

metakeys.sty: A generic framework for extensible Metadata in L^AT_EX^{*}

Michael Kohlhase
Jacobs University, Bremen
<http://kwarc.info/kohlhase>

November 6, 2010

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	2
2	Limitations	3
3	The Implementation	3
3.1	Package Options	3
3.2	Adding Metadata Keys	4
3.3	Showing Metadata Keys/Values	5
3.4	Using better defaults than empty	5
3.5	Finale	6

^{*}Version v0.9 (last revised 2010/09/02)

1 The User Interface

The `metakeys` package supplies the infrastructure for extending `STEX` macros with ontology-based metadata. The `metakeys` infrastructure is intended to support the new metadata infrastructure for the OMDoc format [Koh06] introduced in OMDoc1.3 [Koh10].

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

`\addmetakey` If $\langle group \rangle$ is the name of a metadata key-group $\langle key \rangle$ is a metadata keyword name, then

```
\addmetakey[\langle default \rangle]{\langle group \rangle}{\langle key \rangle}
```

`\metasetkeys` registers $\langle key \rangle$ in the metadata group $\langle group \rangle$, with an optional value $\langle default \rangle$ for $\langle key \rangle$. 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 definition in Figure 1. With these definitions in a used package¹ `\foo[type=bar,id=f4711]` is formatted to

I have seen a *foo* of type **bar** with identifier **f4711**!

```
\addmetakey{foo}{id}
\addmetakey{foo}{type}
\addmetakey[yes]{foo}{visible}
\def\@yes{yes}
\newcommand\foo[1][\metasetkeys{foo}{#1}]
\ifx\foo@visible\@yes % testing for visibility
I have seen a \emph{foo} of type \texttt{\foo@type} with identifier
\texttt{\foo@id}
\fi}
```

Example 1: Defining a macro with metadata

1.3 Showing Metadata Keys/Values

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

<code>\metakeys@show@key</code> <code>\metakeys@show@keys</code> <code>\addmetakey*</code> <code>\hidemetakeys</code> <code>\showmetakeys</code>	<p><code>\metakeys@show@key{⟨key⟩}{⟨value⟩}</code> shows the a single key value pair, and</p> <p><code>\metakeys@show@keys{⟨group⟩}{⟨keys⟩}</code> shows the a list of keys metadata, by default we disregard the <code>⟨group⟩</code> and show <code>⟨keys⟩</code> in a marginpar.</p> <p>For keys that should not be shown in this manner, the <code>\addmetakey</code> macro has a variant <code>\addmetakey*</code>. Its behavior is exactly the same, only that it keeps the key from being shown by the <code>showmeta</code> option.</p> <p>Note that setting the <code>showmeta</code> option will enable metadata presentation on the whole document. But sometimes we want to disable that, e.g. inside figures, where <code>\marginpar</code> is not allowed. Therefore the <code>metakeys</code> package provides the <code>\hidemetakeys</code> macro that reverses this. The <code>\showmetakeys</code> macro re-enables metadata presentation.</p>
--	---

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` TRAC [Ste].

1. none reported yet

3 The Implementation

The `metakeys` package generates two files: the `LATEX` package (all the code between `⟨*package⟩` and `⟨/package⟩`) and the `LATEXML` bindings (between `⟨*ltxml⟩` and `⟨/ltxml⟩`). We keep the corresponding code fragments together, since the documentation applies to both of them and to prevent them from getting out of sync.

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}

```

Finally, we need to declare the end of the option declaration section to `LATEX`.

```

4 \ProcessOptions

```

We build on the `keyval` package which we first need to load.

```

5 \RequirePackage{keyval}[1997/11/10]
6 ⟨/package⟩

```

¹The `~` character is only allowed in packages.

3.2 Adding Metadata Keys

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

```
7 <*package>
8 \newcommand\addmetakey{\@ifstar\addmetakey@star\addmetakey@nostar}
9 </package>
```

`\addmetakey@star` `\addmetakey@star` takes care of the starred form of `\addmetakey`. An invocation of `\addmetakey@star{<default>}{<group>}{<key>}` macro first extends the `\metakeys@clear@<group>@keys` macro then defines the key `<key>` with the `\define@key` macro from the `keyval` package. This stores the key value given in the local macro `\<group>@<key>`.

```
10 <*package>
11 \newcommand\addmetakey@star[3][\metakeys@ext@clear@keys{#2}{#3}{#1}%
12 \metakeys@initialize@showkeys{#2}%
13 \define@key{#2}{#3}[#1]{\expandafter\gdef\csname #2@#3\endcsname{##1}}}
```

`\addmetakey@nostar` `\addmetakey@nostar` takes care of the starred form of `\addmetakey` by first extending the `\metakeys@<group>@showkeys` macro which contains those keys that should be shown and then calling `\addmetakey@star`.

