

blog.sty

Generating HTML Quickly with T_EX*

Uwe Lück[†]

December 20, 2012

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.—`lnavicol.sty` adds a more **professional** look (towards CMS?), and `blogdot.sty` uses `blog.sty` for HTML **beamer** presentations.

Contents

1	Installing and Usage	4
2	Examples	4
2.1	Hello World!	4
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>	7
3.1	Preliminaries	7
3.1.1	Package File Header (Legalese)	7
3.2	<code>\newlet</code>	8
3.3	Processing	8

*This document describes version [v0.81](#) of `blog.sty` as of 2012/12/20.

[†]<http://contact-ednotes.sty.de.vu>

3.3.1	Requirement	8
3.3.2	Output File Names	8
3.3.3	General Insertions	8
3.3.4	Category Codes etc.	9
3.3.5	The Processing Loop	9
3.3.6	<i>Executing</i> Source File Code Optionally	10
3.3.7	“Ligatures”, Package Options	11
3.3.8	$\langle p \rangle$ from Empty Line, Package Option	12
3.4	General HTML Matters	12
3.4.1	General Tagging	12
3.4.2	Attributes	13
3.4.3	Hash Mark	15
3.4.4	“Escaping” HTML Code for “Verbatim”	15
3.4.5	Head	15
3.4.6	Body	17
3.4.7	Comments	17
3.4.8	CSS	17
3.5	Paragraphs and Line Breaks	17
3.6	Physical Markup (Inline)	18
3.7	Logical Markup	19
3.8	Environments	20
3.9	Links	21
3.9.1	Basic Link Macros	21
3.9.2	Special cases of Basic Link Macros	22
3.9.3	Italic Variants	22
3.9.4	Built Macros for Links to Local Files	22
3.9.5	Built Macros for Links to Remote Files	23
3.10	Characters/Symbols	23
3.10.1	Basic Preliminaries	23
3.10.2	Diacritics	24
3.10.3	Greek	24
3.10.4	Arrows	25
3.10.5	Dashes	25
3.10.6	Spaces	25
3.10.7	Quotes, Apostrophe	26
3.10.8	(Sub- and) Superscript Digits/Letters	27
3.10.9	Math	27
3.10.10	Currencies	29
3.10.11	Other	29
3.11	TeX-related	30
3.11.1	Logos	30
3.11.2	Describing Macros	31
3.12	Tables	31
3.12.1	Indenting	31
3.12.2	Starting/Ending Tables	31
3.12.3	Rows	32

3.12.4	Cells	32
3.12.5	“Implicit” Attributes and a “ \TeX -like” Interface	34
3.12.6	Filling a Row with Dummy Cells	35
3.12.7	Skipping Tricks	35
3.13	Misc	36
3.14	Leaving and HISTORY	36
4	“Pervasive Ligatures” with <code>blogligs.sty</code>	40
4.1	<code>blog</code> Required	40
4.2	Task and Idea	40
4.3	Quotation Marks	41
4.4	HTML Elements	41
4.5	Avoiding “Ligatures” though	41
4.6	The End and HISTORY	42
5	Wiki Markup by <code>markblog.sty</code>	42
5.1	Introduction	42
5.2	Similar Packages	42
5.3	Package File Header	43
5.4	<code>blog</code> Required	43
5.5	Replacement Rules	43
5.6	Connecting to \LaTeX commands	44
5.7	The End and HISTORY	44
6	Real Web Pages with <code>inavicol.sty</code>	45
6.1	<code>blog.sty</code> Required	45
6.2	Switches	45
6.3	Page Style Settings (to be set locally)	45
6.4	Possible Additions to <code>blog.sty</code>	46
6.4.1	Tables	46
6.4.2	Graphics	46
6.4.3	HTTP/Wikipedia tooltips	47
6.5	Page Structure	48
6.5.1	Page Head Row	48
6.5.2	Navigation and Main Row	49
6.5.3	Footer Row	49
6.6	The End and HISTORY	50
7	Beamer Presentations with <code>blogdot.sty</code>	50
7.1	Overview	50
7.2	File Header	52
7.3	<code>blog</code> Required	53
7.4	Size Parameters	53
7.5	(Backbone for) Starting a “Slide”	54
7.6	Finishing a “Slide” and “Restart” (Backbone)	55
7.7	Moving to Next “Slide” (User Level)	56

7.8	Constructs for Type Area	56
7.9	Debugging and .cfs	56
7.10	The End and HISTORY	59

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).² It chooses **source** files and the name(s) for the resulting HTML file(s). It may also need to load local settings, such as `\uselanguage` with the `langcode`³ package and settings for converting the editor's text encoding into the encoding that the head of the resulting HTML file advertises—or into HTML named entities (for me, `atari_ht.fdf` has done this).

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.

2 Examples

2.1 Hello World!

This is the **source** code for a “Hello World” example, in `hellowor.tex`:

```
\ProvidesFile{hellowor.tex}[2012/11/30 hello world source]
\head
\title{Hello world!}
\body
Hello [[world]]!
\finish
```

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

²<http://tug.ctan.org/pkg/nicetext>

³<http://tug.ctan.org/pkg/langcode>

The HTML file `hellowor.htm` is generated from `hellowor.tex` by the following **driver** file `mkhellow.tex`:

```

\ProvidesFile{mkhellow.tex}[2012/11/30 blog demo]
\RequirePackage[ligs,mark]{blog}      %% general HTML generation
\BlogInterceptEnvironments*          %% ... using blogexec.sty
\UseBlogLigs                         %% smart markup
\RequirePackage{texlinks}             %% basic link shorthands
\RequirePackage{langcode}             %% \uselangcode...
\RequirePackage{catchdq}              %% " typographically
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% \input{jobname}                     %% call by "echo"
\newcommand{\htmljob}                 %% choose filename base
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
{hellowor}                            %% "Hello world!"
% {hallow} \uselangcode{de}           %% "Hallo Welt!"
% {markblog}                          %% easy syntax overview
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\ResultFile{\htmljob.htm}
\BlogProcessFinalFile[%\TextCodes    %% encoding settings
                        \catchdqs]    %% " typographically
                        {\htmljob.tex}
\stop                                 %% stop LaTeX run

```

2.2 A Style with a Navigation Column

A style of web pages looking more professional (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. 6. 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”⁴. As the footer is the same on all pages of this style, it is added in the driver file

⁴www.webdesign-bu.de/uwe_lueck/schreibt.html

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 blogdemo/writings, hopefully useful as templates.

2.2.1 Driver File makehtml.tex

```

1  \def \GenDate {2012/08/02}          %% {2012/06/07} {2011/11/01}
   \ProvidesFile{makehtml.tex}
   [\GenDate\space TeX engine for "writings"]
   %% reworked 2012/03/13:
5  \RequirePackage[autopars]{blog}[2011/11/20]    %% auto 2012/08/02
   \BlogInterceptEnvironments*
   \RequirePackage{texlinks,lnavicol}
   \input{atari_ht.fdf}          %% 2012/06/07
   \input{writings.fdf}
10  \NoBlogLigs          %% 2012/03/14 TODO remove HTML comments
   %%%%%%%%%%%
   \input{jobname}
   % \def \htmljob
   % {_sitemap}
15  % {index}          \BlogAutoPars
   % {schreibt} \uselangcode{de} \BlogAutoPars %% mod. 2012/02/04
   %%%%%%%%%%%
   % {about}          \BlogAutoPars
   % {contact}          % \tighttrue
20  % {kontakt} \uselangcode{de}          % \tighttrue
   % {tutor} \uselangcode{de} \BlogAutoPars \deeptrue
   %%%%%%%%%%%
   % {writings} \BlogAutoPars \deeptrue
   % {repres} \BlogAutoPars \deeptrue
25  % {critedl} \BlogAutoPars \deeptrue
   % {ednworks} \BlogAutoPars
   % {public} \BlogAutoPars \deeptrue
   % {texproj} \BlogAutoPars % \deeptrue
   %%%%%%%%%%%
30  \ResultFile{\htmljob\htmakeext}
   \WriteResult\writdoctype          %% TODO
   % \BlogCopyFile[\TextCodes          %% \BlogIntercept:
   \BlogProcessFile[\TextCodes          %% 2012/03/13
35  \MakeActiveDef\{"\catchdq}%      %% TODO attributes!?
   ]{\htmljob.tex}
   \WriteResult{\PAGEFOOT}
   \WriteResult{\indentii\rainermaster}
   \WriteResult{\indentii\}

```

```

40 \WriteResult{\indentii\ueberseeport}      %% TODO BlogLigs!?
    \WriteResult{\PAGEEND}
    \ifdeep \WriteResult{\indenti\vspace{280}} \fi
    \WriteResult{\finish}
    \CloseResultFile
45 \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 Präsentation zweier ETC.

25 \rightpar{\textit{Worms-Pfeddersheim, den 19.~August 2011,\\\Uwe}}
    % \rightpar{\textit{München, den 31.~Juli 2011,\\\Uwe}}
    %% <- TODO VERSION

```

3 The File blog.sty

3.1 Preliminaries

3.1.1 Package File Header (Legalese)

```

1 \ProvidesPackage{blog}[2012/12/20 v0.81 simple fast HTML (UL)]
2 %% copyright (C) 2010 2011 2012 Uwe Lueck,
3 %% http://www.contact-ednotes.sty.de.vu

```

```

4  %% -- author-maintained in the sense of LPPL below.
5  %%
6  %% This file can be redistributed and/or modified under
7  %% the terms of the LaTeX Project Public License; either
8  %% version 1.3c of the License, or any later version.
9  %% The latest version of this license is in
10 %%      http://www.latex-project.org/lppl.txt
11 %% We did our best to help you, but there is NO WARRANTY.
12 %%
13 %% Please report bugs, problems, and suggestions via
14 %%
15 %%      http://www.contact-ednotes.sty.de.vu
16 %%

```

3.2 \newlet

`\newlet<cmd><cmd>` is also useful in surrounding files:

```

17 \newcommand*\newlet}[2]{\@ifdefinable#1{\let#1#2}}

```

3.3 Processing

3.3.1 Requirement

We are building on the `fifinddo` package (using `\protected@edef` for Sec. 3.3.7):

```

18 \RequirePackage{fifinddo}[2011/11/21]

```

3.3.2 Output File Names

`\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`):

```

19 \newcommand*\htmakeext{.html}

```

3.3.3 General Insertions

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

```

20 \newcommand*\CLBrk{{^~J}}

```

`_` is turned into an alias for `\space`, so it inserts a blank space. It even works at line ends, thanks to the choice of `\endlinechar` in Sec. 3.3.4.

```

21 \let\_ \space

```

`\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):


```

22 \@ifdefinable\BlogProvidesFile{%
23     \def\BlogProvidesFile#1[#2]{%
24         <!DOCTYPE html>\CLBrk                %% TODO more!? 2012/09/06
25         \comment{ generated from\CLBrk\CLBrk
26             \ \ \ \ \ \ \ \ #1, #2,\CLBrk\CLBrk
27             \ \ \ \ \ with blog.sty,
28             \isotoday\ }}}}
29 \edef\isotoday{%% texblog 2011/11/02, here 2011/11/20
30     \the\year-\two@digits{\the\month}-\two@digits{\the\day}}

```

([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 `\BlogProcessFile`.

