigation and Main Row

`\PAGENAVI` closes the head row and opens the “navigation” column, actually including an `{itemize}` environment. Accordingly, `writings.fdf` has a command `\fileitem`. But it seems that I have not been sure ...

```

95 \newcommand*{\PAGENAVI}{%
96     \indentii\endspancolscell\CLBrk
97     \indentii\starttr\CLBrk
98     \ \comment{NAVIGATION COL}\CLBrk
99     \indentii\FixedWidthCell\pagenavicolwidth
100                                     {\@class{paper}

```

← using `@class=paper` here is my brother’s idea, not sure about it ...

```

101                                     \@valign@t}
102     %% omitting ‘\@height{100\%}’
103     \itemize}

```

`\PAGEMAINvar{<width>}` closes the navigation column and opens the “main content” column. The latter gets width `<width>`:

```

104 \newcommand*{\PAGEMAINvar}[1]{%
105     \indentii\enditemize\ \endFixedWidthCell\CLBrk
106     \ \comment{ MAIN COL }\CLBrk
107     \indentii\FixedWidthCell{#1}{}}

```

... The width may be specified as `\pagemaincolwidth`, then `\PAGEMAIN` works like `\PAGEMAINvar{\pagemaincolwidth}`:

```

108 \newcommand*{\PAGEMAIN}{\PAGEMAINvar\pagemaincolwidth}

```

4.5.3 Footer Row

`\PAGEFOOT` closes the “main content” column as well as the second row, and opens the footer row:

```

109 \newcommand*{\PAGEFOOT}{%
110     \indentii\endFixedWidthCell\CLBrk
111     % \indentii\tablespace{96}\CLBrk %% vs. \pagemaincolwidth
112     %% <- TODO margin right of foot
113     \indentii\endtr\CLBrk
114     \ \comment{ FOOT ROW / }\CLBrk
115     \indentii\spancolscell{2}{\@class{paper} \@align@c}%

```

← again class “paper”!?

```

116 }

```

`\PAGEEND` closes the footer row and provides all the rest ... needed?

```

117 \newcommand*{\PAGEEND}{\indentii\endspancolscell\endTable}

```

4.6 The End and HISTORY

```

118 \endinput
119
120 HISTORY
121
122 2011/04/29   started (? \if...)
123 2011/09/01   to CTAN as 'twocolpg.sty'
124 2011/09/02   renamed
125 2011/10/09f. documentation more serious
126

```

5 Beamer Presentations with the blogdot Package

5.1 Overview

blogdot.sty extends blog.sty in order to construct “HTML slides.” One “slide” is a 3×3 table such that

1. it **fills** the computer **screen**,
2. the center cell is the “**type area**,”
3. the “margin cell” below the center cell is a **link** to the **next** “slide,”
4. the lower right-hand cell is a “**restart**” link.

Six **size parameters** listed in Sec. 5.4 must be adjusted to the screen in `blogdot.cfg` (or in a file with project-specific definitions).

We deliver a file `blogdot.css` containing **CSS** font size declarations that have been used so far; you may find better ones or ones that work better with your screen size, or you may need to add style declarations for additional HTML elements.

Another parameter that the user may want to modify is the “**restart**” **anchor name** `\BlogDotRestart` (see Sec. 5.6). Its default value is `START` for the “slide” opened by the command `\titlescreenpage` that is defined in Sec. 5.5.

That slide is meant to be the “**title slide**” of the presentation. In order to **display** it, I recommend to make and use a **link** to `START` somewhere (such as with `blog.sty`’s `\ancref` command). The *content* of the title slide is *centered* horizontally, so certain commands mentioned *below* (centering on other slides) may be useful.

After `\titlescreenpage`, the next main **user commands** are

`\nextnormalscreenpage{<anchor-name>}` starts a slide whose content is aligned flush left,

`\nextcenterscreenpage{⟨anchor-name⟩}` starts a slide whose content is centered horizontally.

—cf. Sec. 5.7. Right after these commands, as well as right after `\titlescreen‘\-page’`, code is used to generate the content of the **type area** of the corresponding slide. Another `\next...` command closes that content and opens another slide. The presentation (the content of the very last slide) may be finished using `\screenbottom{⟨final⟩}` where `⟨final⟩` may be arbitrary, or `START` may be a fine choice for `⟨final⟩`.

Finally, there are user commands for **centering** slide content horizontally (cf. Sec. 5.8):

`\cheading{⟨digit⟩}{⟨title⟩}` “printing” a heading centered horizontally—even on slides whose remaining content is aligned *flush left* (I have only used `⟨digit⟩=2` so far),

`\begin{textblock}{width}` “printing” the content of a `{textblock}` environment with maximum line width `⟨width⟩` flush left, while that “block” as a whole may be centered horizontally on the slide due to choosing `\nextcenterscreenpage`—especially for **list** environments with entry lines that are shorter than the type area width and thus would not look centered (below a centered heading from `\cheading`).

