metakeys.sty: A generic framework for extensible Metadata in \LaTeX^∗

Michael Kohlhase
FAU Erlangen-Nürnberg
http://kwarc.info/kohlhase
March 20, 2019

Abstract

The metakeys package is part of the \sTeX{} collection, a version of \TeX{}/\LaTeX{} that allows to markup \TeX{}/\LaTeX{} documents semantically without leaving the document format, essentially turning \TeX{}/\LaTeX{} into a document format for mathematical knowledge management (MKM).

This package supplies the infrastructure for extending \sTeX{} macros with OMDoc metadata. This package is mainly intended for authors of \sTeX{} extension packages.

Contents

1 The User Interface 2
1.1 Package Options ........................................... 2
1.2 Adding Metadata Keys to Commands ......................... 2
1.3 Showing Metadata Keys/Values ............................ 3

2 Limitations 4

3 The Implementation 4
3.1 Package Options ........................................... 4
3.2 Adding Metadata Keys ........................................ 4
3.3 Showing Metadata Keys/Values ............................ 6
3.4 Using better defaults than empty .......................... 7

∗Version v0.9 (last revised 2019/03/20)
1 The User Interface

Many of the \LaTeX{} macros and environments take an optional first argument which uses key/value pairs to specify metadata relations of the marked up objects. The metakeys package supplies the infrastructure managing these key/value pairs. It also forms the basis for the rdfmeta package which allows to use these for flexible, user-extensible metadata relations (see [Koh16] for details).

1.1 Package Options

showmeta The metakeys package takes a single option: showmeta. If this is set, then the metadata keys defined by the \addmetakey are shown (see 1.3).

1.2 Adding Metadata Keys to Commands

Key/value pairs in \LaTeX{} are organized in key groups: every \LaTeX{} macro and environment that takes a key/value argument has an associated key group, and only keys that are registered in this group can be utilized. The metakeys package supplies the \addmetakey macro to add a new key to a key group: If \texttt{⟨group⟩} is the name of a key group (key) is a metadata keyword name, then

\addmetakey[⟨default⟩]{⟨group⟩}{⟨key⟩}{⟨dval⟩}

registers (key) in the metadata group (group), with an optional values (default) and (dval) for (key). (default) is the default value for (key), if it is not specified, and (dval) is the value (key) gets, if (key) is given without specifying a value. These two defaults are often used as

\addmetakey[false]{⟨group⟩}{⟨key⟩}{true}

Then, the value of (key) is \texttt{false} if (key) is not given and \texttt{true}, if (key) is specified without value. This is often the best way if we want to use (key) as an indicator to have a feature of name (key) (we can test that with \texttt{\ifx\⟨group⟩@⟨key⟩@true}, if we prepared the macro \texttt{\def@true{true}} earlier).

The keys registered for a metadata group can be used for defining macros with a key/value arguments via the \metasetkeys macro, see for instance the the definition in Figure 1. This macro is used exactly like the \setkeys macro from the keyval package [Car99], but integrates custom initialization and draft display functionality. This usage is mostly for package designers. There is another: If a macro or environment cannot be extended by an optional argument, e.g. because another package already does so (e.g. the document environment is extended – by redefining it – by various packages, which causes problems), the \metasetkeys macro can be used directly.

\addmetalistkey The \addmetalistkey macro is a variant of \addmetakey that adds a list-valued metadata key. The \addmetalistkey{foo}{val} in Figure 1 would allows to use multiple occurrences of the \texttt{val} keys in the metadata argument of \texttt{\foo}, the values of the \texttt{val} keys are collected as a comma-separated list in the token
register \texttt{\foo@vals}. Note that the \texttt{val} key can also deal with comma-separated lists for convenience.

With these definitions in a used package\footnote{Recall that the @ character is only allowed in packages, where comma-separated lists can be iterated over e.g. by the \texttt{@for} macro.} an invocation of

\begin{verbatim}
foo[type=bar,id=f4711,val=4,val=7,val={1,1}]
\end{verbatim}

is formatted to

\begin{verbatim}
I have seen a \texttt{foo} of type \texttt{bar} with identifier \texttt{f4711} and values \texttt{4}, and \texttt{7}, and \texttt{1}, and \texttt{1}!
\end{verbatim}

\begin{verbatim}
\def\@yes{yes}
newcommand\foo[1]{\metasetkeys{foo}{#1}
  \ifx\foo@visible\@yes % testing for visibility
    I have seen a \texttt{\emph{foo}} of type \texttt{\foo@type} with identifier \texttt{\foo@id} and values \texttt{\foo@vals}.
  \let\@join=\relax\def\@thejoin{, and }
  \@for\@I:=\foo@vals\do{\@join\@I\let\@join=\@thejoin}!
  fi}
\end{verbatim}