3.3.4 Category Codes etc.

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

```

31 \newcommand*{\BlogCodes}{%                %% 2010/09/07
32     \endlinechar\ %

```

← Comment character to get space rather than `^^M`!—The tilde `~` is active as in Plain TeX too, it is so natural to use it for abbreviating HTML’s ` `!

```

33 %     \catcode'\~\active
34     \MakeActiveDef\~{\&nbsp;}%%%          for \FDpseudoTilde 2012/01/07

```

‘~’ for HTML convenience (cf. Sec. 3.10.7):

```

35     \MakeActiveLet\~\rq                %% actcodes 2012/08/28
36     \BasicNormalCatCodes}
37 % \MakeOther<\ \MakeOther>            %% rm. 2011/11/20

```

3.3.5 The Processing Loop

```
\BlogProcessFile[changes]{source-file}
```

“copies” the TeX source file *source-file* into the file specified by `\ResultFile`.

```

38 \newcommand*{\BlogProcessFile}[2][{}]{%    %% 2011/11/05
39     \ProcessFileWith[\BlogCodes
40         \let\ProvidesFile\BlogProvidesFile %% 2011/02/24
41         \let\protect\empty                %% 2011/03/24
42         \let\@typeset@protect\empty      %% 2012/03/17
43         #1]{#2}{%
44         \IfFDinputEmpty

```

```

45         {\IfFDpreviousInputEmpty
46          \relax
47          {\WriteResult{\ifBlogAutoPars<p>\fi}}}%
48         \BlogProcessLine           %% 2011/11/05
49     }%
50 }

```

fifinddo v0.5 allows the following

```
\BlogProcessFinalFile[<changes>]{<source-file>}
```

working just like `\BlogProcessFile` except that the final `\CloseResultFile` is issued automatically, no more need having it in the driver file.

```

51 \newcommand*{\BlogProcessFinalFile}{%
52     \FinalInputFiletrue\BlogProcessFile}

```

TODO: optionally include `.css` code with `<style>`.

3.3.6 Executing Source File Code Optionally

For v0.7, `\BlogCopyFile` is renamed `\BlogProcessFile`; and in its code, `\CopyLine` is replaced by `\BlogProcessLine`. The purpose of this is supporting `blogexec.sty` that allows intercepting certain commands in the line. We provide initial versions of `blogexec`’s switching commands that allow invoking `blogexec` “on the fly”:

```

53 \newcommand*{\ProvideBlogExec}{\RequirePackage{blogexec}}

```

`dowith.sty` is used in the present package to reduce package code and documentation space:

```

54 \RequirePackage{dowith}
55 \setdo{\providecommand*#1{\ProvideBlogExec#1}}
56 \DoDoWithAllOf{\BlogInterceptExecute \BlogInterceptEnvironments
57              \BlogInterceptExtra \BlogInterceptHash }

```

`\BlogCopyLines` switches to the “copy only” (“compressing” empty lines) functionality of the original `\BlogCopyFile`:

```

58 \newcommand*{\BlogCopyLines}{%
59 %     \let\BlogProcessLine\CopyLine}
60 \def\BlogProcessLine{%           %% 2011/11/21, corr. 2012/03/14:
61     \WriteResult{\ProcessInputWith\BlogOutputJob}}

```

← This is a preliminary support for “ligatures”—see Sec. 3.3.7. `\NoBlogLigs` sets the default to mere copying:

```

62 \newcommand*{\NoBlogLigs}{\def\BlogOutputJob{LEAVE}}
63 \NoBlogLigs

```

TODO more from `texblog.fdf` here, problems with `writings.fdf`, see its `makehtml.tex`

`\BlogCopyLines` will be the setting with pure `blog.sty`:

64 `\BlogCopyLines`

OK, let's not remove `\BlogCopyFile` altogether, rebirth:

65 `\newcommand*{\BlogCopyFile}{\BlogCopyLines\BlogProcessFile}`

3.3.7 “Ligatures”, Package Options

With v0.7, we introduce a preliminary method to use the “ligatures” `--` and `---` with pure expansion. At this occasion, we also can support the notation `...` for `\dots`, as well as arrows (as in `mdocorr.cfg`). Note that this is somewhat **dangerous**, especially the source must not contain “explicit” HTML comment, comments must use `blog.sty's \comment` or the `{commentlines}` environment. Therefore these “ligatures” must be activated explicitly by `\UseBlogLigs`:

66 `\newcommand*{\UseBlogLigs}{\def\BlogOutputJob{BlogLIGs}}`

In order to work inside braces, the source file better should be preprocessed in “plain text mode.” (TODO: Use `\ifBlogLigs`, and in a group use `\ResultFile` for an intermediate `\htmljob.lig`. And TODO: Use `\let\BlogOutputJob`.) On the other hand, the present approach allows switching while processing with `\EXECUTE!` Also, intercepted commands could apply the replacements on their arguments—using `\ParseLigs{<arg>}`:

67 `\newcommand*{\ParseLigs}[1]{\ProcessStringWith{#1}{BlogLIGs}}`

(`\ProcessStringWith` is from `fifinddo`.)—The package `blogligs.sty` described in Sec. 4 does these things in a more powerful way. You can load it by calling `blog.sty's` package option `[ligs]` (v0.8):

68 `\DeclareOption{ligs}{\AtEndOfPackage{\RequirePackage{blogligs}}}`

The replacement chain follows (TODO move to `.cfg`). As opposed to the file `mdocorr.cfg` for `makedoc.sty`, we are dealing with “normal \TeX ” code (regarding category codes, `fifinddo.sty` as of 2011/11/21 is needed for `\protect`). Moreover, space tokens after patterns are already there and need not be inserted after control sequences.

69 `\FDpseudoTilde`

70 `\StartPrependingChain`

71 `\PrependExpandableAllReplacer{blog...}{...}{\protect\dots}`

72 `\PrependExpandableAllReplacer{blog--}{--}{\protect\endash}`

73 `\PrependExpandableAllReplacer{blog---}{---}{\protect\emdash}`

← Cf. thin surrounding spaces with `\enpardash` (`texblog`, maybe *hair space* U+200A instead of thin space), difficult at code line beginnings or endings and when a paragraph starts with an emdash. I.e., perhaps better don't use it if you want to have such spaces.—‘---’ must be replaced before ‘--’!

74 `\PrependExpandableAllReplacer{blog->}{->}{\protect\to}`

75 `\PrependExpandableAllReplacer{blog<-}{<-}{\protect\gets}`

You also could set `\BlogOutputJob` to a later part of the chain, or more globally change the following:

```
76 \CopyFDconditionFromTo{blog<-}{BlogLIGs}
```

The package `markblog.sty` described in Sec. 5 extends this to some markup resembling wiki editing. This package may be loaded by `blog.sty`’s package option `[mark]` (v0.8):

```
77 \DeclareOption{mark}{\AtEndOfPackage{\RequirePackage{markblog}}}
```

3.3.8 `<p>` from Empty Line, Package Option

As in \TeX an empty line starts a new paragraph, we might “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, this sometimes has unwanted effects (comment lines `TODO`), 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`

```
78 \newif\ifBlogAutoPars
79 \newcommand*{\BlogAutoPars}{\BlogAutoParstrue}
80 \newcommand*{\noBlogAutoPars}{\BlogAutoParsfalse}
```

`\BlogAutoPars` is issued by package option `[autopars]`:

```
81 \DeclareOption{autopars}{\BlogAutoPars}
82 \ProcessOptions
```

See Sec. 3.5 for other ways of breaking paragraphs.

3.4 General HTML Matters

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

3.4.1 General Tagging

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

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

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

are used to avoid repeating element names `<elt-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 \LaTeX ’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:

```

83 \newcommand*\SimpleTagSurr}[2]{<#1>#2</#1>}
84 \newlet\STS\SimpleTagSurr                %% 2010/05/23

```

With the space in `\declareHTMLattrib` as of 2012/08/28, we remove the space between #1 and #2. (Doing this by an option may be better [TODO](#); any separate attribute definitions must take care of this.)

```

85 % \newcommand*\TagSurr}[3]{<#1#2>#3</#1>}

```

... undone 2012/11/16, bad with “direct” use of #2 (with attributes not declared):

```

86 \newcommand*\TagSurr}[3]{<#1 #2>#3</#1>}

```

3.4.2 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 T_EX’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:

```

87 \newcommand*\declareHTMLattrib}[1]{%
88   \def\reserved@a{@#1}%
89   \@ifundefined\reserved@a          %% \res... 2012/09/06
90     {\@namedef{@#1}##1{ #1="##1"}} %% space 2012/08/28
91   \@notdefinable}

```

So after `\declareHTMLattrib{<attr>}`, `\@<attr>` is a T_EX 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*:

```

92 \let@class\relax      %% for tab/arr in latex.ltx
93 \let@title\relax      %% for \title in latex.ltx, %% 2011/04/26
94 \DoWithAllOf\declareHTMLattrib{{class}{id}{style}{title}{lang}{dir}}

```

`@type` is quite frequent too:

```

95 \declareHTMLattrib{type}

```

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

```

96 \declareHTMLattrib{href}

```

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

```

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

```

`@content` appears with `\MetaTag` below:

```
98 \declareHTMLattrib{content}
```

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

```
99 \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).

```
100 \PackageWarning{blog}{Redefining \protect\@width}
101 \let\@width\relax
102 \declareHTMLattrib{width}
```

Same with `@height`:

```
103 \PackageWarning{blog}{Redefining \protect\@height}
104 \let\@height\relax
105 \declareHTMLattrib{height}          %% 2010/07/24
```

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

```
106 \newcommand*{\@align@c}{\@align{center}}
107 \newcommand*{\@align@l}{\@align{left}}
108 \newcommand*{\@align@r}{\@align{right}}
109 % \newcommand*{\@align}[1]{ align="#1"}
110 \declareHTMLattrib{align}          %% 2012/09/08
```

`@valign@t`:

```
111 % \newcommand*{\@valign@t}{v\@align{top}} %% 2011/04/24
112 \newcommand*{\@valign@t}{ valign="top"} %% 2012/09/08
```

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

```
113 \declareHTMLattrib{border}          %% 2011/04/24
114 \declareHTMLattrib{cellpadding}     %% 2010/07/18
115 \declareHTMLattrib{cellspacing}     %% 2010/07/18
116 \declareHTMLattrib{colspan}        %% 2010/07/17
117 \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.

3.4.3 Hash Mark

`#` is needed for numerical specifications in HTML, especially colours 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.

```

118 { \MakeOther\# \gdef\#{#}                %% \M... 2011/11/08
119 % \catcode'\&=12 \gdef\AmpMark{&}        %%   rm. 2011/11/08
120 }

... \CompWordMark etc.?
```

3.4.4 “Escaping” HTML Code for “Verbatim”

`\xmltagcode{<chars>}` yields ‘<chars>’:

```
121 \newcommand*\xmltagcode[1]{\code{\lt#1\gt}}
```

`\xmleltcode{<name>}{<content>}` displays the code for an entire `<name>` element containing `<content>` without attributes:

```
122 \newcommand*\xmleltcode[2]{\code{\lt#1\gt#2\lt/#1\gt}}
```

`\xmleltcode{<name>}{<attrs>}{<content>}` displays the code for an entire `<name>` element *with* attribute text ‘<attrs>’ containing `<content>`:

```
123 \newcommand*\xmleltattrcode[3]{\code{\lt#1 #2\gt#3\lt/#1\gt}}
```

`\xmlentitycode{<name>}` yields the code ‘&<name>;’ for an entity with name `<name>`:

```
124 \newcommand*\xmlentitycode[1]{\code{\&#1;}}
```

3.4.5 Head

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

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

`\MetaTag{<inside>}` creates a `<meta>` tag:

```
126 \newcommand*\MetaTag[1]{\indenti<meta #1>}
```

`\charset{<code-page>}`

```

127 \newcommand*\charset[1]{%
128   \MetaTag{ http-equiv="content-type"@content{text/html; #1}}
129   %% <- space 2012/09/08
```

`\metanamecontent{<name>}{<content>}` obviously:

```
130 \newcommand*{\metanamecontent}[2]{%
131     \MetaTag{\@name{#1}\@content{#2}}}
```