The so far single **example** of a presentation prepared using `blogdot` is `dantev45.htm`, a sketch of applying `fifinddo` to package documentation and HTML generation. A “driver” file is needed for generating the HTML code for the presentation from a `.tex` source by analogy to generating any HTML file using `blog.sty`. For the latter purpose, I have named my driver files `makehtml.tex`. For `dantev45.htm`, I have called that file `makedot.tex`, the main difference to `makehtml.tex` is loading `blogdot.sty` in place of `blog.sty`.

This example also uses a file `dantev45.fdf` that defines some commands that may be more appropriate as user-level commands than the ones presented here (which may appear to be still too low-level-like):

`\teilpage{⟨number⟩}{⟨title⟩}` making a “cover slide” for announcing a new “part” of the presentation in German,

`\labelsection{⟨label⟩}{⟨title⟩}` starting a slide with heading `⟨title⟩` and with anchor `⟨label⟩` (that is displayed on clicking a *link* to `⟨label⟩`)—using

`\nextnormalscreenpage{⟨label⟩}` and `\cheading2{⟨title⟩}`,

`\labelcentersection{⟨label⟩}{⟨title⟩}` like the previous command except that the slide content will be *centered* horizontally, using

`\nextcenterscreenpage{⟨title⟩}`.

Reasons to make HTML presentations may be: (i) As opposed to office software, this is a transparent light-weight approach. Considering *typesetting* slides with \TeX , (ii) \TeX 's advanced typesetting abilities such as automatical page breaking are not very relevant for slides; (iii) a typesetting run needs a second or a few seconds, while generating HTML with `blog.sty` needs a fraction of a second; (iv) adjusting formatting parameters such as sizes and colours needed for slides is somewhat more straightforward with HTML than with \TeX .

5.2 File Header

```

1 \NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
2 \ProvidesPackage{blogdot}[2011/10/10 v0.2 HTML presentations (UL)]
3 %% copyright (C) 2011 Uwe Lueck,
4 %% http://www.contact-ednotes.sty.de.vu
5 %% -- author-maintained in the sense of LPPL below.
6 %%
7 %% This file can be redistributed and/or modified under
8 %% the terms of the LaTeX Project Public License; either
9 %% version 1.3c of the License, or any later version.
10 %% The latest version of this license is in
11 %% http://www.latex-project.org/lppl.txt
12 %% We did our best to help you, but there is NO WARRANTY.
13 %%
14 %% Please report bugs, problems, and suggestions via
15 %%
16 %% http://www.contact-ednotes.sty.de.vu
17 %%
```

5.3 blog Required

blogdot is an extension of `blog` (but what about options? [TODO](#)):

```

18 \RequirePackage{blog}
```

5.4 Size Parameters

I assume that it is clear what the following six page dimension parameters

<code>\leftpagemargin</code>	<code>\rightpagemargin</code>	<code>\upperpagemargin</code>
<code>\lowerpagemargin</code>	<code>\typeareaawidth</code>	<code>\typeareaheight</code>

mean. The choices are what I thought should work best on my 1024×600 screen; but I had to optimize the left and right margins experimentally (with Mozilla Firefox 3.6.22 for Ubuntu canonical - 1.0); see `blogdot.cfg`:

```

19 \newcommand*\leftpagemargin{192}
20 \newcommand*\rightpagemargin{\leftpagemargin}
```

So `\rightpagemargin` ultimately is the same as `\leftpagemargin` as long as you don't redefine it, and it suffices to `\renewcommand \leftpagemargin` in order to get a horizontally centered type area with user-defined margin widths.—Something analogous applies to `\upperpagemargin` and `\lowerpagemargin`:

```
21 \newcommand*\upperpagemargin{80}
22 \newcommand*\lowerpagemargin{\upperpagemargin}
```

A difference to the “horizontal” parameters is (I expect) that the position of the type area on the screen is affected by `\upperpagemargin` only, and you may choose `\lowerpagemargin` just large enough that the next slide won't be visible on any computer screen you can think of.

```
23 \newcommand*\typeareawidth{640}
24 \newcommand*\typeareaheight{440}
```

Centering with respect to web page body may work better on different screens (2011/10/03), but it doesn't work here (2011/10/04).

```
25 % \renewcommand*\body{%
26 %     </head>\CLBrk
27 %     <body \bgcolor{\bodybgcolor} \align@c>}
```

5.5 (Backbone for) Starting a “Slide”

`\startscreenpage{<style>}{<anchor-name>}`

```
28 \newcommand*\startscreenpage[2]{%% 0 2011/09/25!?:
29     \startTable{%
30         \@cellpadding{0} \@cellspacing{0}%
31         \maybe@showmargins
32     }%
33     \CLBrk                               %% 2011/10/03
34     \starttr
```

First cell determines both height of upper page margin `\upperpagemargin` and width of left page margin `\leftpagemargin`:

```
35     \startTd{@width {\leftpagemargin }%
36             \@height{\upperpagemargin}}%
37 %         \textcolor{\bodybgcolor}{XYZ}%
38     \endTd
```