**Example 1:** Defining a macro with metadata

1.3 Showing Metadata Keys/Values

If the \texttt{showmeta} package option is set, the \texttt{metakeys} package sets an internal switch that shows the values of all keys specified with the \texttt{addmetakey} macro. The default behavior is to write the key/value pairs into the margin as \langle \texttt{key} \rangle : \langle \texttt{value} \rangle. Package designers can customize this behavior by redefining the \texttt{\metakeys@show@key} and \texttt{\metakeys@show@keys} macro.

\begin{verbatim}
\metakeys@show@key{(key)}{(value)} shows the a single key value pair, and
\metakeys@show@keys{(group)}{(keys)} shows the a list of keys metadata, by default we disregard the \langle group \rangle and show \langle keys \rangle in a marginpar.
\end{verbatim}

For keys that should not be shown in this manner, the \texttt{addmetakey*} macro has a variant \texttt{addmetakey*}. Its behavior is exactly the same, only that it keeps the key from being shown by the \texttt{showmeta} option.

Note that setting the \texttt{showmeta} option will enable metadata presentation on the whole document. But sometimes we want to disable that, e.g. inside figures, where \texttt{marginpar} is not allowed. Therefore the \texttt{metakeys} package provides the \texttt{hidemetakeys} macro that reverses this. The \texttt{showmetakeys} macro re-enables metadata presentation.
2 Limitations

In this section we document known limitations. If you want to help alleviate
them, please feel free to contact the package author. Some of them are currently
discussed in the \TeX{} GitHub repository \cite{Stex}.

1. none reported yet

3 The Implementation

3.1 Package Options

We declare some switches which will modify the behavior according to the package
options. Generally, an option \xxx{} will just set the appropriate switches to true
(otherwise they stay false). First we have the general options

1 ⟨∗\package⟩
2 \newif\ifmetakeys@showmeta\metakeys@showmetafalse
3 \DeclareOption{showmeta}{\metakeys@showmetatrue}
4 \DeclareOption*{ }
5 \ProcessOptions

We build on the keyval package which we first need to load. For \LaTeX{}XML,
we also initialize the package inclusions.

6 \RequirePackage{keyval}[1997/11/10]
7 \RequirePackage{etoolbox}

3.2 Adding Metadata Keys

\addmetakey

The \addmetakey macro looks at the next character and invokes helper macros
accordingly.

8 \newcommand\addmetakey{\@ifstar\addmetakey@star\addmetakey@nostar}%
\addmetakey@star

\addmetakey@star takes care of the starred form of \addmetakey. An in-
vocation of \addmetakey@star{(default)}{(group)}{(key)} macro first extends
the \metakeys@clear{group}@keys macro then defines the key \langle key \rangle
with the \define@key macro from the keyval package. This stores the key value given in
the local macro \langle group \rangle\langle key \rangle.

9 \newcommand\addmetakey@star[3][{}% 
10 \@ifnextchar[% 
11 \addmetakey@star@aux[#1]{#2}{#3}{}% 
12 ]% 
13 \addmetakey@star@aux[#1]{#2}{#3}[]% 
14 % 
15 ]% 
16 \def\addmetakey@star@aux[#1][#2][#3][#4][% 
17 \metakeys@ext@clear@keys{#2}{#3}{#1}% 
18 \metakeys@initialize@showkeys{#2}% 
19 \define@key{#2}{#3}{#4}%
\addmetakey@nostar \addmetakey@nostar takes care of the no-starred form of \addmetakey by first extending the \metakeys\langle group\rangle@showkeys macro which contains those keys that should be shown and then calling \addmetakey@star.

\newcommand\addmetakey@nostar[3][3]{\metakeys@ext@showkeys{#2}{#3}\addmetakey@star[#1]{#2}{#3}}

\metasetkeys The \metasetkeys\langle group\rangle clears/presets the key of \langle group\rangle via \clear@\langle group\rangle@keys, (if the showmeta option is set) shows them, and then sets the keys via keyvals \setkeys command.