`\author{<name>}` and `\date{<date>}` set according metadata, somewhat opposing L^AT_EX (TODO!?):

```
132 \renewcommand*{\author}{\metanamecontent{author}}
133 \renewcommand*{\date}{\metanamecontent{date}}
```

The name of `\metadescription{<text>}` allows using `\begin{description}` (cf. `secrefenv`):

```
134 \newcommand*{\metadescription}{\metanamecontent{description}}
```

`\keywords{<text>}`:

```
135 \newcommand*{\keywords}{\metanamecontent{keywords}}
```

`\robots{<instructions>}`:

```
136 \newcommand*{\robots}{\metanamecontent{robots}}
137     %% #2 juergenf: index, follow, noarchive
```

`\norobots` for privacy (cf. noarchive.net/meta and *Wikipedia*:

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

`\metanamelangcontent{<name>}{<lang>}{<content>}`,
in addition to the above, uses language code `<lang>`:

```
139 \newcommand*{\metanamelangcontent}[3]{%
140     \MetaTag{\@name{#1}\@lang{#2}\@content{#3}}}
```

So there can be language-dependent descriptions and keywords:

`\langdescription{<text>}` and `\langkeywords{<>}`

```
141 \newcommand*{\langdescription}{\metanamelangcontent{description}}
142 \newcommand*{\langkeywords}{\metanamelangcontent{keywords}}
```

`\stylesheet{<media>}{<css>}` uses `<css>.css` for `media="<media>"`:

```
143 \newcommand*{\stylesheet}[2]{%
144     \space\space                                %% 2010/09/10
145     <link rel="stylesheet" media="#1"%
146         \@type{text/css}%                        %% \@type 2011/10/05
147         \@href{#2.css}>>}
```

Alternatively, style declarations may occur in the `<style>` element. It can be accessed by the `{\style}` environment (cf. Sec. 3.8):

```
148 \newenvironment*{\style}[1]
149     {<style\@type{text/css} media="#1">}
150     {</style>}
```

With `\title{<text>}`, `<text>` heads the browser window:

```
151 \renewcommand*{\title}{\space\space\SimpleTagSurr{title}}
```


3.4.6 Body

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

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

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

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

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

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

3.4.7 Comments

`\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.

```
155 \newcommand*{\comment}[1]{<!--#1-->}
156 % \newcommand{\commentlines}[1]{\comment{^^J#1^^J}} %% 2010/05/07
157 % %% <- TODO bzw. \endlinechar='^^J 2010/05/09 back 2010/05/10
158 \newenvironment{commentlines}[1] %% 2010/05/17
159 {<!--#1}
160 {-->}
```

3.4.8 CSS

`\stylespan{<css-style>}{<text>}` applies the CSS styling `<css-style>` to `<text>`:

```
161 \newcommand*{\stylespan}[1]{\TagSurr{span}{\@style{#1}}}
```

Not sure about `<div>` yet ... [TODO](#)

3.5 Paragraphs and Line Breaks

2010/04/28: `
` for manual line breaking can be generated either by

`\newline` or by `\`:

```
162 \renewcommand*{\newline}{<br>}
163 \let\\\newline
```

Automatic insertion of `<p>` tags for starting new paragraphs according to Sec. 3.3.8 has been difficult, especially comment lines so far insert unwanted paragraph breaks ([TODO](#) 2011/11/20). So here are some ways to use L^AT_EX/Plain T_EX commands—or ...:

```
164 % \def\par{<p>} %% + empty lines !? 2010/04/26
```

← difficult with `\stop`; 2010/09/10: `\endgraf` produces `<p>`—[TODO](#)!?

```
165 \renewcommand*{\endgraf}{<p>} %% was </p> 2012/11/19
```

However, I rather have decided for inserting a literal ‘<p>’ using an editor (keyboard) shortcut.

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

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

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

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

3.6 Physical Markup (Inline)

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

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

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

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

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

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

`\textsf{<text>}` chooses some sans-serif:

```
171 \renewcommand*{\textsf}{\stylespan{font-family:sans-serif}}
```

`\textup{<text>}` may undo surrounding slanting or ...:

```
172 \renewcommand*{\textup}{\stylespan{font-style:normal}}
```

`\textcolor{<color>}{<text>}` is from L^AT_EX’s `color` package that we won’t load for generating HTML, so it is “new” here, it is just natural to use it for coloured text. `` is deprecated, use `` instead:

```
173 \newcommand*{\textcolor}[1]{\stylespan{color:#1}}
```

T_EX/L^AT_EX’s `\underbar{<text>}` is redirected to the `<u>` element:

```
174 \renewcommand*{\underbar}{\SimpleTagSurr u}
```

3.7 Logical Markup

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

```
175 \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.

`\code{⟨text⟩}` marks `⟨text⟩` as “code,” just accessing to `<code>` element, while standard L^AT_EX does not provide a `\code` command:

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

`\emph{⟨text⟩}` is L^AT_EX’s command again, but somewhat abused, expanding to `⟨text>`:

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

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

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

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

`\var{⟨symbol(s)⟩}` accesses the `<var>` element:

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

For tagging acronyms, HTML offers the `<acronym>` element, and the TUGboat macros provide `\acro{⟨LETTERS⟩}`. I have used the latter for some time in my package documentations anyway. For v0.7, I add the latter here as an alias for `\acronym{⟨LETTERS⟩}` (supporting both naming policies mentioned in Sec. 3.8):

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

```
181 \newlet\acro\acronym
```

`\newacronym{⟨LETTERS⟩}` saves you from doubling the `⟨LETTERS⟩` when you want to create the shorthand macro `\⟨LETTERS⟩`:

```
182 \newcommand*{\newacronym}[1]{%
183   \expandafter\newcommand\expandafter*\csname#1\endcsname{%
184     \acronym{#1}}}
```

However, `<acronym>` is deprecated. You may use `\abbr{⟨LETTERS⟩}` and `\newabbr{⟨LETTERS⟩}` instead:

```
185 \newcommand*{\abbr}{\SimpleTagSurr{abbr}}    %% 2012/09/13
186 \newcommand*{\newabbr}[1]{%
187   \expandafter\newcommand\expandafter*\csname#1\endcsname{%
188     \abbr{#1}}}
```

3.8 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.)

```
189 \let\begin\@nameuse
190 \def\end#1{\csname end#1\endcsname}
```

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

As formerly with physical markup, 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 ``:

```
191 \newcommand*{\useHTMLelement}{%
192   \ifstar{\@useHTMLelement[</li>]}{\@useHTMLelement}}
193 \newcommand*{\@useHTMLelement}[3][ ]{%
194   \@namedef{#2}{<#3>}%
195   \@namedef{end#2}{#1\CLBrk</#3>}}      %% \CLBrk 2012/04/03
```

Applications:

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

```
196 \useHTMLelement{small}{small}
```

`{center}`:

```
197 % \renewenvironment*{center}{<p align="center">}{</p>}
198 % \renewenvironment*{center}{<p \@align@c>}{</p>}
199 \useHTMLelement{center}{center}
```

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

```
200 \useHTMLelement*{enumerate}{ol}
201 \useHTMLelement*{itemize} {ul}
```

`\begin{enumtype}{⟨type⟩}` starts an enumeration environment with enumeration type `⟨type⟩` which can be one out of `1`, `a`, `A`, `i`, `I` (somewhat resembling the functionality of the `enumerate` package):

```
202 \newenvironment{enumtype}[1]{<ol \@type{#1}>{</ol>}}
```

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:

```
203 \useHTMLelement{verbatim}{pre}
```

`{quote}`:

```
204 \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 (missing again 2011/10/11!?):

```
205 \renewcommand*{\item}{%
206     \indenti</li>\CLBrk                %% 2011/10/11
207     \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?).

```
208 \useHTMLelement{description}{dl}
209 \newcommand*{\ditem}[1]{\indenti<dt>\strong{#1}<dd>}
```

3.9 Links

3.9.1 Basic Link Macros

`\hanc{⟨name⟩}{⟨text⟩}` makes `⟨text⟩` an anchor with HTML label `⟨name⟩` like `hyperref`'s `\hypertarget{⟨name⟩}{⟨text⟩}` (that we actually provide as well, towards printing from the same source):

```
210 \newcommand*{\hanc}[1]{\TagSurr a{\@name{#1}}}
211 \newlet\hypertarget\hanc
```

`\hancref{⟨name⟩}{⟨target⟩}{⟨text⟩}` makes `⟨text⟩` an anchor with HTML label `⟨name⟩` and at the same time a link to `⟨target⟩`:

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

`\href{⟨name⟩}{⟨text⟩}` makes `⟨text⟩` a link to `⟨name⟩` (as with `hyperref`):

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

3.9.2 Special cases of Basic Link Macros

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

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

`\ancref{⟨name⟩}{⟨text⟩}` makes `⟨text⟩` a link to an anchor `⟨name⟩` on the same web page. This is especially useful for a “table of contents”—a list of links to sections of the page. It is just like `hyperref`’s `\hyperlink{⟨name⟩}{⟨text⟩}`:

```
215 \newcommand*{\ancref}[1]{\href{##1}}
216 \newlet\hyperlink\ancref
```

`\autref{⟨text⟩}` makes `⟨text⟩` a link to an anchor named `⟨text⟩` itself:

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

3.9.3 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., `⟨text⟩` 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` → `it` replacement has not been completed yet.)

```
218 % \newcommand*{\emhref}[2]{\href{#1}{\emph{#2}}}
219 \newcommand*{\ithref}[2]{\href{#1}{\textit{#2}}}
220 \newcommand*{\itancref}[2]{\ancref{#1}{\textit{#2}}}% 2010/05/30
221 \newcommand*{\emancref}[2]{\ancref{#1}{\emph{#2}}}
```

3.9.4 Built Macros for Links to Local Files

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

`\fileref{⟨filename-base⟩}`.

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{⟨filename⟩}`, overlooking that it was the same as `\href`.

```
222 % \newcommand*{\FileRef}[1]{\TagSurr a{\@href{#1}}}
223 \newcommand*{\htext}{.htm}                               %% 2011/10/05
224 \newcommand*{\fileref}[1]{\href{#1\htext}}
225 % \newcommand*{\emfileref}[2]{\fileref{#1}{\emph{#2}}}
226 \newcommand*{\itfileref}[2]{\fileref{#1}{\textit{#2}}}
```

`\fileancref{⟨file⟩}{⟨anchor⟩}{⟨text⟩}` links to anchor `⟨anchor⟩` on web page `⟨file⟩`:

```

227 \newcommand*{\fileancref}[2]{%
228   \TagSurr a{\@href{#1\htext{##2}}}}
229 % \newcommand*{\emfileancref}[3]{\fileancref{#1}{#2}{\emph{#3}}}

← 2010/05/31 →

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

3.9.5 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. This may change (as with `blogdot.sty`, 2011/10/12, or more generally with local non-HTML files), so let us have a backbone `\hnewref{<prot>}{<host-path/#frag>}{<text>}` that makes `<text>` a link to `<prot><host-path/#frag>`:

```

231 \newcommand*{\hnewref}[2]{%
232   \TagSurr a{\@href{#1#2" target="_blank"}}}
```

So

`\httpref{<host-path/#frag>}{<text>}`

makes `<text>` a link to `http://<host-path/#frag>`:

```

233 \newcommand*{\httpref}{\hnewref{http://}}
```

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

```

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

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

```

235 \newlet\urlref\urlhttpref
```

... and `\ctanref` has changed its meaning there as of 2011/10/21. `texlinks` sometimes uses a “permanent alias” `\NormalHTTPref` of `\httpref`:

```

236 \newlet\NormalHTTPref\httpref
```

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

```

237 \newcommand*{\httpsref}{\hnewref{https://}}
```

3.10 Characters/Symbols

3.10.1 Basic Preliminaries

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

```

238 \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 ...)

```
239 \newcommand*{\declareHTMlsymbol}[1]{\@namedef{#1}{&#1;}}
```

`\declareHTMlsymbols{<name>}{<list>}` essentially issues

```
\declareHTMlsymbol{<attr>}\declareHTMlsymbols{<list>}
```

while `\declareHTMlsymbols{}` essentially does nothing—great, this is an explanation by recursion!

```
240 \newcommand*{\declareHTMlsymbols}{\DoWithAllOf\declareHTMlsymbol}
```

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

```
241 \newcommand*{\renderHTMlsymbol}[2]{\renewcommand*{#1}{&#2;}}
```

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

```
242 \renderHTMlsymbol{\&}{amp}
```

```
243 \let\%\PercentChar
```

3.10.2 Diacritics

For the difference between diacritic and accent, see *Wikipedia*.