Using `\typeareawidth`:

```
39 %     \startTd{@width{\typeareawidth}}\endTd
40     \simplecell{%
41         \CLBrk
42         \hanc{#2}{\hvspace{\typeareawidth}%
43                     {\upperpagemargin}}%
44         \CLBrk
45     }%
```

Final cell of first row determines right margin width:

```

46      \startTd{\@width{\leftmargin}}\endTd
47      \endtr
48      %      \tablerow{#1}%
49      \starttr
50      \emptycell\startTd{\@height{\typeareaheight}#1}%
51      }

```

`\ShowBlogDotMargins` shows borders of the page margins:

```

52      \let\maybe@showmargins\@empty
53      \newcommand*{\ShowBlogDotMargins}{%
54          \def\maybe@showmargins{\@frame@box rules="all"}}

```

However, the rules seem to affect horizontal positions ...

`\titlescreenpage` (`\STARTscreenpage` [TODO?](#)) opens the title page (I thought). To get it to your screen, (make and) click a link like

`\ancref{START}{startLpresentation}` :

```

55      \newcommand*{\titlescreenpage}{%
56          \startscreenpage{\@align@c}{START}}

```

5.6 Finishing a “Slide” and “Restart” (Backbone)

`\screenbottom{<next-anchor>}` finishes the current slide and links to the `<next-anchor>`, the anchor of a slide opened by

`\startscreenpage{<style>}{<next-anchor>}`.

More precisely, the margin below the type area is that link. The corner at its right is a link to the anchor to whose name `\BlogDotRestart` expands.

```

57      \newcommand*{\screenbottom}[1]{%
58          \endTd\emptycell
59          \endtr
60          \CLBrk
61          \starttr
62              \emptycell
63              \CLBrk
64              \simplecell{\ancref{#1}%
65                          {\hvspace{\typeareawidth}%
66                          {\lowerpagemargin}}}%
67              \CLBrk
68              \simplecell{\ancref{\BlogDotRestart}%
69                          {\hvspace{\rightmargin}%
70                          {\lowerpagemargin}}}%
71          \endtr
72          \CLBrk
73          \endTable
74      }

```

The default for `\BlogDotRestart` is `START`—the title page. You can `\renewcommand` it so you get to a slide containing an overview of the presentation.

```
75 \newcommand*\BlogDotRestart{START}
```

5.7 Moving to Next “Slide” (User Level)

`\nextscreenpage{<style>}{<anchor-name>}` puts closing the previous slide and opening the next one—having anchor name `<anchor-name>`—together. `<style>` is for style settings for the next page, made here for choosing between centering the page/slide content and aligning it flush left.

```
76 \newcommand*\nextscreenpage[2]{%
77     \screenbottom{#2}\CLBrk
78     \hrule           \CLBrk
79     \startscreenpage{#1}{#2}}
```

`\nextcenterscreenpage{<anchor-name>}` chooses centering the slide content:

```
80 \newcommand*\nextcenterscreenpage{\nextscreenpage{@align@c}}
```

`\nextnormalscreenpage{<anchor-name>}` chooses flush left on the type area determined by `\typeareawidth`:

```
81 \newcommand*\nextnormalscreenpage{\nextscreenpage{}}
```

5.8 Constructs for Type Area

If you want to get centered titles with `<h2>` etc., you should declare this in `.css` files. But you may consider this way too difficult, and you may prefer to declare this right in the HTML code. That’s what I do! I use `\cheading{<digit>}{<text>}` for this purpose.

```
82 \newcommand*\cheading[1]{\TagSurr{h#1}{@align@c}}
```

`\begin{textblock}{<width>}` opens a `{textblock}` environment. The latter will contain text that will be flush left in a narrower text area—of width `<width>`—than the one determined by `\typeareawidth`. It may be used on “centered” slides. It is made for lists whose entries are so short that the page would look unbalanced under a centered title with the list adjusted to the left of the entire type area. (Thinking of standard L^AT_EX, it is almost the `{minipage}` environment, however lacking the footnote feature, in that respect it is rather similar to `\parbox` which however is not an environment.)

```
83 \newenvironment{textblock}[1]
84     {\startTable{@width{#1}}\starttr\startTd{}}
85     {\endTd\endtr\endTable}
```


5.9 Test for blogdot.cfg

```

86 \InputIfFileExists{blogdot.cfg}{%
87   \typeout{
88     * Using local settings from \string'blogdot.cfg\string' *}%
89   }{}

```

5.10 The End

```

90 \endinput

```

5.11 VERSION HISTORY

```

91 v0.1    2011/09/21f.  started
92          2011/09/25   spacing/padding off
93          2011/09/27   \CLBrk
94          2011/09/30   \BlogDotRestart
95 v0.2    2011/10/03   four possibly independent page margin
96                      parameters; \hvspace moves to texblog.fdf
97          2011/10/04   renewed \body commented out
98          2011/10/07   documentation
99          2011/10/08   added some labels
100         2011/10/10   v etc. in \ProvidesPackage
101

```