The democodetools and democodelisting Packages - Version 1.0.1beta

Alceu Frigeri

September 2022

Abstract

This is ‘yet another doc/docx/doc3’ package. It is designed to be ‘as class independent as possible’, meaning: it makes no assumption about page layout (besides ‘having a marginpar’) or underline macros. Furthermore, it’s assumed that \maketitle and the abstract environment were modified by the underline class, so alternatives (based on the article class) are provided. The main idea is to be able to document a package/class loading it first and then this, so that it is possible not only to document the ‘syntax’ but also to show the end result ‘as is’ when using that other specific class/package.

Contents

1 Introduction ......................................... 1
1.1 Current Version ...................................... 2
2 democodelisting Package .............................. 2
  2.1 In Memory Code Storage .............................. 2
  2.2 Code Display/Demo .................................. 2
  2.2.1 ⟨dclisting-keys⟩ ...................................... 3
3 democodetools Package ............................... 4
  3.1 Environments ........................................ 4
  3.2 Describe Commands .................................. 4
  3.3 Macros Typeset ...................................... 5
  3.4 Args Typeset ...................................... 5
  3.5 Keys Typeset ...................................... 5
  3.6 Others ...................................... 6

1 Introduction

The packages/classes doc/docx/doc3 (and for that matter doctools) where designed to be used with dtx files, which is handy for package developers, as long as one is fine with the ‘default article’ format (which is true most of the time). This came to be from the willingness of having the ‘new look and feel’ used in doc3, but, instead of having to rely on a standard class, being able to use any class as the base one, which allows to ‘demonstrate the documented commands with the final layout’.
\texttt{democodelisting} defines a few macros to display and demonstrate \LaTeX\ code verbatim (using \texttt{listings} and \texttt{scontents}), whilst \texttt{democodetools} defines a series of macros to display/enumerate macros and environments (somewhat resembling the \texttt{doc3} style).

1.1 Current Version

This doc regards to \texttt{democodelisting} version 1.0.1beta and \texttt{democodetools} version 1.0.1beta. Those two packages are 'usable' but they haven’t been thoroughly tested, nor should anyone consider them stable (they might be considered more or less stable but more due the 'maintainer' lack of time than anything else. Use it at your own risk.)

2 democodelisting Package

It requires two packages: \texttt{listings} and \texttt{scontents}

Defines an environment: \texttt{stcode} and

4 commands: \texttt{\DemoCode}, \texttt{\DisplayCode}, \texttt{\TabbedDisplayCode} and \texttt{\setdclisting}.

2.1 In Memory Code Storage

Thanks to \texttt{scontents} (\texttt{expl3} based) it’s possible to store \LaTeX\ code snippets in a \texttt{expl3} key.

\begin{stcode}[\langle\text{keys}\rangle]\end{stcode}

\texttt{stcode} This environment is an alias to \texttt{scontents} environment (from \texttt{scontents} package), all \texttt{scontents} keys are valid, with an additional one: \texttt{st} which is an alias to the \texttt{store-env} key. The environment body is stored verbatim in the \texttt{st} named key.

2.2 Code Display/Demo

\DisplayCode just typesets \texttt{\langle\text{name}\rangle} (the key-name created with \texttt{stcode}), in verbatim mode with syntax highlight.

\DemoCode first typesets \texttt{\langle\text{name}\rangle}, as above, then it \texttt{executes} said code. Finally \texttt{\TabbedDemoCode} does the same, but typesetting it, and executed code, side by side. N.B. Both typeset and executed code are placed inside a \texttt{minipage} so that, when \texttt{executing} the code, one can have, for instance, 'normal' paragraph indentation.

