blog.sty

—

Generating HTML Quickly with TEX*

Uwe Lück†

January 28, 2011

Abstract

blog.sty provides TEX macros for generating web pages, based on processing text files using the fifinddo package. Some LATEX 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"* TEX's macro language for generating HTML code and entirely *ignores* TEX's typesetting capabilities.

Contents

*This document describes version v0.4 of blog.sty as of 2011/01/24.

†`http://contact-ednotes.sty.de.vu`

1 Installing and Usage

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

User commands are described near their implementation below.

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

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

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

```
\RequirePackage{blog}
```

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

The driver file can 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 [TODO: present in nicetext].

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

[1] `http://www.tex.ac.uk/cgi-bin/texfaq2html?label=inst-wlcf`

2 Example

My "TEX-generated pages"[2] use a **driver** file `makehtml.tex`. To choose a page to generate, I "uncomment" just one of several lines that set the "current conversion job" from a list. I choose the example of a simple "site map:" `texmap.htm` is generated from **source** file `texmap.tex`.

2.1 Driver File `makehtml.tex`

```
\def \GenDate {2011/01/26}
\ProvidesFile{makehtml.tex}
             [\GenDate \space- UL's TeX-generated web pages]
\RequirePackage{blog}[2011/01/24]
\input{atari.fdf} \input{lang-en.fdf}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\def \htmljob
% {texmap}                                              %% 2011/01/25
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% {aaoe1550}                                          %% 2010/09/14
% {heyctan}
% {jobs}          %%% \BlogAutoPars
% {makeshow}
% {private}     \input{lang-de.fdf}
% {README}
{texhax}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% "CFG:"
\newcommand*{\enlastrev}{%
  \rightpar{%
    \textit{Last~revised~\GenDate\
            \copyright~\FileRef{contacten.html}{%% 2010/09/26
            Uwe~L\uml{u}ck}}}}
\newcommand*{\delastrev}{%
  \rightpar{%
    \textit{Letzte \uml{A}nderung~\GenDate\
            \copyright~\FileRef{contacten.html}{%% 2010/09/26
            Uwe~L\uml{u}ck}}}}
\newcommand*{\entotopofpage}{%
    \rightpar{\ancref{top-of-page}{[\textit{\to~top~of~page}~]}}}
\newcommand*{\detotopofpage}{%
    \rightpar{\ancref{top-of-page}{[\textit{\to~Seitenanfang}~]}}}
\newcommand*{\file}{\code}
\newenvironment*{itquote}{\quote<i>}{</i>\endquote}
\newcommand*{\mystrong}{\textcolor{\#aa0000}}
```

[2] `www.webdesign-bu.de/uwe_lueck/texmap.htm`

```
\newcommand*{\myalert}{\textcolor{red}}
\newcommand*{\nextview}[1]{%                          %% corr. 2010/09/15
    \rightpar{\ancref{#1}{[\to]|%
              \ancref{top-of-page}{~\uparrow~]}}}
\newcommand*{\nextruleview}[1]{%                      %% 2010/09/15
    \nextview{#1}\hrule\hanc{#1}{}}
\newcommand*{\TOPref}[1]{\bytopicref{#1}{JF}}
\newcommand*{\htext}{.htm}
\newcommand*{\fivebreaks}{\\\\\\\\\\}
%% 2010/09/10:
\newcommand*{\Hrule}{\totopofpage\hrule}
%% 2010/09/15:
\renewcommand*{\body}{</head><body bgcolor="\bodybgcolor">}
% \newcommand*{\bodybgcolor}{\#ffffff}
%% <- 2010/09/16 ->
\newcommand*{\bodybgcolor}{\#fffe7}                   %% 2010/11/23
% \newcommand*{\bodybgcolor}{\#ffffdd}                  %% 2010/11/17
% \newcommand*{\bodybgcolor}{\#faffe7}                  %% 2010/11/17
% \newcommand*{\bodybgcolor}{\#fcffe7}                  %% 2010/11/23
% \newcommand*{\bodybgcolor}{\#fbfff0}                  %% 2010/11/23
% \newcommand*{\bodybgcolor}{\#fffffe}                  %% 2010/11/23
%% 2010/09/11:
\newcommand*{\textopofpage}{%
%   \hanc{top-of-page}{[~\code{\Fileref{texmap} > \htmljob}~]}}
  \hanc{top-of-page}{%
    \small
      \file{uwe_lueck}
      \ifx\htmljob\texmapName\else \code{>} \Fileref{texmap}\fi
      \ \code{>} \file{\htmljob}}%
    \endsmall}
\newcommand*{\texmapName}{texmap}
\newcommand*{\Fileref}[1]{\fileref{#1}{\file{#1}}}
\newcommand*{\texrobots}{\robots{index,follow,noarchive}}
\newcommand*{\texstylesheet}{\stylesheet{all}{plain}}
%% 2010/09/12:
\newcommand*{\prl}[1]{#1}               %% prg lang, corr. 2010/12/15
\newcommand*{\src}[1]{\STS{sup}{[#1]}}
%% 2010/11/13 TODO -- catcodes.sty!?:
{\catcode`\"\active \gdef\catchdq#1"{\dqtd{#1}}}
%% 2010/11/23:
\newcommand*{\idx}[1]{\textcolor{green}{\code{\lt#1\gt}}}
\renewcommand*{\,}{ }                          %% 2011/01/01
\newcommand*{\pardash}{\,\emdash\,} %% for playing     2011/01/14
\newcommand*{\UKFAQref}[1]{\ukfaqref{#1}{UK~FAQ}}   %% 2011/01/03

\ResultFile{\htmljob\htext}
```

```
\typeout{^^J\screenqtd{blog.sty} generating \screenqtd{\htmljob\htext}}
  \errorcontextlines=6
%   \tracingmacros=1
\BlogCopyFile[\AtariCodes
              \MakeActiveDef\"{\catchdq}%
             ]{\htmljob.tex}
% \CopyFile[\BlogCodes\AtariCodes]{\htmljob.tex}
\CloseResultFile
\stop
```