HTML entities `é` (é), `ç` (ç), `ô` (ô) etc. can be accessed by T_EX's accent commands `\'`, `\c`, `\^`, `\'`, `\"`:

```
244 % \declareHTMlsymbol{eacute}
245 % \declareHTMlsymbol{ocirc}
246 \renewcommand*{\'}[1]{&#1acute;}
247 \renewcommand*{\c}[1]{&#1cedil;}
248 \renewcommand*{\^}[1]{&#1circ;}
249 \renewcommand*{\'}[1]{&#1grave;}
250 \renewcommand*{\"}[1]{&#1uml;}
```

former `\uml{<char>}` is obsolete, use `\"<char>` (or `\"<char>`) instead.

3.10.3 Greek

```
251 \declareHTMlsymbols{{Alpha}{alpha}                                %% 2012/01/06
252 {Beta}{beta}{Gamma}{gamma}{Delta}{delta}{Epsilon}{epsilon}
253 {Zeta}{zeta}{Eta}{eta}{Theta}{theta}{Iota}{iota}{Kappa}{kappa}
254 {Lambda}{lambda}{My}{my}{Ny}{ny}{Xi}{xi}{Omicron}{omikron}
255 {Pi}{pi}{Rho}{rho}{Sigma}{sigma}{sigmaf}{Tau}{tau}
256 {Upsilon}{upsilon}{Phi}{phi}{Chi}{chi}{Psi}{psi}
257 {Omega}{omega}                                                    %% render -> declare 2011/02/26
258 {thetasym}{upsih}{piv} }
```


3.10.4 Arrows

—somewhat completed 2012/07/25.

`\downarrow`, `\leftarrow`, `\leftrightarrow`, `\rightarrow`, `\uparrow`:

```
259 \renderHTMLsymbol {\downarrow}    {darr}    %% 2010/09/15
260 \renderHTMLsymbol {\leftarrow}    {larr}
261 \renderHTMLsymbol {\leftrightarrow}{harr}
262 \renderHTMLsymbol {\rightarrow}    {rarr}
263 \renderHTMLsymbol {\uparrow}      {uarr}    %% 2010/09/15
```

Aliases `\gets` and `\to` were implemented first as stand-alones, now are treated by `\let`:

```
264 \let \gets \leftarrow
265 \let \to  \rightarrow
```

`\Downarrow`, `\Leftarrow`, `\Leftrightarrow`, `\Rrightarrow`, `\Uparrow`
(i.e., double variants):

```
266 \renderHTMLsymbol {\Downarrow}    {dArr}
267 \renderHTMLsymbol {\Leftarrow}    {lArr}
268 \renderHTMLsymbol {\Leftrightarrow}{hArr}
269 \renderHTMLsymbol {\Rrightarrow}    {rArr}
270 \renderHTMLsymbol {\Uparrow}      {uArr}
```

`\crarrow` accesses HTML’s `crarr` entity (symbol for return key), named “downwards arrow with tip leftwards” in Unicode (U+21b2):

```
271 \newcommand*{\crarrow}{&crarr;}          %% 2012/09/13
```

3.10.5 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)TeX, so here I adopt a *third* policy besides (i) and (ii) earlier; cf. LaTeX’s `\textendash` and `\textdash`.—`\newcommand` does not accept macros whose names start with `end`, so: `\endash`, `\emdash` ...

```
272 \def          \endash  {\&dash;}          %% \end... illegal
273 \newcommand*{\emdash} {\&mdash;}
```

3.10.6 Spaces

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

```
274 \renderHTMLsymbol{\enspace}{ensp}
275 \renderHTMLsymbol{\quad}{emsp}
276 \renewcommand*  {\qquad}{\quad\quad}
```

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

```

277 % \renderHTMlsymbol{\thinspace}{thinsp}
278 % \renderHTMlsymbol{\,} {thinsp}
279 \declareHTMlsymbol{thinsp}
280 \renderHTMlsymbol{\thinspace}{\#8239}
281 \renderHTMlsymbol{\,} {\#8239}

```

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

```

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

```

3.10.7 Quotes, Apostrophe

`\lq`, `\rq`

```

283 \renderHTMlsymbol{\lq} {lsquo}
284 \renderHTMlsymbol{\rq} {rsquo}

```

In order to use the right single quote for the HTML apostrophe, we must save other uses before. `\urlapostr` is the version of the right single quote for URLs of Wikipedia articles:

```

285 % \newcommand*{\screenqtd}[1]{‘#1’} %% rm. 2011/11/08
286 \newcommand*{\urlapostr} {’} %% 2010/09/10

```

The actual change of `’` is in `\BlogCodes` (Sec. 3.3.4).

`\bdquo` (bottom), `\ldquo`, `\rdquo`, `\sbquo` (single bottom):

```

287 \declareHTMlsymbol{bdquo} %% 2011/09/23
288 \declareHTMlsymbols{{ldquo}{rdquo}}
289 \declareHTMlsymbol{sbquo} %% 2010/07/01
290 \declareHTMlsymbols{{laquo}{raquo}}

```

Angled quotes `\laquo` and `\raquo` as well as their “single” versions `\lsaquo` and `\rsaquo`:

```

291 \declareHTMlsymbols{{laquo}{lsaquo}{raquo}{rsaquo}} %% 2012/10/25

```

As of 2012/09/17, `\asciidq` and `\asciidqtd{\no-dqs}` (e.g., for attributes after `\catchdqs` or typesetting code) move to package `catchdq.sty` in the `catcodes` bundle.

`\quot` accesses the same symbol in HTML’s terms (e.g., for displaying code):

```

292 \declareHTMlsymbol{quot} %% 2012/01/21

```

`\endqtd{\text}` quotes in the English style using double quote marks, `\enqtd{\text}` uses single quote marks instead, `\dedqtd{\text}` quotes in German style, `\quoted{\text}` uses straight double quotation marks. Settings from the `langcode` package may need to be overridden. (A warning might be nice then [TODO](#))

```

293 \def\endqtd#1{\ldquo#1\rdquo}
294 \def\enqtd #1{\lq#1\rq} %% 2010/09/08
295 \def\dedqtd#1{\bdquo#1\ldquo}
296 \def\deqtd #1{\sbquo#1\lq} %% corr. 2012/10/25
297 \newcommand*\quoted{[1]{\quot#1\quot} %% 2012/01/21

```

`\quoted{text}` surrounds *text* with “straight” single quotation marks, useful for other kinds of quoting in computer code:

```

298 \newcommand*\squoted{[1]{\urlapostr#1\urlapostr} %% 2012/01/21

```

3.10.8 (Sub- and) Superscript Digits/Letters

As Plain T_EX and L^AT_EX provides an alias `\sp` for `^`, I use `\spone`, `\sptwo`, `\spthree`, `\spa`, and `\spo` for superscript 1, 2, 3, ‘a’, and ‘o’:

```

299 \newcommand*\spone{\&sup1;}
300 \newcommand*\sptwo{\&sup2;}
301 \newcommand*\spthree{\&sup3;}
302 \newcommand*\spa{\&ordf;}
303 \newcommand*\spo{\&ordm;}

```

For slanted fractions, I think of xfrac’s `\sfrac{⟨numerator⟩}{⟨denominator⟩}`. `\sfrac{1}{2}`, `\sfrac{1}{4}`, and `\sfrac{3}{4}` work so far:

```

304 \newcommand*\sfrac{[2]{\&frac#1#2;}

```

3.10.9 Math

Symbols (T_EX math type “Ord”)—`\aleph`:

```

305 \renderHTLsymbol{\aleph}{alefsym}

```

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.

```

306 \newcommand*\degrees{\&deg;}

```

We stick to T_EX’s `\emptyset`

```

307 \renderHTLsymbol{\emptyset}{empty} %% 2011/04/14

```

`\exists` and `\forall`:

```

308 \renderHTLsymbol{\exists}{exist}
309 \declareHTLsymbol{forall}

```

`\prime` can be used for minutes, `\Prime` for seconds:

```

310 \renderHTLsymbol{\prime}{prime} \declareHTLsymbol{Prime}

```

Relations Because `<` and `>` are used for HTML’s element notation, we provide aliases `\gt`, `\lt` for mathematical `<` and `>`—and for reference to HTML (or just XML) code (see Sec. 3.4.4):

```
311 \declareHTMLsymbols{{gt}{lt}}
```

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

```
312 \declareHTMLsymbols{{ge}{le}{ne}}
```

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

```
313 \let\geq\ge \let\leq\le \let\neq\ne
```

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

```
314 \declareHTMLsymbol{sub}                %% 2011/04/04
315 \let\subset\sub                        %% 2011/05/08
316 \declareHTMLsymbol{sube}              %% 2011/03/29
317 \let\subseteq\sube                    %% 2011/05/08
```

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

```
318 \renderHTMLsymbol{\langle}{lang}
319 \renderHTMLsymbol{\rangle}{rang}
```

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

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

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

```
321 \begingroup
322   \Delimiters\[\] \gdef\{{\} \gdef\}{\}]
323 \endgroup
```

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

```
324 \renderHTMLsymbol{\ast}{lowast}        %% 2011/03/29
```

`\pm` renders the plus-minus symbol:

```
325 \renderHTMLsymbol{\pm}{plusmn}
```

TeX and HTML agree on `\cap`, `\cup`, and `\times`: 2011/05/08 2011/04/04

```
326 \declareHTMLsymbols{{cap}{cup}{times}} %% 2012/01/06
```

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

```
327 \declareHTMLsymbol{minus}              %% 2011/03/31
```

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

```
328 \renderHTMlsymbol{\circ}{\#x2218}           %% 2011/04/28
329 \renderHTMlsymbol{\cdot}{\middot}          %% 2011/05/07
```

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

```
330 \declareHTMlsymbol{sdot}                   %% 2011/05/08
```

Operators `\prod`, `\sum`:

```
331 \renderHTMlsymbol{\prod}{product}
332 \declareHTMlsymbol{sum}
```

3.10.10 Currencies

`\cent`, `\currency`, `\euro`, `\pound`, `\yen`:

```
333 \declareHTMlsymbols{{cent}{currency}{euro}{pound}{yen}}
```

You get the \$ symbol simply by `\$`.

