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		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'.

democodelisting defines a few macros to display and demonstrate LATEX code verbatim (using listings and scontents), whilst democodetools defines a series of macros to display/enumerate macros and environments (somewhat resembling the doc3 style).

1.1 Current Version

This doc regards to democodelisting version 1.0.1beta and 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: listings and scontents

Defines an environment: stcode and

4 commands: \DemoCode, \DisplayCode, \TabbedDisplayCode and \setdclisting.

2.1 In Memory Code Storage

Thanks to scontents (expl3 based) it's possible to store LATEX code snippets in a expl3 key.

stcode

```
\boldsymbol{\beta} = \boldsymbol{\beta}
```

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

2.2 Code Display/Demo

\DisplayCode \DemoCode \TabbedDemoCode

\DisplayCode just typesets \(\st-name \) (the key-name created with \(stcode \)), in verbatim mode with syntax highlight.

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

For Example:

LATEX Code:

```
\begin{stcode}[st=stmeta]
    Some \LaTeX~coding, for example: \ldots.
\end{stcode}
This will just typesets \Key{stmeta}:
    \DisplayCode{stmeta}
and this will demonstrate it, side by side with source code:
    \TabbedDemoCode[numbers=left,codeprefix={inner code},resultprefix={inner result}]{stmeta}
```

LATEX Result:

This will just typesets stmeta:

LATEX Code:

```
Some \LaTeX~coding, for example: \ldots.
```

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

inner code inner result

Some \LaTeX~coding, for example: Some LATEX coding, for example:

\setdclisting

$\stack { \langle dclisting-keys \rangle }$

Instead of setting/defining $\langle dclisting-keys \rangle$ per $\backslash Demo/\backslash Display$ call, one can set those *globally*, better said, *in the called context group*.

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

2.2.1 \(\dclisting\)-keys\(\rangle\)

Using a key = value syntax, one can fine tune listings syntax highlight.

⟨dclisting-keys⟩

settexcs, settexcs2, settexcs3

texcs, texcs2, texcs3

texcsstyle, texcs2style, texcs3style

Those define sets of LaTeX commands (csnames), the set variants initialize/redefine those sets (an empty list, clears the set), while the others extend those sets. The style ones redefines the command display style (an empty \langle par\rangle resets the style to it's default).

setkeywd, setkeywd2, setkeywd3 keywd, keywd2, keywd3 keywdstyle, keywd2style, keywd3style Same for other keywords sets.

setemph, setemph2, setemph3
emph, emph2, emph3
emphstyle, emph2style, emph3style
for some extra emphasis sets.

numbers, numberstyle

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

stringstyle, commentstyle

to redefine strings and comments formatting style.

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

Macros Envs

```
\begin{Macros} {\langle macrolist \rangle} \\ begin{Envs} {\langle envlist \rangle} \\ \end{tabular}
```

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

Syntax

\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}{Macros,Envs}
\begin{Syntax}%
\Macro{\begin{Macros}}{macrolist}
\Macro{\begin{Envs}}{envlist}
\end{Syntax}
Those are the two main ...
\end{Envs}
```

Args Keys Values Options

```
\begin{Args+}
\begin{Keys}
\begin{Keys+}
\begin{Values}
\begin{Values+}
\begin{Options}
\begin{Options+}
```

\begin{Args}

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

3.2 Describe Commands

 $\label{lem:decomp} $$\operatorname{DescribeOption*+ [\langle type \rangle] \{\langle arg \rangle\}}$$

\DescribeMacro

\DescribeArg \DescribeKey

\DescribeValue

\DescribeOption

\DescribePackage

```
\DescribeMacro*!+{\langle} [\langle arglist \rangle] {\langle}

* typesets the macro name in bold face.
! \langle marglist \rangle is treated as an expandable code, 'as is'.

+ the macro name is typeseted in text.

\langle csname \rangle macro name (\detokenize)

\langle arglist \rangle csv list of optional args.

\langle marglist \rangle csv list of mandatory args.

\DescribeArg*+ [\langle type \rangle] {\langle} \rangle
\DescribeKey*+ [\langle type \rangle] {\langle} \rangle
\DescribeValue*+ [\langle type \rangle] {\langle} \rangle
\DescribeValue*+ [\langle type \rangle] {\langle} \rangle
\DescribeValue*+ [\langle type \rangle] {\langle} \rangle
\rangle
\text{Type} \rangle \rangle \langle \rangle \rang
```

```
\DescribePackage*+ [\langle type\rangle] {\langle arg\rangle}

* typesets it in bold face.

+ typesets in text (not in marginpar)

\langle type\rangle key/arg/... format

\langle arg\rangle key/arg/... name.
```

3.3 Macros Typeset

\Macro

```
\label{eq:macro} $$\operatorname{{csname}} < \operatorname{{embl}} = [\operatorname{{olist}}] {\operatorname{{mlist}}} \\ \operatorname{{acro}} {\operatorname{{csname}}} < \operatorname{{embl}} > {\operatorname{{card}}} \\
```

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

LaTeX Code:

LATEX Result:

3.4 Args Typeset

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

defaults to Meta (it's the csname of any valid formatting command, like Meta,
textbf, etc.)

 $\langle arg \rangle$ the argument name itself.

LATEX Code:

LATEX Result:

3.5 Keys Typeset

```
\label{eq:Keylst} $$ \Keylst \ \G(\ensuremath{\mbox{keylst}} \Keylse \KeyUse {\key}\value $$
```

To typeset a $\langle \texttt{Key} \rangle$ or $\langle \texttt{keylst} \rangle$ (csv list). $\langle \texttt{pre} \rangle$ is just prepended to $\langle \texttt{key} \rangle$ whilst $\langle \texttt{default} \rangle$ is the default key value. $\langle \texttt{KeyUse} \rangle$ is just a short-cut for a, possible, common construction.

	ĿΤ	FΧ	Code:
--	----	----	-------

LATEX Result:

\Key{Akey}	Akey	
\Keylst[Bkey]{Akey,Bkey}	Akey, Bkey $ ext{keyA} = \langle ext{arg} angle$	Default: Bkey
\KeyUse{keyA}{arg}	$\text{Reyn} = \langle \text{arg} \rangle$	

\Env \Pack \Value

```
\label{eq:local_energy} $$ \Env [\langle pre \rangle] {\langle key \rangle} $$ \Pack [\langle pre \rangle] {\langle key \rangle} $$ \Value [\langle pre \rangle] {\langle key \rangle} $$
```

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

3.6 Others

\MetaFmt

\MetaFmt* [\langle type \rangle]

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

\MarginNote

$MarginNote {\langle text \rangle}$

As the name implies, to add small margin notes.

\dcAuthor \dcDate \dcTitle \dcMakeTitle $\dcAuthor \{\langle name \rangle\} \\ \dcDate \{\langle date \rangle\}$

 $\dcTitle {\langle title \rangle}$

\dcMakeTitle

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).

dcAbstract

\begin{dcAbstract} \end{dcAbstract}

Same as above, for the abstract.