2.2 Source File texmap.tex

```
\comment{ 2011/01/25 \string\endash\ -> \string\pardash\ }
\comment{ 2010/12/05 \string\emdash\ -> \string\endash\  }
\head \charset{ISO-8859-1} %%% {utf-8}
  \texrobots
  \texstylesheet
  \title{TeX-generated pages - U. L.}
\body \textopofpage
\heading1{Uwe Lück's \TeX-generated/related pages}
\emdash\,I'm playing with a different style of pages here.
  \hrule\\  %%%  \endgraf
The present page leads you to:

\begin{enumerate}
  \item \FileRef{index.html}{\file{index}}\pardash my English main page
  \item \FileRef{schreibt.html}{\file{schreibt}}\pardash my German main page
\hrule
  \item \Fileref{aaoe1550}\pardash UMTS stick with Linux netbook
  \item \Fileref{heyctan}\pardash CTAN discoveries
  \item \Fileref{jobs}\pardash coaching             %% explained 2010/09/24
  \item \Fileref{makeshow}\pardash\TeX\ almost WYSIWYG \dots
  \item \Fileref{texhax}\pardash studies on texhax postings
\end{enumerate}

  \hrule
\enlastrev
\entotopofpage
\fivebreaks \fivebreaks
\finish
```

3 The Package File

3.1 Package File Header (Legalize)

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01] %% \newcommand* etc.
```

```
\ProvidesPackage{blog}[2011/01/24 v0.4 simple fast HTML (UL)]
%% copyright (C) 2010 2011 Uwe Lueck,
%% http://www.contact-ednotes.sty.de.vu
%% -- author-maintained in the sense of LPPL below.
%%
%% This file can be redistributed and/or modified under
%% the terms of the LaTeX Project Public License; either
%% version 1.3c of the License, or any later version.
%% The latest version of this license is in
%%     http://www.latex-project.org/lppl.txt
%% We did our best to help you, but there is NO WARRANTY.
%%
%% Please report bugs, problems, and suggestions via
%%
%%   http://www.contact-ednotes.sty.de.vu
%%
```

3.2 Processing

We are building on the fifinddo package:

```
\RequirePackage{fifinddo}
```

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