3.10.11 Other

The tilde `\~` is used for its wonderful purpose, by analogy to T_EX(`\TODO` overridden by `\FDpseudoTilde`):

```
334 \renderHTMlsymbol{\~}{\nbsp}
```

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

```
335 { \MakeOther\~ \gdef\tilde{\~} \gdef\tildechar{\~}}
```

Horizontal ellipsis: `\dots` ...

```
336 \renderHTMlsymbol {\dots} {\hellip}
```

`\ss` from Plain T_EX and L^AT_EX is used for the “s-z ligature”, the German “sharp s”:

```
337 \renderHTMlsymbol{\ss}{szlig}
```

Plain T_EX’s and L^AT_EX’s `\-` becomes a soft hyphen:

```
338 \renderHTMlsymbol{\-}{shy}
```

`\copyright`:

```
339 \renderHTMlsymbol{\copyright}{copy}
```

`\bullet`

```
340 \renderHTMlsymbol{\bullet}{bull}
```

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

```
341 \renderHTMlsymbol{\S}{sect}
```

`\dagger`, `\ddagger`:

```
342 \renderHTMlsymbol{\dagger}{dagger}
```

```
343 \renderHTMlsymbol{\ddagger}{Dagger}
```

`\P` renders the paragraph sign or pilcrow:

```
344 \renderHTMlsymbol{\P}{para}
```

Sometimes (due to certain local settings) the notations `&&\characters`; or `&&&\number`; (for Unicode) may not be available. We provide

```
\htmlentity{\characters}
```

as well as

```
\unicodeentity{\decimal}
```

and

```
\unicodehexentity{\hexadecimal}
```

for such situations:

```
345 \newcommand*\htmlentity[1]{&#1;}
```

```
346 \newcommand*\unicodeentity[1]{&##1;}
```

```
347 \newcommand*\unicodehexentity[1]{&\#x#1;}
```

3.11 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.

3.11.1 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.

```
348 \newlet\prg\@firstofone
349 \newcommand*\BibTeX{\prg{BibTeX}} %% 2010/09/13
350 \renewcommand*\TeX{\prg{TeX}}
351 \renewcommand*\LaTeX{\prg{LaTeX}}
352 \newcommand*\allTeX{\prg{(La)TeX}} %% 2010/10/05
353 \newcommand*\LuaTeX{\prg{LuaTeX}}
354 \newcommand*\pdfTeX{\prg{pdfTeX}}
355 \newcommand*\XeTeX{\prg{XeTeX}} %% 2010/10/09
356 \newcommand*\TeXbook{\TeXbook} %% 2010/09/13
```

3.11.2 Describing Macros

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

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

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

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

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

`\metavar{\<name>}`:

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

3.12 Tables

I am not so sure about this section ...

3.12.1 Indenting

There are three levels of indenting:

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

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

```
360 {\catcode'\ =12%% 2010/05/19
```

```
361 \gdef\indenti{ }\gdef\indentii{ }\gdef\indentiii{ }}
```

3.12.2 Starting/Ending Tables

`\startTable{\<attributes>}` and `\endTable` have been made for appearing in different macros, such as in the two parts of a `\newenvironment`:

```
362 \newcommand*\startTable[1]{<table #1>}
```

```
363 \def\endTable{</table>}
```

`\@frame@box` among the `\startTable \<attributes>` draws a frame around the table, `\@frame@groups` separates “groups” by rules:

```
364 \newcommand*\@frame@box{\@frame{box}}
```

```
365 \newcommand*\@frame@groups{\@frame{groups}}
```

`\begin{allrulestable}{\<cell-padding>}{\<width>}` starts a table environment with all possible rules and some code cosmetic. `\<width>` may be empty ...

```
366 \newenvironment{allrulestable}[2]
```

```
367 {\startTable{\@cellpadding{#1} \@width{#2}}
```

```
368 \frame@box\ rules="all"\CLBrk %% \ 2011/10/12
```

```
369 \tbody} %% <- tbody 2011/10/13, '\ ' 2011/11/09 ->
```

```
370 {\ \endtbody\CLBrk\endTable}
```

`<tbody>...</tbody>` seemed to be better with `\HVspace` for `blogdot.sty`, so it gets an environment `{tbody}` (i.e., macros `\tbody` and `\endtbody`):

```
371 \useHTMLElement{tbody}{tbody}
```

3.12.3 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 ...

`\begin{TableRow}{\langle comment \rangle}{\langle attributes \rangle}`

starts an environment producing an HTML comment $\langle comment \rangle$ and a table row with attributes $\langle attributes \rangle$, including code cosmetic.

```
372 \newenvironment*{TableRow}[2]{%% lesser indentation 2011/04/25
373   \ \comment{ #1 }\CLBrk
374   \indent<tr #2>%
375   }{%
376   \indent\endtr}                                %% \endtr 2011/11/08
```

`\begin{tablecoloredrow}{\langle comment \rangle}{\langle background-color \rangle}`

is a special case of `{TableRow}` where `@bgcolor` is the only attribute:

```
377 \newenvironment{tablecoloredrow}[2]
378   {\TableRow{#1}{\@bgcolor{#2}}}
379   {\endTableRow}
```

`\begin{tablecoloredboldrow}{\langle comment \rangle}{\langle background-color \rangle}`

is like `{tablecoloredrow}` except that content text is rendered in boldface (TODO horizontal centering?):

```
380 \newenvironment{tablecoloredboldrow}[2]                %% 2011/11/03/08
381   {\TableRow{#1}{\@bgcolor{#2}
382     \@style{font-weight:bold}}}
383   {\endTableRow}
```

`\begin{tablerow}{\langle comment \rangle}` is a special case of `{TableRow}` where the only attribute yields “top” vertical alignment (TODO strange):

```
384 \newenvironment{tablerow}[1]{\TableRow{#1}{\@valign@t}}
385   {\endTableRow}
```

`\starttr` and `\endtr` delimit a row; these commands again have been made for appearing in different macros. There is *no* code indenting, probably for heavy table nesting where indenting was rather useless (? TODO only in `texblog.fdf`? there indents would have been useful).

```
386 \newcommand*{\starttr}{<tr>}
387 \def\endtr{</tr>}
```

3.12.4 Cells

`\simplecell{\langle content \rangle}` produces the most *simple* kind of an HTML table cell:

```
388 \newcommand*{\simplecell}{\SimpleTagSurr{td}}    %% 2010/07/18
```

`\TableCell{\langle attributes \rangle}{\langle content \rangle}` produces the most *general* kind of a cell, together with a code indent:


```

389 \newcommand*{\TableCell}[2]{\indentiii\startTd{#1}#2\endTd}

\colorwidthcell{<color>}{<width>}{<content>}} uses just the @bgcolor and
the @width attribute:

390 \newcommand*{\colorwidthcell}[2]{\TableCell{\@bgcolor{#1}\@width{#2}}{<content>}}

\tablewidthcell{<color>}{<width>}{<content>}} uses just the @bgcolor and
the @width attribute:

391 \newcommand*{\tablewidthcell}[1]{\TableCell{\@width{#1}}{<content>}}

\tablecell{<content>}} is like \simplecell{<content>}, except that it has a
code indent:

392 \newcommand*{\tablecell}{\TableCell{}}

\tableCell{<content>}} is like \tablecell{<content>}, except that the con-
tent <content> is horizontally centered. The capital C in the name may be
considered indicating “centered”:

393 \newcommand*{\tableCell}{\TableCell\@align@c}

Idea: use closing star for environment variants!?

\begin{bigtablecell}{<comment>}} starts an environment yielding a ta-
ble cell element without attributes, preceded by a HTML comment <comment>
unless <comment> is empty. At least the HTML tags are indented:

394 \newenvironment{bigtablecell}[1]{\BigTableCell{#1}{}}
395                                     {\endBigTableCell}
396 %                                     {\ifx\#1\%                               %% 2010/05/30
397 %                                     \indentii\ \comment{#1}\CLBrk
398 %                                     \fi
399 %                                     \indentiii<td>}
400 %                                     {\indentii</td>}                               %% !? 2010/05/23

\begin{BigTableCell}{<comment>}{<attributes>}}
is like \begin{bigtablecell}{<comment>}} except that it uses attributes
<attributes>:

401 \newenvironment{BigTableCell}[2]
402     {\ifx\#1\%\\indentii\ \comment{#1}\CLBrk\fi
403     \indentiii\startTd{#2}}
404     {\indentii\endTd}                               %% TODO indent? 2010/07/18

\startTd{<attributes>}} and \endTd delimit a cell element and may appear in
separate macros, e.g., in an environment definition. There is no code cosmetic.
And finally there is \StartTd that yields less confusing code without attributes:

405 \newcommand*{\startTd}[1]{<td #1>}
406 \newcommand*{\StartTd}{<td>}                               %% 2011/11/09
407 \def\endTd{</td>}

\emptycell uses <td /> instead of <td></td> for an empty cell:

408 \newcommand*{\emptycell}{<td />}                               %% 2011/10/07

```

3.12.5 “Implicit” Attributes and a “T_EX-like” Interface

After some more experience, much musing, and trying new tricks, I arrive at the following macros (v0.7). (i) When a page or a site has many tables that use the same attribute values, these should not be repeated for the single tables, rather the values should be invoked by shorthand macros, and the values should be determined at a single separate place. We will have `\stdcellpadding`, `\stdtableheadcolor` and `\stdtableheadstyle`. (ii) As with T_EX, `\cr` should suffice to *close* a *cell* and a *row*, and then to *open* another *row* and its first *cell*. And there should be a single command to close a cell within a row and open a next one.

We use `\providecommand` so the user can determine the values in a file for `blog` where `blogexec` is loaded later. `\stdcellpadding` should correspond to the CSS settings, the value of 6 you find here is just what I used recently.

```
409 \providecommand*\stdcellpadding{6}
```

For `\stdtableheadcolor`, I provide a gray, `#EEEEEE`, that the German Wikipedia uses for articles about networking protocols (unfortunately, it doesn't have a CSS-3X11 color name):

```
410 \providecommand*\stdtableheadcolor{\#EEEEEE}
```

`\stdtableheadstyle` demands a boldface font. In general, it is used for the `@style` attribute:

```
411 \providecommand*\stdtableheadstyle{font-weight:bold}
```

`\begin{stdallrulestable}` starts an `{allrulestable}` environment with “standard” cell padding and empty width attribute, then opens a “standard” row element with a “standard” comment as well as a cell:

```
412 \newenvironment{stdallrulestable}{%
413   \allrulestable{\stdcellpadding}{}\CLBrk
414   \TableRow{standard all-rules table}%
415   {\@bgcolor{\stdtableheadcolor}
416   \@style{\stdtableheadstyle}}\CLBrk
417   \indentii\StartTd
```

`\end{stdallrulestable}` will provide closing of a cell and a row, including a code cosmetic:

```
418   }\indenti\endTd\CLBrk\endTableRow\CLBrk
419   \endallrulestable}
```

`\endcell` closes a cell and opens a new one. The idea behind this is that an active character will invoke it. The name is inspired by `\endgraf` and `\endline` from Plain T_EX and L^AT_EX (`\newcommand` does not work with `\end...`):

```
420 \def\endcell{\endTd\StartTd}
```

Plain TeX's and L^AT_EX's `\cr` and `\endline` are redefined for closing and opening rows and cells, including code cosmetic:

```
421 \renewcommand*{\cr}{\indentii\endTd\CLBrk\indenti\endtr\CLBrk
422 \indenti\startTR\CLBrk\indentii\StartTd}
423 \let\endline\cr
```

`\startTR` is a hook defaulting to `\starttr`:

```
424 \newlet\startTR\starttr
```

3.12.6 Filling a Row with Dummy Cells

These macros were made, e.g., for imitating a program window with a title bar (spanning something more complex below), perhaps also for a Gantt chart. `\FillRow{}{<attributes>}` produces a cell without text, spanning `` columns, with additional attributes `<attributes>`.

```
425 \newcommand*{\FillRow}[2]{\indentiii\startTd{\@colspan{#1} #2}\endTd}
```

`\fillrow{}` instead only uses the `@colspan` attribute:

```
426 \newcommand*{\fillrow}[1]{\FillRow{#1}{}}
```

`\fillrowcolor{}{<color>}` just uses the `@colspan` and the `@bgcolor` attributes:

```
427 \newcommand*{\fillrowcolor}[2]{\FillRow{#1}{\@bgcolor{#2}}}
```