```
14 \newcommand\addmetakey@nostar[3][\metakeys@ext@showkeys{#2}{#3}\addmetakey@star[#1]{#2}{#3}%
15 </package>
```

`\metasetkeys` The `\metasetkeys{<group>}` clears/presets the key of `<group>` via `\clear@<group>@clearkeys`, (if the `showmeta` option is set) shows them, and then sets the keys via `keyvals` `\setkeys` command.

```
16 <*package>
17 \newcommand\metasetkeys[2]{\@nameuse{clear@#1@keys}\setkeys{#1}{#2}%
18 \ifmetakeys@showmeta%
19 \edef\@keys{\@nameuse{#1@showkeys}}%
20 \metakeys@show@keys{#1}{\@for\@I:=\@keys\do{\metakeys@show@keyval{#1}{\@I}}}%
21 \fi}
22 </package>
```

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

```
23 <*package>
24 \newcommand\metakeys@ext@clear@keys[3]{\@metakeys@ext@clear@keys{#1}{#1@#2}{#3}}
25 \newcommand\@metakeys@ext@clear@keys[3]{\@ifundefined{clear@#1@keys}%
26 {\expandafter\def\csname clear@#1@keys\endcsname%
27 {\expandafter\gdef\csname #2\endcsname{#3}}}%
28 {\expandafter\g@addto@macro\csname clear@#1@keys\endcsname%
29 {\expandafter\gdef\csname #2\endcsname{#3}}}%
30 </package>
```

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.

```

31 ⟨*package⟩
32 \newcommand\metakeys@initialize@showkeys[1]%
33 {\@ifundefined{#1@showkeys}{\expandafter\def\csname #1@showkeys\endcsname{}}{}}%

```

`\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.

```

34 \newcommand\metakeys@ext@showkeys[2]{\@ifundefined{#1@showkeys}%
35 {\expandafter\def\csname #1@showkeys\endcsname{#2}}%
36 {\expandafter\edef\csname #1@showkeys\endcsname{\csname #1@showkeys\endcsname,#2}}}

```

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

```

37 \newcommand\metakeys@show@key[2]{\metakeys@show@key{#2}{#1}}
38 \newcommand\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.

```

39 \newcommand\metakeys@show@keys[2]{\marginpar{\scriptsize #2}}

```

`\metakeys@show@keyval` `\metakeys@show@keyval{⟨group⟩}\meta{key}` shows the key/value pair of a given key `⟨key⟩`.

```

40 \newcommand\metakeys@show@keyval[2]{\expandafter\@metakeys@show@key\csname #1@#2\endcsname{#2}}
41 ⟨/package⟩

```

`\showmetakeys`

```

42 ⟨package⟩\newcommand\showmetakeys{\metakeys@showmetatrue}
43 ⟨ltxml⟩DefConstructor('showmetakeys', '');

```

`\hidemetakeys`

```

44 ⟨package⟩\newcommand\hidemetakeys{\metakeys@showmetafalse}
45 ⟨ltxml⟩DefConstructor('hidemetakeys', '');

```

3.4 Using better defaults than empty

`\addmetakeynew` `\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.

```

46 ⟨*package⟩
47 \newcommand\addmetakeynew[3][\metakeys@ext@clear@keys{#2}{#3}{#1}%
48 \define@key{#2}{#3}{\expandafter\gdef\csname #2@#3\endcsname{##1}}}

```

EdNOTE: `\metakeys@unspecified` Ain internal macro for unspecified values. It is used to initialize keys.¹

```

49 \newcommand\metakeys@unspecified{an metakeys-defined key left unspecified}

\metakeysifus This just tests for equality of the first arg with \metakeys@unspecified
50 \newcommand\metakeysifus[4]{\message{testing #1@#2=\csname#1@#2\endcsname}%
51 \expandafter\ifx\csname #1@#2\endcsname\metakeys@unspecified{#3}\else{#4}\fi}
52 \end{package}

```

3.5 Finale

Finally, we need to terminate the file with a success mark for perl.

```

53 \ltxml{1;

```

¹EdNOTE: MK: we could probably embed an package error or warning in here

References

- [Koh06] Michael Kohlhase. *OMDOC – An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, Aug. 2006. URL: <http://omdoc.org/pubs/omdoc1.2.pdf>.
- [Koh10] Michael Kohlhase. “An Open Markup Format for Mathematical Documents OMDoc [Version 1.3]”. Draft Specification. 2010. URL: <https://svn.omdoc.org/repos/omdoc/branches/omdoc-1.3/doc/spec/main.pdf>.
- [Ste] *Semantic Markup for LaTeX*. Project Homepage. URL: <http://trac.kwarc.info/sTeX/> (visited on 12/02/2009).