The codedescribe and codelisting Packages
Version 1.1
Alceu Frigeri
May 2023

Abstract
This documentation package is designed to be ‘as class independent as possible’, depending only on expl3, scontents, listing and pifont. Unlike other packages of the kind, a minimal set of macros/commands/environments is defined: most/all defined commands have an 'object type' as a keyval parameter, allowing for an easy expansion mechanism (instead of the usual 'one set of macros/environments' for each object type).

No assumption about page layout is made (besides 'having a marginpar'), or underlying macros, so that it can be used with any document class.

Contents
1 Introduction 1
   1.1 Single versus Multi-column Classes ........................................... 2
   1.2 Current Version ................................................................. 2
2 codelisting Package 2
   2.1 In Memory Code Storage ...................................................... 2
   2.2 Code Display/Demo .............................................................. 2
       2.2.1 Code Keys ................................................................. 3
3 codedescribe Package 4
   3.1 Package Options ............................................................... 4
   3.2 Object Type keys ............................................................... 4
       3.2.1 Format Keys ............................................................... 4
       3.2.2 Format Groups ......................................................... 5
       3.2.3 Object Types ............................................................ 5
       3.2.4 Customization ........................................................... 5
   3.3 Environments ................................................................. 6
   3.4 Commands ................................................................. 7
   3.5 Auxiliary Command / Environment ......................................... 8

1 Introduction
This package aims to document both Document level (i.e. final user) commands, as well as Package/Class level commands. It’s fully implemented using expl3 syntax and structures, in special l3coffins, l3seq and l3keys. Besides those scontents and listing packages are used to typeset code snippets. The package pifont is needed just to typeset those (open)stars, in case one wants to mark a command as (restricted) expandable.

No other package/class is needed, any class can be used as the base one, which allows to demonstrate the documented commands with any desired layout.

codelisting defines a few macros to display and demonstrate \TeX{} code (using listings and scontents), whilst codedescribe defines a series of macros to display/enumerate macros and environments (somewhat resembling the doc3 style).

*https://github.com/alceu-frigeri/codedescribe
1.1 Single versus Multi-column Classes

This package 'can' be used with multi-column classes, given that the \linewidth and \columnsep are defined appropriately. \linewidth shall defaults to text/column real width, whilst \columnsep, if needed (2 or more columns) shall be greater than \marginparwidth plus \marginparsep.

1.2 Current Version

This doc regards to codelistings version 1.1 and codelisting version 1.1. Those two packages are fairly stable, and given the ⟨obj-type⟩ mechanism (see below, 3.2) it can be easily extended without changing its interface.

2 codelisting Package

It requires two packages: listings and scontents, defines an environment: codestore and 2 main commands: \tscode and \tsdemo and 1 auxiliary command: \setcodekeys.

2.1 In Memory Code Storage

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

\begin{codestore} \[(stcontents-keys)\]
\end{codestore}

This environment is an alias to scontents environment (from scontents package), all scontents keys are valid, with two additional ones: \texttt{st} and \texttt{store-at} which are aliases to the \texttt{store-env} key. If an 'isolated' \texttt{(st-name)} is given (unknown \texttt{key}), it is assumed ‘by Default’ that the environment body shall be stored in it (for use with \tscode and \tsdemo).

2.2 Code Display/Demo

\setcodekeys \{(code-keys)\}

One has the option to set ⟨code-keys⟩ (see 2.2.1) per \tscode/\tsdemo call, or globally, better said, in the called context group.

\texttt{N.B.}: All \tscode and \tsdemo commands create a local group in which the ⟨code-keys⟩ are defined, and discarded once said local group is closed. \setcodekeys defines those keys in the current context/group.

\tscode* \{(code-keys)\} \{(st-name)\}
\tsdemo* \{(code-keys)\} \{(st-name)\}

\texttt{tscode} just typesets \texttt{(st-name)} (the key-name created with stcode), in verbatim mode with syntax highlight. The non-star version centers it and use just half of the base line. The star version uses the full text width.
\texttt{tsdemo} first typesets \texttt{(st-name)}, as above, then it \texttt{executes} said code. The non-start versions place them side-by-side, whilst the star versions places one following the other.
For Example:

\LaTeX Code:
\begin{codestore}[stmeta]
  Some \LaTeX\ coding, for example: \ldots.
\end{codestore}