3.12.7 Skipping Tricks

`\HVspace{<text>}{<width>}{<height>}` may change, needed for `blogdot.sty` but also for `\vspace{<height>}` with `texblog`. It is now here so I will be careful when I want to change something. `<tbody>` improved the function of `\HVspace` constructions as link text with `blogdot.sty`.

```
428 \newcommand*{\HVspace}[3]{%
429 \CLBrk
430 \startTable{\@width{#2} \@height{#3}
431 \@border{0}
432 \@cellpadding{0} \@cellspacing{0}}%
433 \tbody
434 \CLBrk %% 2011/10/14
435 \tablerow{HVspace}% %% 2011/10/13
```

← inserting text at top for `blogdot` attempts—that finally did not help anything (2011/10/15) →

```
436 \simplecell{#1}%
437 \endtablerow %% 2011/10/13
438 \CLBrk %% 2011/10/14
439 \endtbody
440 \endTable
441 \CLBrk}
```

```

\hvspace{<width>}{<height>} ...:
442 \newcommand*{\hvspace}{\HVspace{}}

\vspace{<height>} ... (TODO: {0}!?):
443 \renewcommand*{\vspace}[1]{\hvspace{}{#1}}

```

3.13 Misc

\TeX 's `\hrule` (rather deprecated in \LaTeX) is redefined to produce an HTML horizontal line:

```
444 \renewcommand*{\hrule}{<hr>}
```

For references, there were

```

445 % \catcode'\^=\active
446 % \def^#1{\SimpleTagSurr{sup}{#1}}

```

and

```
447 % \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.

3.14 Leaving and HISTORY

```

448 \endinput
449 VERSION HISTORY
450 v0.1 2010/08/20 final version for DFG
451 v0.2 2010/11/08 final documentation version before
452 moving some functionality to 'fifinddo'
453 v0.3 2010/11/10 removed ^^J from \head
454 2010/11/11 moving stuff to fifinddo.sty; \BlogCopyFile
455 2010/11/12 date updated; broke too long code lines etc.;
456 \CatCode replaced (implemented in niceverb only);
457 \ifBlogAutoPars etc.
458 2010/11/13 doc: \uml useful in ...; \texcs
459 2010/11/14 doc: argument for {commentlines},
460 referring to environments with curly braces,
461 more on \ditem
462 2010/11/15 TODO: usage, templates
463 2010/11/16 note on {verbatim}
464 2010/11/23 doc. corr. on \CtanPkgRef
465 2010/11/27 "keyword"; \CopyLine without 'fd'
466 2010/12/03 \emhttpref -> \ithttpref
467 2010/12/23 '%' added to \texhaxpref
468 2011/01/23 more in \Provides...
469 2011/01/24 updated copyright; resolving 'td' ("today")
470 JUST STORED as final version before texlinks.sty

```

```

471 v0.4      2011/01/24  moving links to texlinks.sty
472 v0.41     2011/02/07  \NormalHTTTPref
473          2011/02/10  refined call of 'texlinks'
474 part of MOREHYPE RELEASE r0.3
475 v0.5      2011/02/22  \BlogProvidesFile
476          2011/02/24  ... in \BlogCopyFile
477          2011/02/25  ordering symbols
478          2011/02/26  subsection Greek; note on \declareHTMLsymbol
479          2011/03/04  diacritics
480          2011/03/06  \cs
481          2011/03/09  \var
482          2011/03/16  \robots
483          2011/03/19  doc. \fileancref arg.s corr.
484          2011/03/29  \Sigma, ...
485          2011/03/31  \minus
486          2011/04/04  \times, \sub, \delta
487          2011/04/11  Greek completed
488          2011/04/14  \emptyset
489          2011/04/22  \deqtd
490          2011/04/24  doc.: folding, \stylesheet, ordered "tables";
491                      @border, @align, @valign
492          2011/04/25  lesser indentation with TableRow
493          2011/04/26  \,, \thinspace, \@title; doc. \@name
494          2011/04/28  [\circ] PROBLEM still
495          2011/04/29  \rightitpar
496          2011/05/07  \cdot
497          2011/05/08  extended doc. on math symbols; \sdot;
498                      \ast replaces \lowast; \subset, \subseteq;
499                      \angled
500          2011/05/09  \euro
501          2011/05/11  |\geq| etc.; new section "logical markup"
502          2011/05/12  corr. doc. \heading
503          2011/05/14  right mark of \deqtd was rsquo instead of lsquo!
504          2011/05/18  \S and note on \StoreOtherCharAs
505          2011/06/27  \httpsref; doc: \acro
506          2011/07/22  \thinspace vs. \thinsp; 'fifinddo's
507          2011/07/25  "todo" on \description
508          2011/08/18f.removing \FileRef, 0.42-> 0.5
509          2011/08/31  clarified use of \urlapostr
510 part of MOREHYPE RELEASE r0.4
511 v0.6      2011/09/08  doc. uses \HTML, \lq/\rq with &circ;,
512                      doc. fix 'mult-'; \degrees
513          2011/09/21  \acronym
514          2011/09/22  \metavar; TODO \glqq...
515          2011/09/23  \bdquo
516          2011/09/25  doc. 'Characters/Symbols'; \figurespace
517          2011/09/27  "universal" attributes completed, reworked doc.
518          2011/09/30  end lists with </li>
519          2011/10/01  \dagger, \ddagger
520          2011/10/04  \item includes </li> [2011/10/11: ???]

```

```

521      2011/10/05 {style}; doc. \acronym -> \acro, \pagebreak,
522                rm. \description; {center} accesses <center>,
523                \useHTMLEnvironment replaces \declareHTMLelement
524                and \renderHTMLelement, message "generating"
525      2011/10/07 \emptycell
526      2011/10/10 doc.: page breaks, $$->\[/\]
527 part of MOREHYPE RELEASE r0.5
528 v0.61 2011/10/11 </li> in \item again, \Provides... v wrong
529      2011/10/12 \hnewref, '\ ' in allrulestable
530      2011/10/14 \CLBrk's
531      2011/10/15 doc. note on \HVspace/blogdot
532 part of MOREHYPE RELEASE r0.51
533 v0.62 2011/10/16 \hyperlink, \hypertarget; doc. fixes there
534      2011/10/20 \textcolor by <span>, \textsf
535      2011/10/21 \ctanref now in texlinks.sty;
536                doc.: grammar with 'that'
537      2011/10/22 \BlogCopyFile message removed
538 part of MOREHYPE RELEASE r0.52
539 v0.7  2011/11/03 {tablecoloredboldrow}
540      2011/11/05 \ContentAtt -> \@content,
541                \BlogCopyFile -> \BlogProcessFile (blogexec),
542                doc. different \pagebreak's
543      2011/11/06 run \BlogCopyLines, doc. \[...\]
544      2011/11/07 \ProvideBlogExec
545      2011/11/08 \endtr in \endTableRow, using \MakeOther,
546                right quote change moves to \BlogCodes,
547                \BlogInterceptHash; rm. \AmpMark & doc. about it,
548                mod. on #; doc. for tables; start doc. "implicit"
549                table attributes and "TeX-like" interface
550      2011/11/09 \tablecolorcell(?); cont. "implicit" etc.;
551                \StartTd
552      2011/11/20 \isotoday, \BlogProcessFinalFile,
553                catcodes of '<' '>' untouched; restructured,
554                structured processing, misc -> ordinary
555      2011/11/21 BlogLIGs
556      2011/11/23 \xmltagcode, \xmlentitycode, \c;
557                doc: <p>, \secref, \pagebreak
558      2011/11/24 doc: example results for diacritics
559      2011/11/27 \ParseLigs; doc. rm. \pagebreak
560      2011/12/12 \title uses \SimpleTagSurr
561      2011/12/19 doc. fix {tablerow}
562      2011/12/21 \asciidq, \asciidqtd
563      2012/01/06 \acro; using dowith.sty (\declareHTMLsymbols);
564                doc.: cross-referring for naming policies
565      2012/01/07 \MakeActiveDef~ for \FDpseudoTilde
566      2012/01/11 (C)
567      2012/01/21 \quot, \quoted. \squoted
568      2012/02/04 \newacronym
569      2012/03/14 removed hidden and another comment with
570                \BlogCopyLines, fixed latter, TODO on \NoBlogLigs

```

```

571      2012/03/17 tweaked \@typeset@protect for \EXECUTE
572      2012/03/30 space in stdallrules... after @bgcolor
573      2012/04/03 \CLBrk in \@useHTMLelement
574      2012/04/09 \htmlentity, \unicodeentity
575      2012/05/13 \ss; better comment on \uml;
576      #EEEEEE not "web-safe"
577      2012/05/15 xEDIT folding in tables section
578 part of MOREHYPE RELEASE r0.6
579 v0.8 2012/06/07 \underbar
580      2012/07/25 arrows completed [no: 2012/09/13];
581      doc. "police" -> "policy"
582      2012/07/30 \spanstyle, applied; doc. \pagebreak
583      2012/08/01 \textup
584      2012/08/02 doc. corr. braces for \DeclareHTMLsymbols
585      2012/08/06 sec. currencies
586      2012/08/07 divided math section, using \declareHTMLsymbols,
587      various additional symbols
588      2012/08/23 \startTR
589      2012/08/28 \MakeActiveLet\`\'rq with 'actcodes.sty',
590      attributes start with space
591      2012/09/02 about -> around
592      2012/09/06 Content-T -> content-t - bugfix?,
593      \BlogProvidesFile with DOCTYPE, some attribute
594      lists rely on space from \declareHTMLattrib,
595      there another \reserved@a;
596      "Head": \metanamecontent, \metanamecontent
597      2012/09/07 "Head": \author, \date, \metadescription,
598      \keywords; lang variants
599      2012/09/08 \TagSurr and \MetaTag without space,
600      \declareHTMLattrib{align}, \@valign@t adjusted;
601      \pagebreak[3]
602      2012/09/13 \crarrow, "Fonts" -> "Physical markup" etc.,
603      \abbr, \newabbr
604      2012/09/14 \xmleltcode, \xmleltattrcode; el-name -> elt-name
605      2012/09/17 \asciidq + \asciidqtd move to 'catchdq.sty'
606      2012/10/03 \newlet;
607      doc.: label process -> catcodes, using \secref
608      moved \ast; \exists, \forall
609      2012/10/24 quotes: completed, override 'langcode.sty'
610      2012/10/25 using \DeclareHTMLsymbols for quotes, corr. there,
611      \spone etc., \sfrac
612      2012/10/28 spanstyle -> stylespan
613      2012/11/16 \TagSurr and \MetaTag with space again
614      2012/11/19 \endgraf -> <p>
615      2012/11/29 'blogligs.sty', 'markblog.sty' ([ligs], [mark])
616 part of MOREHYPE RELEASE r0.7
617 v0.81 2012/12/20 \-, {enumtype}
618 part of MOREHYPE RELEASE r0.8
619

```

4 “Pervasive Ligatures” with **blogligs.sty**

This is the code and documentation of the package mentioned in Sec. 3.3.7, loadable by option `\ligs`. See below for what is offered.

```

1 \NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
2 \ProvidesPackage{blogligs}[2012/11/29 v0.2
3     pervasive blog ligatures (UL)]
4 %% copyright (C) 2012 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
18 %%

```

4.1 **blog** Required

`blogdot` is an extension of `blog`, and must be loaded *later* (but what about options? [TODO](#)):

```

19 \RequirePackage{blog}

```

4.2 Task and Idea

`\UseBlogLigs` as offered by `blog.sty` does not work inside macro arguments. You can use `\ParseLigs{<text>}` at such locations to enable “ligatures” again. `blogligs.sty` saves you from this manual trick. Many macros have one “text” argument only, others additionally have “attribute” arguments. Most macros `<elt-cmd>{<text>}` of the first kind are defined to expand to `\SimpleTagSurr{<elt>}{<text>}` or to `\TagSurr{<elt>}{<attrs>}{<text>}` for some HTML element `<elt>` and some attribute assignments `<attrs>`. When a macro in addition to a “text” element has “attribute” parameters, `\TagSurr` is used as well.