For Example:

\LaTeX\ Code:

\begin{stcode}[st=stmeta]\end{stcode}

This will just typeset \texttt{\Key\{stmeta\}}:

\DisplayCode{stmeta}

and this will demonstrate it, side by side with source code:

\TabbedDemoCode[\texttt{numbers=left,\texttt{codeprefix=\{inner code\},\texttt{resultprefix=\{inner result\}\}}}]{stmeta}
This will just typeset *stmeta*:

\begin{verbatim}
Some \LaTeX-\textit{coding, for example:} \ldots
\end{verbatim}

and this will demonstrate it, side by side with source code:

\begin{tabular}{ll}
inner code & inner result \\
\begin{verbatim}
Some \LaTeX-\textit{coding, for example:} \ldots
\end{verbatim} & Some \TeX\ coding, for example: \ldots
\end{tabular}

\setdclisting{(dclisting-keys)}

Instead of setting/defining \texttt{(dclisting-keys)} per \texttt{\Demo/\Display} call, one can set those \textit{globally}, better said, \textit{in the called context group}.

N.B.: All \texttt{\Display/\Demo} commands create a local group (\texttt{\begingroup}) in which the \texttt{(dclisting-keys)} are defined, and discarded once said local group is closed. \texttt{\setdclisting} defines those keys in the \textit{current} context/group (\texttt{\def, \edef})

\subsection{(dclisting-keys)}

Using a \texttt{key = value} syntax, one can fine tune listings syntax highlight.

\texttt{settexcs, settexcs2, settexcs3}
\texttt{texcs, texcs2, texcs3}
\texttt{texcsstyle, texcs2style, texcs3style}

Those define sets of \TeX\ commands (csnames), the \texttt{set} variants initialize/re-define those sets (an empty list, clears the set), while the others extend those sets. The \texttt{style} ones redefines the command display style (an empty \texttt{par} resets the style to it’s default).

\texttt{setkeywd, setkeywd2, setkeywd3}
\texttt{keywd, keywd2, keywd3}
\texttt{keywdstyle, keywd2style, keywd3style}

Same for other \texttt{keywords} sets.

\texttt{setemph, setemph2, setemph3}
\texttt{emph, emph2, emph3}
\texttt{emphstyle, emph2style, emph3style}

for some extra emphasis sets.

\texttt{numbers, numberstyle}

\texttt{numbers} possible values are \texttt{none} (default) and \texttt{left} (to add tinny numbers to the left of the listing). With \texttt{numberstyle} one can redefine the numbering style.

\texttt{stringstyle, commentstyle}

to redefine \texttt{strings} and \texttt{comments} formatting style.

\texttt{bckgndcolor}

to change the listing background’s color.
codeprefix, resultprefix
those set the codeprefix (default: \LaTeX Code:) and resultprefix (default: \LaTeX Result:)

3 democodetools Package

3.1 Environments

\begin{Macros} {⟨macrolist⟩}
\begin{Envs} {⟨envlist⟩}
Those are the two main environments to describe Macros and Environments. Both typeset ⟨macrolist⟩ (csv list) or ⟨envlist⟩ (csv list) in the margin. N.B. Each element of the list gets \detokenize

\begin{Syntax}
The Syntax environment sets the fontsize and activates \obeylines, so one can list macros/cmds/keys use, one per line.
\LaTeX Code:
\begin{Envs}
\begin{Syntax}
\Macro\begin{Macros}{}{macrolist}
\Macro\begin{Envs}{}{envlist}
\end{Syntax}
\end{Envs}
\end{Syntax}
Those are the two main ...
\end{Envs}

\begin{Args}
\begin{Keys}
\begin{Keys+}
\begin{Values}
\begin{Values+}
\begin{Options}
\begin{Options+}
Those environments are all the same, starting a dedicated description list. Together with the many Description... commands, one can list all Options, Args, Keys, Values as needed. The + form are meant to be used with the Description...+ forms, for in text lists. The non + form are meant to have the many ‘descriptors’ in the margin par.

3.2 Describe Commands

\DescribeMacro*!+ {⟨csname⟩} [⟨oarglist⟩] [⟨marglist⟩]
* typesets the macro name in bold face.
! ⟨marglist⟩ is treated as an expandable code, ’as is’.
+ the macro name is typeseted in text.
⟨csname⟩ macro name (\detokenize)
⟨oarglist⟩ csv list of optional args.
⟨marglist⟩ csv list of mandatory args.