\newcommand\metasetkeys[2][2]{\@nameuse{clear@#1@keys}\setkeys{#1}{#2}\ifmetakeys@showmeta% \edef\@@keys{\@nameuse{#1@showkeys}}% \metakeys@show@keys{#1}{% \@for\@I:=\@@keys\do{\metakeys@show@keyval{#1}{\@I}}% }% \fi%}

\metakeys@ext@clear@keys \metakeys@ext@clear@keys\langle group\rangle\langle key\rangle\langle default\rangle extends (or sets up if this is the first \addmetakey for \langle group\rangle) the \clear@\langle group\rangle@keys macro to set the default value \langle default\rangle for \langle key\rangle. The \clear@\langle group\rangle@keys macro is used in the generic \metasetkeys macro below. The variant \metakeys@ext@clear@keys is provided for use in the sref package.

\newrobustcmd\metakeys@ext@clear@keys[3][3]{\@metakeys@ext@clear@keys{#1}{#1@#2}{#3}}

\addmetalistkey \addmetalistkey\newrobustcmd\addmetalistkey{% \@ifstar\addmetalistkey@star\addmetalistkey@nostar%}

\newrobustcmd\addmetalistkey@star[3][3]{\metakeys@ext@clear@keys{#2}{#3}{#1}}
3.3 Showing Metadata Keys/Values

\metakeys@initialize@showkeys \metakeys@initialize@showkeys{⟨group⟩} sets up the ⟨group⟩@showkeys macro which is used to store the keys to be shown of the metadata in the generic \setmetakeys macro below.

\newrobustcmd\metakeys@initialize@showkeys[1]{%\@ifundefined{#1@showkeys}{\csdef{#1@showkeys}{}%}{%\csedef{#1@showkeys}{\csuse{#1@showkeys},#1}%}}%

\metakeys@ext@showkeys \metakeys@ext@showkeys{⟨group⟩}{⟨key⟩} extends (or sets up) the ⟨group⟩@showkeys macro which is used to store the keys to be shown of the metadata in the generic \setmetakeys macro below.

\newrobustcmd\metakeys@ext@showkeys[2]{%\@ifundefined{#1@showkeys}{\csdef{#1@showkeys}{#2}{}%}{%\csedef{#1@showkeys}{\csuse{#1@showkeys},#2}%}}%

\metakeys@show@key \metakeys@show@key{⟨key⟩}{⟨value⟩} shows the a single key value pair, as a default we just write ⟨key⟩:⟨value⟩.

\newrobustcmd\metakeys@show@key[2]{%\edef\@test{#2}\\ifx\@test\@empty\else #1:#2\quad\fi%}

\metakeys@show@keys \metakeys@show@keys{⟨group⟩}{⟨keys⟩} shows the metadata, by default we disregard the ⟨group⟩ and show ⟨keys⟩ in a marginpar.

\newrobustcmd\metakeys@show@keys[2]{\marginpar{⟨scriptsize #2⟩}}%
\meta{metakey} shows the key/value pair of a given key \textit{(key)}.

\newrobustcmd\meta{metakey}[2]{%\expandafter\@metakeys@show\csname #1@#2\endcsname{#2}%}

\showmetakeys
\newrobustcmd\showmetakeys{\meta{metakeys@showmetatrue}}%

\hidemetakeys
\newrobustcmd\hidemetakeys{\meta{metakeys@showmetafalse}}%

3.4 Using better defaults than empty

\addmetakeynew is an experimental version of \addmetakey which gives \omd@unspecified as an optional argument, so that it is used as the default value here and then test for it in \omfidus. But unfortunately, this does not work yet.

\newrobustcmd\addmetakeynew[3][4]{%\metakeys@ext@clear@keys{#2}{#3}{#1}%\define@key{#2}{#3}{%\csgdef{#2@#3}{##1}%%}}%

EdN:1\meta{metakeys@unspecified} An internal macro for unspecified values. It is used to initialize keys.\footnote{EdNote: MK: we could probably embed an package error or warning in here}

\newrobustcmd\meta{metakeys@unspecified}{an metakeys-defined key left unspecified}%

\metakeysifus This just tests for equality of the first arg with \meta{metakeys@unspecified}

\newrobustcmd\metakeysifus[4]{%\message{testing #1@#2=\csname#1@#2\endcsname}%%%%%%%%%%%%%%%%%%}%

\endinput
Change History

v0.1

General: First version . . . . . . . . . 1

v0.8

General: This is almost done . . . . 1

v0.9

General: make sure that showkeys is always initialized . . . . . . . . . 1

References


[sTeX] KWARC/sTeX. URL: \url{https://github.com/KWARC/sTeX} (visited on 05/15/2015).