```

20 % \let\blogtextcolor\textcolor
21 % \renewcommand*{\textcolor}[2]{\blogtextcolor{#1}{\ParseLigs{#2}}}

```


4.3 Quotation Marks

“Inline quote” macros `<qtd>{<text>}` to surround `<text>` by quotation marks do not follow this rule. We are just dealing with English and German double quotes that I have mostly treated by `catchdq.sty`. “`<text>`” then (eventually) expands to either `\deqtd{<text>}` or `\endqtd{<text>}`, so we redefine these:

```
22 \let\blogdedqtd\dedqtd
23 \renewcommand*{\dedqtd}[1]{\blogdedqtd{\ParseLigs{#1}}}

24 \let\blogendqtd\endqtd
25 \renewcommand*{\endqtd}[1]{\blogendqtd{\ParseLigs{#1}}}
```

4.4 HTML Elements

When the above rule holds:

```
26 \let\BlogTagSurr\TagSurr
27 \renewcommand*{\TagSurr}[3]{%
28   \BlogTagSurr{#1}{#2}{\ParseLigs{#3}}}
29 \let\BlogSimpleTagSurr\SimpleTagSurr
30 \renewcommand*{\SimpleTagSurr}[2]{%
31   \BlogSimpleTagSurr{#1}{\ParseLigs{#2}}}
```

4.5 Avoiding “Ligatures” though

`\noligs{<text>}` saves `<text>` from “ligature” replacements (except in arguments of macros inside `<text>` where `blogligs` enables ligatures):

```
32 \newcommand*{\noligs}{} \let\noligs\@firstofone %% !!!
```

I have found it useful to disable replacements within `\code{<text>}`:

```
33 \renewcommand*{\code}[1]{\STS{code}{\noligs{#1}}}
```

TODO: kind of mistake, `\STS` has not been affected anyway so far, then defining `\code` as `\STS{code}` should suffice.

`\NoBlogLigs` has been meant to disable “ligatures” altogether again. I am not sure about everything ...

```
34 \renewcommand*{\NoBlogLigs}{%
35   \def\BlogOutputJob{LEAVE}%
36   % \let\deqtd\blogdeqtd %% rm. 2012/06/03
37   \let\TagSurr\BlogTagSurr
38   \let\SimpleTagSurr\BlogSimpleTagSurr
39   \FDnormalTilde
40   \MakeActiveDef\~{\&nbsp;}% %% TODO new blog cmd
41 }
```

TODO: `\UseBlogLigs` might be redefined likewise (in fact `blogligs` activates ligatures inside text arguments unconditionally at present, I keep this for now since I have used it this way with `texblog.fdf` over months, and changing it may be dangerous where I have used tricky workarounds to overcome the `texblog.fdf` mistake). But with

```
\BlogInteceptEnvironments
```

this is not needed when you use `\NoBlogLigs` for the contents of some L^AT_EX environment.

4.6 The End and HISTORY

```
42 \endinput
```

VERSION HISTORY

```
43 v0.1    2012/01/08ff. developed in 'texblog.fdf'
44 v0.2    2012/11/29   own file
45
46
```

5 Wiki Markup by `markblog.sty`

5.1 Introduction

This is the code and documentation of the package mentioned in Sec. 3.3.7, loadable by option `[mark]`. See below for what is offered. You should also find a file `'markblog.htm'` that sketches it. Moreover, `'texlinks.pdf'` describes in detail to what extent Wikipedia's "piped links" with `'[[<wikipedia-link>]]'` is supported.

5.2 Similar Packages

`wiki.sty` from the `nicetext`⁵ bundle has offered some Wikipedia-like markup as a front-end for ordinary typesetting with L^AT_EX (for DVI/PDF), implemented in a way very different from what is going on here, rather converting markup sequences *during* typesetting.

More similar to the present approach is the way how Wikipedia section titles in package documentation is implemented by `makedoc` from the `nicetext` bundle, based on `preprocessing` by `fifinddo`.

In general, John MacFarlane's `pandoc` (cf. German Wikipedia) converts between wiki-like (simplified) markup and L^AT_EX markup. (It deals with rather fixed markup rules, while we here process markup sequences independently of an entire markup *language*.)

⁵<http://tug.ctan.org/pkg/nicetext>

Another straightforward and well-documented way to *preprocess* source files for converting simplified markup into T_EX markup is Paul Isambert’s *interpreter*. It relies on LuaT_EX where Lua does the preprocessing.

5.3 Package File Header

```

1 \NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
2 \ProvidesPackage{markblog}[2012/11/29 v0.2
3     wiki markup with blog.sty (UL)]
4 %% copyright (C) 2012 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
18 %%
```

5.4 `blog` Required

`blogdot` is an extension of `blog` and must be loaded *later* (but what about options? [TODO](#)):

```

19 \RequirePackage{blog}
```

5.5 Replacement Rules

2012/01/06f.:

```

20 \FDpseudoTilde
```

`\[[<wikipedia-link>]]`: a `fifinddo` job is defined that passes to the “ligature” job for arrows in `blog.sty`:

```

21 \MakeExpandableAllReplacer{blog[[]{[[]{\protect\catchdbrkt}{blog<-}
22 \def\catchdbrkt#1]{\Wikiref{#1}}}} %% + t 2012/01/09
```

The stars are inspired by Markdown (thanks to Uwe Ziegenhagen October 2011), while I have own ideas about them.

```

23 \MakeExpandableAllReplacer{blog**}{**}
24     {\protect\doublestar:}{blog[[}
25 \MakeExpandableAllReplacer{blog***}{***}
26     {\protect\triplestar:}{blog**}
27 % \CopyFDconditionFromTo{blog***}{BlogLIGs}
```

Apostrophes:

```

28 \MakeActiveDef\'\noexpand'
29 \MakeExpandableAllReplacer{blog\string'\string'}{' '}
30 {\protect\doubleapostr:}{blog***}
31 \MakeExpandableAllReplacer{blog\string'\string'\string'}{' ' '}
32 {\protect\tripleapostr:}{blog\string'\string'}
33 \MakeOther\'

```

Replacing three apostrophes by ‘\tripleapostr’ becomes the first job called with ‘\UseBlogLigs’:

```

34 \CopyFDconditionFromTo{blog' ' '}{BlogLIGs}

```

5.6 Connecting to L^AT_EX commands

`\MakePairLaTeXcmd#1#2` replaces ‘#1 $\langle text \rangle$ #1’ by ‘#2{ $\langle text \rangle$ }’:

```

35 \newcommand*{\MakePairLaTeXcmd}[2]{%
36   \@ifdefinable#1{\def#1:##1#1:{#2{##1}}}}
37   %% ":" for "... " 2012/01/30

```

`** $\langle text \rangle$ **` is turned into ‘\mystrong{ $\langle text \rangle$ }’, and `*** $\langle text \rangle$ ***` is turned into ‘\myalert{ $\langle text \rangle$ }’. I have used two shades of red for them:

```

38 \MakePairLaTeXcmd\doublestar\mystrong
39 \MakePairLaTeXcmd\triplestar\myalert

```

As in editing Wikipedia, `'' $\langle text \rangle$ ''` renders $\langle text \rangle$ in italics (or *slanted*), and `''' $\langle text \rangle$ '''` renders $\langle text \rangle$ bold.

```

40 \MakePairLaTeXcmd\doubleapostr\textit
41 \MakePairLaTeXcmd\tripleapostr\textbf

```

5.7 The End and HISTORY

```

42 \endinput

```

VERSION HISTORY

```

43 v0.1    2012/01/06ff. developed in 'texblog.fdf'
44 v0.2    2012/11/29   own file
45

```

6 Real Web Pages with lnavicol.sty

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

```

1  \ProvidesPackage{lnavicol}[2011/10/13
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
18 %%

```

6.1 blog.sty Required

—but what about options ([TODO](#))?

```

19  \RequirePackage{blog}

```

6.2 Switches

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

```

20  \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`:

```

21  \newif\ifdeep

```

6.3 Page Style Settings (to be set locally)

```

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

```

6.4 Possible Additions to **blog.sty**

6.4.1 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 $\langle style \rangle$:

```
27 \newenvironment{spancolscell}[2]{%
28     \starttr\startTd{\@colspan{#1} #2 %
29         \@width{100\%}}% %% TODO works?
30     }\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.

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

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

```
34 \newenvironment{pagehiddencells}
35     {\indentii\hiddencells}
36     {\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:

```
37 \newenvironment{FixedWidthCell}[2]
38     {\startTd{#2}\startTable{\@width{#1}}%
39     \starttr\startTd{}}
40     {\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:

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

6.4.2 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.

```
42 \newcommand*{\simpleinclgrf}[1]{\IncludeGrf{alt="" \@border{0}}%
43     {#1}}
```

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

```

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

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

45 \newcommand*{\includegraphic}[6]{%
46   \IncludeGrf{%
47     \@width{#1} \@height{#2} %% data; presentation:
48     \@border{#4}
49     alt="#5" \@title{#6}}%
50   {#3}}

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

51 \newcommand*{\insertgraphic}[9]{%
52   \IncludeGrf{%
53     \@width{#1} \@height{#2} %% data; presentation:
54     \@border{#4}
55     align="#5" hspace="#6" vspace="#8"
56     alt="#8" \@title{#9}}%
57   {#3}}

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

58 \newcommand*{\inclgrfref}[7]{%
59   \fileref{#4}{\includegraphic{#1}{#2}{#3}%
60     {#5}{#6}{#7}}}

```

6.4.3 HTTP/Wikipedia tooltips

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

```

61 \newcommand*{\httptipref}[2]{%
62   \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:

63 \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:

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

```

6.5 Page Structure

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

6.5.1 Page Head Row

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

```

68 \newcommand*{\PAGEHEAD}{%
69   \startTable{%
70     \@align@c\
71     \@bgcolor{\pagebgcolor}%
72     \@border{0}%%           %% TODO local
73     \pagespacing
74     \iftight \else \@width\pagewidth \fi
75   }\CLBrk
76   %% omitting <tbody>
77   \ \comment{ HEAD ROW }\CLBrk
78   \indentii\spancolscell{2}{}%
79 }
80 % \newcommand*{\headgrf} [1]{%           %% rm. 2011/10/09
81 %   \indentiii\simplecell{\simpleinclgrf{#1}}
82 % \newcommand*{\headgrfskiptitle}[3]{%
83 %   \pagehiddencells
84 %   \headgrf{#1}\CLBrk
85 %   \headskip{#2}\CLBrk
86 %   \headtitle1{#3}\CLBrk
87 %   \endpagehiddencells}

```

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

```

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

```

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

```

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

```

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

```

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

```


6.5.2 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 ...

```

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

```

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

```

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

```

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

```

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

```

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

```

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

```

6.5.3 Footer Row

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

```

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

```

← again class “paper”!?