```
\newif\ifBlogAutoPars
\newcommand*{\BlogAutoPars}{\BlogAutoParstrue}
\newcommand*{\noBlogAutoPars}{\BlogAutoParsfalse}
\DeclareOption{autopars}{\BlogAutoPars}
\ProcessOptions
\MakeOther\< \MakeOther\>               %% TODO ...
\newcommand*{\BlogCopyFile}[2][]{%
    \ProcessFileWith[\BlogCodes #1]{#2}{%
        \IfFDinputEmpty
            {\IfFDpreviousInputEmpty
                \relax
                {\WriteResult{\ifBlogAutoPars<p>\fi}}}%
            \CopyLine
    }%
}
```

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

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

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

3.3 General HTML Matters

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

3.3.1 General Tagging

`\TagSurr{`⟨*el-name*⟩`}{`⟨*attr*⟩`}{`⟨*content*⟩`}`

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

`\SimpleTagSurr{`⟨*el-name*⟩`}{`⟨*content*⟩`}`

are used to avoid repeating element names ⟨*el-name*⟩ in definitions of TEX macros that refer to "entire" elements—as opposed to elements whose content often spans lines (as readable HTML code). We will handle the latter kind of elements using 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:

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

3.3.2 Attributes

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

`\@`⟨*attr*⟩`{`⟨*value*⟩`}` to "abbreviate" ⟨*attr*⟩`="`⟨*value*⟩`"`

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

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

So after `\declareHTMLattrib{`⟨*attr*⟩`}`, `\@`⟨*attr*⟩ is a TEX macro expecting one parameter for the specification.

A few frequent attributes are declared this way here. `@href` is most important for that "hyper-text:"

```
\declareHTMLattrib{href}
```

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

```
\declareHTMLattrib{name}                    %% 2010/11/06
% \expandafter\show\csname @href\endcsname
```

`@bgcolor` and other attributes following are especially used in formatting **tables**:

```
\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 LATEX (for specifying the width of a `\hrule` or `\vrule` from TEX; again just saving TEX tokens rather than for readibility).

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

Same with `@height`:

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

We can enumerate the specifications allowed for `@align` ... actually it isn't used/introduced here ... we just abbreviate (indeed!) entire attribute specifications:

```
\newcommand*{\@align@c}{align="center"} %% 2010/08/03
\newcommand*{\@align@l}{align="left"}   %% 2010/07/17
\newcommand*{\@align@r}{align="right"}
```

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

```
\declareHTMLattrib{cellpadding}         %% 2010/07/18
\declareHTMLattrib{cellspacing}         %% 2010/07/18
\declareHTMLattrib{colspan}             %% 2010/07/17
\declareHTMLattrib{frame}               %% 2010/07/24
```

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

3.3.3 HTML's Special Symbols

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

```
62  {\catcode`\&=12   \catcode`\#=12
63   \gdef\AmpMark{&} \gdef\#{#}}
```

... `\CompWordMark` etc.?

3.3.4 Head

`\head` produces the first two tags that an HTML file must start with (we sometimes use `^^J` for line breaks in the HTML file to get some readibility of the generated code):

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

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

```
65  \newcommand*{\MetaTag}[1]{\space\space<meta #1>}
66  \newcommand*{\ContentAtt}[1]{content="#1"}
```

`\charset{`⟨*code-page*⟩`}`

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

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

```
69  \newcommand*{\description}[1]{%
70    \MetaTag{\@name{description} \ContentAtt{#1}}}
```

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

`\robots{`⟨*instructions*⟩`}`:

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

`\stylesheet{`⟨*media*⟩`}{`⟨*css*⟩`}` !?

```
73  \newcommand*{\stylesheet}[2]{%
74    \space\space                      %% 2010/09/10
75    <link rel="stylesheet" media="#1" type="text/css" \@href{#2.css}>}
```

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

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

3.3.5 Body

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

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

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

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

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

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

3.4 Fonts

`\heading{`⟨*level*⟩`}{`⟨*text*⟩`}` prints ⟨*text*⟩ with size dependent on ⟨*level*⟩. The latter may be one out of `1`, `2`, `3`, `4` (I think).

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

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

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

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

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

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

```
82  \renewcommand*{\textbf}{\SimpleTagSurr b}
83  \renewcommand*{\texttt}{\SimpleTagSurr{tt}}     %% 2010/06/07
```

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

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

`\code{`⟨*text*⟩`}` is different than the former font commands, it is *not* a standard LaTeX macro. It is similar to `\verb`, which however doesn't work with a pair of curly braces. The macro name rather is derived from the HTML element name `code`:

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

`\emph{`⟨*text*⟩`}` is LaTeX's command again, but somewhat abused, expanding to '`<em>`⟨*text*⟩`</em>`':

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

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

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

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

3.5 Environments

We reduce LaTeX'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 TeX 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.)

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

`{center}`—TODO cf. `<center>` 2010/07/18:

```
\renewenvironment*{center}{<p align="center">}{</p>}
```

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

As formerly with "fonts," we have *two* policies for **choosing macro names**: (i) using an *existing* HTML element name, (ii) using a LaTeX command name for accessing a somewhat similar HTML element having a *different* name.