This will just typeset \tsobj[key]{stmeta}:
\tdemo[numbers=left,ruleht=0.5, \\
  codeprefix=(inner sample code), \\
  resultprefix=(inner sample result)] {stmeta}

\LaTeX Result:

This will just typeset \texttt{stmeta}:

Sample Code:

<table>
<thead>
<tr>
<th>inner sample code</th>
<th>inner sample result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Some \LaTeX\ coding, for example: \ldots.</td>
<td>Some \LaTeX\ coding, for example: \ldots.</td>
</tr>
</tbody>
</table>

\subsection{2.2.1 Code Keys}

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

| settexcs | settexcs, settexcs2 and settexcs3 |
| texcs | texcs, texcs2 and texcs3 |
| texcsstyle | texcsstyle, texcs2style and texcs3style |

Those define sets of \LaTeX\ commands (csnames), the \texttt{set} variants initialize/redefine those sets (an empty list, clears the set), while the others extend those sets. The \texttt{style} ones redefine the command display style (an empty \texttt{⟨value⟩} resets the style to it’s default).

| setkeywd | setkeywd, setkeywd2 and setkeywd3 |
| keywd | keywd, keywd2 and keywd3 |
| keywdstyle | keywdstyle, keywd2style and keywd3style |

Same for other \texttt{keywords} sets.

| setemph | setemph, setemph2 and setemph3 |
| emph | emph, emph2 and emph3 |
| emphstyle | emphstyle, emph2style and emph3style |

for some extra emphasis sets.

| numbers | numbers and numberstyle |
| numberstyle | 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. |

| stringstyle | stringstyle and commentstyle |
| codestyle | to redefine \texttt{strings} and \texttt{comments} formatting style. |
3 codedescribe Package

This package aims at minimizing the number of commands, having the object kind (if a macro, or a function, or environment, or variable, or key ...) as a parameter, allowing for a simple 'extension mechanism': other object types can be easily introduced without having to change, or add commands.

3.1 Package Options

It has a single package option:

\texttt{nolisting}

it will suppress the \texttt{codelisting} package load. In case it’s not necessary or one wants to use a different package for \LaTeX{} code listing.

3.2 Object Type keys

The applied text format is defined in terms of \texttt{⟨obj-types⟩}, which are defined in terms of \texttt{⟨format-groups⟩} and each one defines a 'formatting function', 'font shape', bracketing, etc. to be applied.

3.2.1 Format Keys

There is a set of primitive \texttt{⟨format-keys⟩} used to define \texttt{⟨format-groups⟩} and \texttt{⟨obj-types⟩}, which are:

- \texttt{meta} to typeset between angles,
- \texttt{xmeta} to typeset *verbatim* between angles,
- \texttt{verb} to typeset *verbatim*,
- \texttt{xverb} to typeset *verbatim*, suppressing all spaces,
- \texttt{code} to typeset *verbatim*, suppressing all spaces and replacing a TF by \texttt{TF},
- \texttt{nofmt} in case of a redefinition, to remove the 'base' formatting,
- \texttt{slshape} to use a slanted font shape,
- \texttt{slshape} to use an italic font shape,
- \texttt{slshape} in case of a redefinition, to remove the 'base' shape,
- \texttt{lbracket} defines the left bracket (when using \texttt{\textbackslash tsargs}). \textbf{Note:} this key must have an associated value,
- \texttt{rbracket} defines the right bracket (when using \texttt{\textbackslash tsargs}). \textbf{Note:} this key must have an associated value,
- \texttt{color} defines the text color. \textbf{Note:} this key must have an associated value (a color, as understood by \texttt{xcolor} package).
3.2.2 Format Groups

Using \defgroupfmt one can (re-)define custom \langle format-groups \rangle. There is, though, a set of pre-defined ones as follow:

- \texttt{meta} which sets \texttt{meta} and \texttt{color}
- \texttt{verb} which sets \texttt{color}
- \texttt{oarg} which sets \texttt{meta} and \texttt{color}
- \texttt{code} which sets \texttt{code} and \texttt{color}
- \texttt{syntax} which sets \texttt{color}
- \texttt{keyval} which sets \texttt{slshape} and \texttt{color}
- \texttt{option} which sets \texttt{color}
- \texttt{defaultval} which sets \texttt{color}
- \texttt{env} which sets \texttt{slshape} and \texttt{color}
- \texttt{pkg} which sets \texttt{slshape} and \texttt{color}

\textbf{Note:} \texttt{color} was used in the list above just as a ‘reminder’ that a color is defined/associated with the given group.