```

117 }

```

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

```

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

```

6.6 The End and HISTORY

```

119 \endinput
120
121 HISTORY
122
123 2011/04/29   started (? \if...)
124 2011/09/01   to CTAN as 'twocolpg.sty'
125 2011/09/02   renamed
126 2011/10/09f. documentation more serious
127 2011/10/13   '....:' OK
128

```

7 Beamer Presentations with blogdot.sty

7.1 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. 7.4 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. 7.6). Its default value is `START` for the “slide” opened by the command `\titlescreenpage` that is defined in Sec. 7.5.

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. 7.7. 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. 7.8):

`\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` (fifinddo-info bundle), 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 .

Limitations: First I was happy about how it worked on my netbook, but then I realized how difficult it is to present the “slides” “online.” Screen sizes (centering) are one problem. (Without the “restart” idea, this might be much easier.) Another problem is that the “hidden links” don’t work with Internet Explorer as they work with Firefox, Google Chrome, and Opera. And finally, in internet shops some HTML entities/symbols were not supported. In any case I (again) became aware of the fact that HTML is not as “**portable**” as PDF.

Some **workarounds** are described in Sec. 7.9. `\FillBlogDotTypeArea` has two effects: (i) providing an additional link to the *next* slide for MSIE, (ii) *widening* and centering the *type area* on larger screens than the one which the presentation originally was made for. An optional argument of `\TryBlogDotCFG` is offered for a `.cfg` file overriding the original settings for the presentation. Using it, I learnt that for “portability,” some manual line breaks (`\`, `
`) should be replaced by “ties” between the words *after* the intended line break (when the line break is too ugly in a wider type area). For keeping the original type area width on wider screens (for certain “slides”, perhaps when line breaks really are wanted to be preserved), the `{textblock}` environment may be used. Better HTML and CSS expertise may eventually lead to better solutions.

The **name** ‘blogdot’ is a “pun” on the name of the `powerdot` package (which in turn refers to “PowerPoint”).

7.2 File Header

```

1  \NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
2  \ProvidesPackage{blogdot}[2012/11/21 v0.41 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 %%
```

7.3 blog Required

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

```
18 \RequirePackage{blog}
```

7.4 Size Parameters

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

```
\leftpagemargin, \rightpagemargin, \upperpagemargin,
\lowerpagemargin, \typeareawidth, \typeareaheight
```

mean. The choices are what I thought should work best on my 1024×600 screen (in fullscreen mode); but I had to optimize the left and right margins experimentally (with Mozilla Firefox 3.6.22 for Ubuntu canonical - 1.0). It seems to be best when the horizontal parameters together with what the browser adds (scroll bar, probably 32px with me) sum up to the screen width.

```
19 \newcommand*{\leftpagemargin}{176}
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>}
```

`\CommentBlogDotWholeWidth` produces no HTML code ...

```
28 \global\let\BlogDotWholeWidth\@empty
```

... unless calculated with `\SumBlogDotWidth`:

```

29 \newcommand*{\SumBlogDotWidth}{%
30     \relax{%                               %% \relax 2011/10/22 magic ...
31         \count@ \typeareawidth
32         \advance \count@ \leftmargin
33         \advance \count@ \rightmargin
34         \typeout{ * blogdot slide width = \the \count@ \space}%
35         \xdef \CommentBlogDotWholeWidth{%
36             \comment{ slide width = \the \count@ \ }}}

```

7.5 (Backbone for) Starting a “Slide”

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

```

37 \newcommand*{\startscreenpage}[2]{%% 0 2011/09/25!?:
38     \\CLBrk                               %% 2012/11/19

```

← \\ suddenly necessary, likewise in `texblog.fdf` with `\NextView` and `\nextruleview`. Due to recent firefox?

```

39     \startTable{%
40         \@cellpadding{0} \@cellspacing{0}%
41         \maybe@blogdot@borders           %% 2011/10/12
42         \maybe@blogdot@frame             %% 2011/10/14
43     }%
44     \CLBrk                               %% 2011/10/03
45     \starttr

```

First cell determines both height of upper page margin `\uppermargin` and width of left page margin `\leftmargin`:

```

46     \startTd{\@width {\leftmargin }%
47         \@height {\uppermargin}}%
48     %     \textcolor{\bodybgcolor}{XYZ}%
49     \endTd

```

Using `\typeareawidth`:

```

50     %     \startTd{\@width {\typeareawidth}}\endTd
51     \simplecell{%
52         \CLBrk
53         \hanc{#2}{\hvspace {\typeareawidth}%
54             {\uppermargin}}%
55         \CLBrk
56     }%

```

Final cell of first row determines right margin width:

```

57     \startTd{\@width {\leftmargin}}\endTd
58     \endtr
59     \starttr
60     \emptycell \startTd{\@height {\typeareaheight} #1}%
61 }

```

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

`\ancref{START}{start presentation}` :

```
62 \newcommand*\titlescreenpage{%
63   \startscreenpage{\@align@c}{START}}
```

7.6 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.

```
64 \newcommand*\screenbottom}[1]{%
65   \ifFillBlogDotTypeArea
66     <p>\ancref{#1}{\BlogDotFillText}%    %% not </p> 2011/10/22
67   \fi
68   \endTd\emptycell
69   \endtr
70   \CLBrk
71   \tablerow{bottom margin}%          %% 2011/10/13
72   \emptycell
73   \CLBrk
74   \startTd{\@align@c}%
75   \ancref{#1}{\HVspace{\BlogDotBottomFill}%
```

← seems to be useless now (2011/10/15).

```
76   {\typeareawidth}%
77   {\lowerpagemargin}}%
78   \endTd
79   \CLBrk
80   \simplecell{\ancref{\BlogDotRestart}%
81             {\hvspace{\rightpagemargin}%
82              {\lowerpagemargin}}}%
83   \endtablerow
84   \CLBrk
85   \endTable
86 }
```

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.

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

7.7 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.

```
88 \newcommand*{\nextscreenpage}[2]{%
89     \screenbottom{#2}\CLBrk
90     \hrule           \CLBrk
91     \startscreenpage{#1}{#2}}
```

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

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

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

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

7.8 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.

```
94 \newcommand*{\cheading}[1]{\CLBrk\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 `\typeareaawidth`. 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.)

```
95 \newenvironment*{textblock}[1]
96     {\startTable{\@width{#1}}\starttr\startTd{}}
97     {\endTd\endtr\endTable}
```

7.9 Debugging and .cfigs

`\ShowBlogDotBorders` shows borders of the page margins and may be undone by `\DontShowBlogDotBorders`:

```
98 \newcommand*{\ShowBlogDotBorders}{%
99     \def\maybe@blogdot@borders{rules="all"}}
100 \newcommand*{\DontShowBlogDotBorders}{%
101     \let\maybe@blogdot@borders\@empty}
102 \DontShowBlogDotBorders
```


`\ShowBlogDotFrame` shows borders of the page margins and may be undone by `\DontShowBlogDotFrame`:

```

103 \newcommand*\ShowBlogDotFrame}{%
104     \def\maybe@blogdot@frame{\@frame@box}}
105 \newcommand*\DontShowBlogDotFrame}{%
106     \let\maybe@blogdot@frame\@empty}
107 \DontShowBlogDotFrame

```

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

`\BlogDotFillText` is a dirty trick ... seems to widen the type area and this way centers the text on wider screens than the one used originally. Of course, this can corrupt intended line breaks.

```

108 \newcommand*\BlogDotFillText}{%           %% 2011/10/11
109     \center
110     \BlogDotFillTextColor{%           %% 2011/10/12
111 %           X\X           %% insufficient
112           X X X X X X X X X X X X X X X X X
113           X X X X X X X X X X X X X X X X X
114           X X X X X X X X X X
115           X X X X X X X X X X
116 %           X X X X X X X X X X X X X X X X X
117     }
118     \endcenter
119 }

```

`\FillBlogDotTypeArea` fills `\BlogDotFillText` into the type area, also as a link to the next slide. This may widen the type area so that the text is centered on wider screens than the one the HTML page was made for. The link may serve as an alternative to the bottom margin link (which sometimes fails).

`\FillBlogDotTypeArea` can be undone by `\DontFillBlogDotTypeArea`:

```

120 \newcommand*\FillBlogDotTypeArea}{%
121     \let\ifFillBlogDotTypeArea\iftrue
122     \typeout{ * blogdot filling type area *}}           %% 2011/10/13
123 \newcommand*\DontFillBlogDotTypeArea}{%
124     \let\ifFillBlogDotTypeArea\iffalse}
125 \DontFillBlogDotTypeArea

```

`\FillBlogDotBottom` fills `\BlogDotFillText` into the center bottom cell. I tried it before `\FillBlogDotTypeArea` and I am not sure ... It can be undone by `\DontFillBlogDotBottom`:

```

126 \newcommand*\FillBlogDotBottom}{%
127     \let\BlogDotBottomFill\BlogDotFillText}

```

... actually, it doesn't seem to make a difference! (2011/10/13)

```

128 \newcommand*\DontFillBlogDotBottom}{\let\BlogDotBottomFill\@empty}
129 \DontFillBlogDotBottom

```

`\DontShowBlogDotFillText` makes `\BlogDotFillText` invisible, `\ShowBlogDotFillText` makes it visible. Until 2011/10/22, `\textcolor` (blog.sty) used the `` element that is deprecated. I still use it here because it seems to suppress the `hover` CSS indication for the link. (I might offer a choice—[TODO](#))

```

130 \newcommand*{\DontShowBlogDotFillText}{%
131 % \def\BlogDotFillColor{\textcolor{\bodybgcolor}}
132 \def\BlogDotFillColor{%
133 \TagSurr{font}{color="\bodybgcolor"}}
134 \newcommand*{\ShowBlogDotFillText}{%
135 \def\BlogDotFillColor{\textcolor{red}}}
136 \DontShowBlogDotFillText

```

As of 2011/10/21, `texlinks.sty` provides `\ctanfileref{<path>}{<file-name>}` that uses an online T_EX archive according to

`\usemirrorctan` or `\usetugctan`.

This is preferable for an online version of the presentation. In `dantev45.htm`, this is used for example files. When, on the other hand, internet access during the presentation is bad, such example files may instead be loaded from the “current directory.” `\usecurrdirctan` modifies `\ctanfileref` for this purpose (i.e., it will ignore *<path>*):

```

137 \newcommand*{\usecurrdirctan}{%
138 \renewcommand*{\ctanfileref}[2]{%
139 \hnewref{}{##2}{\filenamefmt{##2}}}

```

(Using a local TDS tree would be funny, but I don’t have good idea for this right now.)

`\TryBlogDotCFG` looks for `blogdot.cfg`,

`\TryBlogDotCFG[<file-name-base>]`

looks for *<file-name-base>.cfg* (for recompiling a certain file):

```

140 \newcommand*{\TryBlogDotCFG}[1][blogdot]{%
141 \InputIfFileExists{#1.cfg}{%
142 \typeout{
143 * Using local settings from \string'#1.cfg\string' *}%
144 }{}%
145 }
146 \TryBlogDotCFG

```

7.10 The End and HISTORY

147 \endinput

VERSION HISTORY

148 v0.1 2011/09/21f. started
 149 2011/09/25 spacing/padding off
 150 2011/09/27 \CLBrk
 151 2011/09/30 \BlogDotRestart
 152 used for DANTE meeting
 153 v0.2 2011/10/03 four possibly independent page margin
 154 parameters; \hvspace moves to texblog.fdf
 155 2011/10/04 renewed \body commented out
 156 2011/10/07 documentation
 157 2011/10/08 added some labels
 158 2011/10/10 v etc. in \ProvidesPackage
 159 part of morehype RELEASE r0.5
 160 v0.3 2011/10/11 \HVspace, \BlogDotFillText
 161 2011/10/12 commands for \BlogDotFillText
 162 2011/10/13 more doc. on "debugging";
 163 \ifFillBlogDotTypeArea, \tablerow, messages
 164 2011/10/14 \maybe@blogdot@frame
 165 2011/10/15 doc. note: \HVspace useless
 166 part of morehype RELEASE r0.51
 167 v0.4 2011/10/21 \usecurrdirctan
 168 2011/10/22 FillText with <p> instead of </p>, its color
 169 uses ; some more reworking of doc.
 170 part of morehype RELEASE r0.6
 171 v0.41 2012/11/19 \startscreenpage with \\\; doc. \\\
 172 2012/11/21 updating version infos, doc. \pagebreak
 173