`\declareHTMLelement{`⟨*el-name*⟩`}` creates a *new* ⟨*el-name*⟩ "environment" according to policy (i):

```
\newcommand*{\declareHTMLelement}[1]{%
    \newenvironment*{#1}{<#1>}{</#1>}}
```

`\renderHTMLelement{`⟨*ltx-env*⟩`}{`⟨*el-name*⟩`}` *redefines* LaTeX's ⟨*ltx-env*⟩ environment to use HTML's ⟨*el-name*⟩ element according to policy (ii):

```
\newcommand*{\renderHTMLelement}[2]{%
    \renewenvironment*{#1}{<#2>}{</#2>}}
```

Applying former auxiliaries:

`\small` is a LaTeX command from a *class*—that we won't load, therefore we can create a *new* `{small}` environment using `<small>` according to policy (i):

```
\declareHTMLelement{small}
```

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

```
\renderHTMLelement{enumerate}{ol}
\renderHTMLelement{itemize}  {ul}
```

With blog.sty, {verbatim} really doesn't work much like its original LaTeX variant. TeX macros inside still are expanded, and you must care yourself for wanted "quoting":

```
\renderHTMLelement{verbatim} {pre}      %% 2010/09/10
```

{quote} is defined in LaTeX classes only again. To use it for policy (ii), we give it a dummy definition so \render... won't complain:

```
\let\quote\empty
\renderHTMLelement{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.

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

← 2010/05/23, 2010/06/10 →

```
% \renewcommand*{\item}{<li>}
```

LaTeX'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.

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

3.6 Links

3.6.1 Basic Link Macros

\hanc{⟨*id*⟩}{⟨*text*⟩} makes ⟨*text*⟩ an anchor with HTML label ⟨*id*⟩ (like hyperref's \hypertarget):

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

\hancref{⟨*id*⟩}{⟨*url*⟩}{⟨*text*⟩} makes ⟨*text*⟩ an anchor with HTML label ⟨*id*⟩ and at the same time a link to ⟨*url*⟩:

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

\href{⟨*id*⟩}{⟨*text*⟩} makes ⟨*text*⟩ a link to ⟨*url*⟩:

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

3.6.2 Special cases of Basic Link Macros

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

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

`\ancref{`⟨*id*⟩`}{`⟨*text*⟩`}` makes ⟨*text*⟩ a link to an anchor ⟨*id*⟩ on the same web page. This is especially useful for a "table of contents"—a list of links to sections of the page.

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

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

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

3.6.3 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.)

```
111  % \newcommand*{\emhref}[2]{\href{#1}{\emph{#2}}}
112  \newcommand*{\ithref}[2]{\href{#1}{\textit{#2}}}
113  \newcommand*{\itancref}[2]{\ancref{#1}{\textit{#2}}}%% 2010/05/30
114  \newcommand*{\emancref}[2]{\ancref{#1}{\emph{#2}}}
```

3.6.4 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 the Atari emulator (Hatari). I could switch to `.html` some time using symbolic links. 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*⟩`}`. **Only now (v0.2) I realize** it's just the same as `\href`!

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

← 2010/09/11 →

```
116  \newcommand*{\fileref}[1]{\FileRef{#1\htext}}
117  % \newcommand*{\emfileref}[2]{\fileref{#1}{\emph{#2}}}
118  \newcommand*{\itfileref}[2]{\fileref{#1}{\textit{#2}}}
```

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

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

← 2010/05/31 →

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

3.6.5 Built Macros for Links to Remote Files

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

`\httpref{`⟨*host-path[#frag]*⟩`}{`⟨*text*⟩`}`

making ⟨*text*⟩ a link to `http://`⟨*host-path[#frag]*⟩:

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

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

```
\RequirePackage{texlinks}
```

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

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

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

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

3.7 Symbols

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

```
\@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)

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

`\renderHTMLsymbol{`⟨*macro*⟩`}{`⟨*name*⟩`}` *redefines* macro ⟨*macro*⟩ to expand to `&`⟨*name*⟩`;`:

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

These auxilaries applied: `\ccedil` (TODO re-order) ...

```
\declareHTMLsymbol{ccedil}              %% 2010/08/17
```

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

```
\renderHTMLsymbol {\gets}     {larr}
\renderHTMLsymbol {\to}       {rarr}
\renderHTMLsymbol {\uparrow}  {uarr}    %% 2010/09/15
\renderHTMLsymbol {\downarrow}{darr}    %% 2010/09/15
```