\DescribeArg*+ [⟨type⟩] {⟨arg⟩}
\DescribeKey*+ [⟨type⟩] {⟨arg⟩}
\DescribeValue*+ [⟨type⟩] {⟨arg⟩}
\DescribeOption*+ [⟨type⟩] {⟨arg⟩}
3.3 Macros Typeset

\Macro
\Macro\{\csname\}\{\embl\}\{\olist\}\{\mlist\}
\Macro!\{\csname\}\{\embl\}\{\par.desc.\}

When describing a macro \csname (Command Sequence, csname) the \olist and \mlist are comma separated lists (csv) of optional and mandatory arguments. \embl are optional, single char, ‘embellishment’ tokens, like \*, \+. Finally, in the ! form, the \par.desc. is any text representing the macro parameter list (for non regular, non usual, cases).

\LaTeX\ Code: \LaTeX\ Result:
\Macro\{\Macro\}\{\opt1,\opt2\}\{\arg3\}
\Macro!\{\Macro\}<!>\{\larg\embl\lpar\par.desc.\}\}

3.4 Args Typeset

\oarg \marg \parg \xarg \Arg \Meta

\oarg \marg \parg \xarg \Arg \Meta \{\arg\}

Those are meant to typeset the diverse kinds of ‘command’s arguments’ (mandatory, optional, parenthesis . . .). \Meta{\arg} typesets \arg as \{\arg\}.

\LaTeX\ Code: \LaTeX\ Result:
\oarg{\fam}\parg{xtra}\marg{textbf}{text}\xarg{x-text}

3.5 Keys Typeset

\Key \Keylist \KeyUse

\Key \Keylist \KeyUse\{\key\}

\Key \Keylist \KeyUse\{\default\}\{\keylist\}

To typeset a \key or \keylist (csv list). \pre is just prepended to \key whilst \default is the default key value. \KeyUse is just a short-cut for a, possible, common construction.
\begin{LTXCode}
\begin{tabular}{|l|l|}
\hline
\LaTeX{} Code & \LaTeX{} Result \\
\hline
\Key{Akey} & Akey \\
\KeyList[Bkey]{Akey, Bkey} & Akey, Bkey \\
\KeyUse{keyA}{arg} & keyA = ⟨arg⟩ \\
\hline
\end{tabular}
\end{LTXCode}

\begin{LTXCode}
\begin{tabular}{|l|}
\hline
\Env \Pack \Value \\hline
\Env \pack {⟨pre⟩} {⟨key⟩} \\
\Pack {⟨pre⟩} {⟨key⟩} \\
\Value {⟨pre⟩} {⟨key⟩} \\
\hline
\end{tabular}
\end{LTXCode}

Similar to \Key{} above, they will typeset a \langle Key \rangle. \langle pre \rangle is just prepended to \langle key \rangle whilst \langle default \rangle is the default key value.

### 3.6 Others

\begin{LTXCode}
\MetaFmt{} \MetaFmt{}* \langle{type}\rangle \\
\MetaFmt{}* \langle{type}\rangle \text{ It sets the font size, series, face as defined by \langle{type}\rangle, \langle{type}\rangle being one of Oarg, Marg, Parg, Xarg, Macro, Code, Key, KeyVal, Option, Value, Default. The star version uses bold.}
\end{LTXCode}

\MarginNote{} \MarginNote{} \langle{text}\rangle \\
\MarginNote{} \langle{text}\rangle \text{ As the name implies, to add small margin notes.}

\begin{LTXCode}
\dcAuthor{} \langle{name}\rangle \\
\dcDate{} \langle{date}\rangle \\
\dcTitle{} \langle{title}\rangle \\
\end{LTXCode}

\dcMakeTitle{} \text{ Those allow one to define (as in standard article, book, report classes) the document author, date and date \dcMakeTitle{} will write a typical title+author heading (as in the article class).}

\begin{LTXCode}
\begin{dcAbstract} \end{dcAbstract} \\
\begin{dcAbstract} \end{dcAbstract} \text{ Same as above, for the abstract.}
\end{LTXCode}