3.2.3 Object Types

Using \defobjectfmt one can (re-)define custom \langle obj-types \rangle. Similarly, there is a set of predefined ones, as follow:

- \texttt{arg, meta} based on (group) \texttt{meta}
- \texttt{verb, xverb} based on (group) \texttt{verb plus verb or xverb}
- \texttt{marg} based on (group) \texttt{meta plus brackets}
- \texttt{oarg, parg, xarg} based on (group) \texttt{oarg plus brackets}
- \texttt{code, macro, function} based on (group) \texttt{code}
- \texttt{syntax} based on (group) \texttt{syntax}
- \texttt{keyval, key, keys, values} based on (group) \texttt{keyval}
- \texttt{option} based on (group) \texttt{option}
- \texttt{defaultval} based on (group) \texttt{defaultval}
- \texttt{env} based on (group) \texttt{env}
- \texttt{pkg, pack} based on (group) \texttt{pkg}

3.2.4 Customization

One can add user defined groups(objects) or change the pre-defined ones with the following commands:

\begin{verbatim}
\defgroupfmt \{\langle format-group \rangle\}\{\langle format-keys \rangle\}
\defgroupfmt \new: 2023/05/16 \langle format-group \rangle is the name of the new group (or one being redefined, which can be one of the standard ones). \langle format-keys \rangle is any combination of the keys defined in 3.2.1

For example, one can redefine the \texttt{code} group standard color with \defgroupfmt{\texttt{code}}{\texttt{color=red}} and all obj-types based on it will be typeset in red (in the standard case: \texttt{code}, \texttt{macro} and \texttt{function} objects).
\end{verbatim}

\begin{verbatim}
\defobjectfmt \{\langle obj-type \rangle\}\{\langle format-group \rangle\}\{\langle format-keys \rangle\}
\defobjectfmt \new: 2023/05/16 \langle obj-type \rangle \texttt{is the name of the new object (being defined (or redefined), \langle format-group \rangle is the base group to be used. \langle format-keys \rangle allows for further differentiation.}

For instance, the many optional \langle *arg \rangle are defined as follow:
\end{verbatim}
3.3 Environments

\begin{codedescribe} ([obj-type]) \{[csv-list]\}
\ldots
\end{codedescribe}

[Note 1:] This is the main environment to describe \textit{Macros}, \textit{Functions}, \textit{Variable}, \textit{Environments} and \textit{etc}. \{csv-list\} is typeset in the margin. The optional \{obj-type\} defines the object-type format.

\textbf{Note:} One can change the rule color with the key \texttt{rulecolor}, for instance \begin{codedescribe}[rulecolor=white]\end{codedescribe} will remove the rules.

\textbf{Note 2:} Besides that, one can use the keys \texttt{new}, \texttt{update} and \texttt{note} to further customize it as: \begin{codedescribe}[new=2023/05/01,update=2023/05/1,note={this is an example}]\end{codedescribe}

\textbf{Note 3:} Finally, one can use \texttt{EXP} and \texttt{rEXP} to add a star ★ or a hollow star✩, as per expl3/doc3 conventions (if expandable, restricted expandable or not).

\begin{codesyntax}
\ldots
\end{codesyntax}

The \texttt{codesyntax} environment sets the fontsize and activates \texttt{obeyslines, obeyspaces}, so one can list macros/cmds/keys use, one per line.

\textbf{Note:} \texttt{codesyntax} environment shall appear only once, inside of a \texttt{codedescribe} environment. Take note, as well, this is not a verbatim environment!

For example, the code for \texttt{codedescribe} (entry above) is:

\texttt{\begin{codedescribe}[env,new=2023/05/01,update=2023/05/1,note={this is an example}]\begin{codesyntax}\ldots\end{codesyntax}\end{codedescribe}}

\begin{describelist}\ldots\end{describelist}

This sets a \texttt{description} like 'list'. In the non-star version the \{items-name\} will be typeset on the marginpar. In the star version, \{item-description\} will be indented by \{item-indent\} (defaults to: 20mm). \{obj-type\} defines the object-type format used to typeset \{item-name\}.

\begin{describe}\ldots\end{describe}

This is the \texttt{describelist} companion macro. In case of the \texttt{describe*}, \{item-name\} will be typeset in a box \{item-indent\} wide, so that \{item-description\} will be fully indented, otherwise \{item-name\} will be typed in the marginpar.
3.4 Commands

\typesetobj \typesetobj [{\(obj\)-type}] {{\(csv\)-list}}
\tsobj \tsobj [{\(obj\)-type}] {{\(csv\)-list}}

This is the main typesetting command (most of the others are based on this). It can be used to typeset a single 'object' or a list thereof. In the case of a list, each term will be separated by commas. The last two by \texttt{sep} (defaults to: and).