> and <: `\gt`, `\lt` ...

```
\declareHTMLsymbol{gt}                  %% greater than 2010/06/13
\declareHTMLsymbol{lt}                  %% less than    2010/06/13
```

Redefinitions of `\&` and `\%` (well, `\PercentChar` is fifinddos version of LaTeX's `\@percentchar`):

```
\renderHTMLsymbol {\&}    {amp}
\let\%\PercentChar                      %% 2010/07/01
```

Horizontal ellipsis: `\dots` ...

```
\renderHTMLsymbol {\dots} {hellip}
```

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

```
\renderHTMLsymbol {~}     {nbsp}
```

But now we need a replacement `\tilde` for URLs involving home directories of institution members:

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

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* police besides (i) and (ii) earlier; cf. LaTeX's `\textemdash` and `\textendash`.—`\newcommand` does not accept macros whose names start with `end`, so: `\endash`, `\emdash` ...

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

"Math" spaces `\enspace`, `\quad`, `\qquad`:

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

Quotes: `\lq`, `\rq` ... (TODO own subsection)

```
149  \renderHTMLsymbol{\lq}    {lsquo}
150  % \newcommand*   {\rsquo} {&rsquo;} %% removed 2010/04/26
151  \renderHTMLsymbol{\rq}    {rsquo}
```

In order to use the right single quote for the HTML apostrophe, we must save other uses before. `\screentqtd{⟨text⟩}` is used for screen messages, and `\urlapostr` is the version of the right single quote for URLs (TODO which? Wikipedia?):

```
152  \newcommand*{\screenqtd}[1]{‘#1’}
153  \newcommand*{\urlapostr}  {’}       %% 2010/09/10
```

Here finally is the change of `’`:

```
154  \MakeActiveDef\’{&rsquo;}
```

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

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

```
155  \declareHTMLsymbol{ldquo}
156  \declareHTMLsymbol{rdquo}
157  \declareHTMLsymbol{sbquo}           %% 2010/07/01
158  \renewcommand*{\prime}{&prime;}
159  \declareHTMLsymbol{Prime}
160  % \newcommand*{\Prime}{&Prime;}
```

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

```
161  \def\endqtd#1{\ldquo#1\rdquo}       %% \newcommand: "\end"
162  \newcommand*\enqtd[1]{\lq#1\rq}     %% 2010/09/08, \new... 2010/11/08
163  \newcommand*{\dedqtd}[1]{&bdquo;#1\ldquo}
```

... TODO `\bdquo`!?

Some more letters: `\eacute`, `\ocirc` (“rôle”), `\Omega` ...

```
164  \declareHTMLsymbol{eacute}
165  \declareHTMLsymbol{ocirc}
166  \renderHTMLsymbol{\Omega}   {Omega}     %% 2010/08/24
```

`\uml{⟨char⟩}` yields the umlaut for ⟨*char*⟩ (useful in macro definitions):

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

`\copyright`:

```
168  \renderHTMLsymbol{\copyright}{copy}     %% 2010/08/24
```

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

```
169  \begingroup
170      \Delimiters\[\] \gdef\{[{] \gdef\}[}]
171  \endgroup
```

3.8 TEX-related

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

3.8.1 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 LATEX packages. However, sans-serif is of doubtable 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.

```
\newcommand*{\prg}[1]{} \let\prg\@firstofone
\newcommand*{\BibTeX}{\prg{BibTeX}} %% 2010/09/13
\renewcommand*{\TeX}{\prg{TeX}}
\renewcommand*{\LaTeX}{\prg{LaTeX}}
\newcommand*{\allTeX}{\prg{(La)TeX}}%% 2010/10/05
\newcommand*{\LuaTeX}{\prg{LuaTeX}}
\newcommand*{\pdfTeX}{\prg{pdfTeX}}
\newcommand*{\XeTeX}{\prg{XeTeX}}   %% 2010/10/09
\newcommand*{\TeXbook}{TeXbook}     %% 2010/09/13
```

3.8.2 Else

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

`\texcs{\`⟨*tex-cmd-name*⟩`}` or `\texcs\`⟨*tex-cmd-name*⟩ (care for spacing yourself):

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

3.9 Tables

These macros have proved somewhat bad. It may be better to re-implement tables support alltogether.