\textbf{Note:} One can change the last 'separator' with the key \texttt{sep}, for instance \texttt{\tsobj [env,sep=or] \{\}} (in case one wants to produce an 'or' list of environments). Additionally, one can use the key \texttt{comma} to change the last separator to a single comma, like \texttt{\tsobj [env,comma] \{}}.

\typesetargs \typesetargs [{\(obj\)-type}] {{\(csv\)-list}}
\tsargs \tsargs [{\(obj\)-type}] {{\(csv\)-list}}

Those will typeset \texttt{{\(csv\)-list}} as a list of parameters, like \texttt{[(arg1)] [(arg2)] [(arg3)]}, or \texttt{[(arg1)]{(arg2)}{(arg3)}}, etc. \texttt{\{\(obj\)-type\}} defines the formatting AND kind of brackets used (see 3.2): \texttt{[\]} for optional arguments (oarg), \texttt{\{} for mandatory arguments (marg), and so on.

\typesetmacro \typesetmacro {\(macro\)-list} {{\(oargs\)-list}}\{\(margs\)-list\}
\tsmacro \tsmacro {\(macro\)-list} {{\(oargs\)-list}}\{\(margs\)-list\}

This is just a short-cut for \texttt{\tsobj[code]{macro-list} \tsargs[oarg]{oargs-list} \tsargs[marg]{margs-list}}.

\typesetmeta \typesetmeta {\(name\)}
\tsmeta \tsmeta {\(name\)}

Those will just typeset \texttt{(name)} between left/right 'angles' (no other formatting).

\typesetverb \typesetverb [{\(obj\)-type}] {{\(verbatim\) text}}
\tsverb \tsverb [{\(obj\)-type}] {{\(verbatim\) text}}

Typesets \texttt{{\(verbatim\) text}} as is (verbatim...). \texttt{\(obj\)-type} defines the used format. The difference with \texttt{\tsobj [verb]{something}} is that \texttt{(something)} can contain commas (which, otherwise, would be interpreted as a list separator in \texttt{\tsobj}).

\textbf{Note:} This is meant to short expressions, and not multi-line, complex code (one is better of, then, using 2.2). \texttt{\(verbatim\) text} must be balanced ! otherwise, some low level \TeX{} errors may pop out.

\typesetmarginnote \typesetmarginnote {\(note\)}
\tsmarginnote \tsmarginnote {\(note\)}

Typesets a small note at the margin.

\begin{tsremark} \[\texttt{N.B.}\] \end{tsremark}
\texttt{tsremark}

The environment body will be typeset as a text note. \texttt{(N.B.)} (defaults to Note:) is the note begin (in boldface). For instance:

\begin{tabular}{|c|c|}
\hline
\texttt{\LaTeX{} Code:} & \texttt{\LaTeX{} Result:} \\
\hline
Sample text. Sample test. \begin{tsremark} \[\texttt{N.B.}\] This is an example. \end{tsremark} & Sample text. Sample test. \texttt{\textbf{N.B.}} This is an example. \\
\hline
\end{tabular}
3.5 Auxiliary Command / Environment

In case the used Document Class redefines the \maketitle command and/or abstract environment, alternatives are provided (based on the article class).

\begin{verbatim}
\typesettitle \{\{title-keys\}\}
\tstitle \{\{title-keys\}\}
\end{verbatim}

This is based on the \maketitle from the article class. The \{title-keys\} are:

- \textbf{title} The used title.
- \textbf{author} Author’s name. It’s possible to use \footnote command in it.
- \textbf{date} Title’s date.

\begin{verbatim}
\begin{tsabstract}
...
\end{tsabstract}
\end{verbatim}

This is the \texttt{abstract} environment from the \texttt{article} class.

\begin{verbatim}
\typesetdate \{\{title-keys\}\}
\tsdate \{\{title-keys\}\}
\end{verbatim}

\texttt{new: 2023/05/16}

This provides the current date (Month Year, format).