There are three levels of indenting:

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

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

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

2010/07/17:

```
\newcommand*{\startTable}[1]{<table #1>}
\def\endTable{</table>}
```

2010/07/18:

```
\newcommand*{\starttr}{<tr>}
\def\endtr{</tr>}
\newcommand*{\startTd}[1]{<td #1>}
\def\endTd{</td>}
\newcommand*{\simplecell}{\SimpleTagSurr{td}}
\newcommand*{\@frame@box}{\@frame{box}}
\newcommand*{\@frame@groups}{\@frame{groups}}
\newenvironment{allrulestable}[2]
  {\startTable{\@cellpadding{#1} \@width{#2}
               \@frame@box rules="all"}^^J%
   \indenti<tbody>}
  {\indenti</tbody>^^J\endTable}
```

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 ...

```
\newenvironment*{TableRow}[2]{%% weniger ^^J 2010/05/18
%   \indenti\ \comment{ #1 }^^J%
```

← 2010/05/23 →

```
%   \ \comment{ #1 }^^J%
```

← 2010/07/18 →

```
    \indenti \comment{ #1 }^^J%
    \indentii<tr #2>%
    }{%
    \indentii</tr>}
\newenvironment{tablecoloredrow}[2]
    {\TableRow{#1}{\@bgcolor{#2}}}
    {\endTableRow}
```

"top" 2010/05/18:

```
\newenvironment{tablerow}[1]{\TableRow{#1}{valign="top"}}
                            {\endTableRow}
% \newcommand*{\TableCell}[2]{\indentiii<td #1>#2</td>}
% \newcommand*{\TableCell}[2]{\indentiii\TagSurr{td}{#1}{#2}}
  %% <- 2010/07/18 ->
\newcommand*{\TableCell}[2]{\indentiii\startTd{#1}#2\endTd}
```

2010/06/15:

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

Idea: use closing star for environment variants!?

```
% %% 2010/05/19:
\newenvironment{bigtablecell}[1]{\BigTableCell{#1}{}}
                                {\endBigTableCell}
%                {\ifx\\#1\\%             %% 2010/05/30
%                   \indentii\ \comment{#1}^^J%
%                 \fi
%                 \indentiii<td>}
%                {\indentii</td>}         %% !? 2010/05/23
```

Generalization 2010/06/28:

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

2010/06/05:

```
\newenvironment{BigTableCell}[2]
    {\ifx\\#1\\\indentii\ \comment{#1}^^J\fi
     \indentiii\startTd{#2}}
    {\indentii\endTd}           %% TODO indent? 2010/07/18
```

3.10 Misc

`\comment{`⟨*comment*⟩`}` produces a one-line HTML comment. By contrast, there is an environment `{commentlines}{`⟨*comment*⟩`}` for mult-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.

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

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

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

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

```
\let\ \space
```

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

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

← difficult with \stop; 2010/09/10: \endgraf produces !?

```
\renewcommand*{\endgraf}{</p>}
```

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

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

\rightpar{⟨*text*⟩} places ⟨*text*⟩ flush right. I have used this for 'Last revised ...' and for placing navigation marks.

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

For references, there were

```
% \catcode`\^=\active
% \def^#1{\SimpleTagSurr{sup}{#1}}
```

and

```
% \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.

3.11 The End

```
\endinput
```

3.12 VERSION HISTORY

```
v0.1    2010/08/20  final version for DFG
v0.2    2010/11/08  final documentation version before
                    moving some functionality to 'fifinddo'
v0.3    2010/11/10  removed ^^J from \head
        2010/11/11  moving stuff to fifinddo.sty; \BlogCopyFile
        2010/11/12  date updated; broke too long code lines etc.;
                    \CatCode replaced (implemented in niceverb only);
                    \ifBlogAutoPars etc.
        2010/11/13  doc: \uml useful in ...; \texcs
        2010/11/14  doc: argument for {commentlines},
                         referring to environments with curly braces,
                         more on \ditem
        2010/11/15  TODO: usage, templates
        2010/11/16  note on {verbatim}
```

```
267           2010/11/23  doc. corr. on \CtanPkgRef
268           2010/11/27  "keyword"; \CopyLine without ‘fd’
269           2010/12/03  \emhttpref -> \ithttpref
270           2010/12/23  ‘%’ added to \texhaxpref
271           2011/01/23  more in \Provides...
272           2011/01/24  updated copyright; resolving ‘td’ ("today")
273           JUST STORED as final version before texlinks.sty
274   v0.4    2011/01/24  moving links to texlinks.sty
275
```