

memoize-ext: Extensions for memoize

Clea F. Rees*

v0.4.2 11956 2026/06/04

Abstract

memoize-ext provides extensions for Živanović’s package `memoize` (2024). In particular, it supports the memoization of content in tagged PDFs and presentations produced with Wright’s `ltx-talk` (2026).

Contents

| | | |
|-----------|------------------------------|----------|
| I | Usage | 3 |
| 1 | Basics | 3 |
| 2 | expl3 | 4 |
| 3 | l3draw | 4 |
| 4 | ltx-talk | 4 |
| 5 | Tagged PDF | 5 |
| 5.1 | TikZ pictures | 5 |
| 5.2 | Other content | 6 |
| 5.2.1 | expl3 functions | 7 |
| II | Implementation | 8 |
| | memoize-ext | 8 |
| | memoize-ext-expl3 | 14 |
| | memoize-ext-l3draw | 21 |

*Bug tracker: codeberg.org/cfr/memoize-ext/issues | Code: codeberg.org/cfr/memoize-ext | Mirror: github.com/cfr42/memoize-ext

| | |
|-------------------------------|----|
| memoize-ext-sockets | 23 |
| memoize-ext-tag | 24 |
| memoize-ext-talk | 35 |

Part I

Usage

1 Basics

Usage is simple.

```
\documentclass{<class>}
\usepackage{memoize-ext}
```

The package will automatically load `memoize` and pass any unrecognised options onto that package.

Note that to create a tagged presentation with `ltx-talk`, the package should be loaded *after* the class¹.

```
\DocumentMetadata{tagging=on,lang=en-GB,pdfversion=2.0,pdfstandard=UA-2,}
\documentclass{ltx-talk}
\usepackage{memoize-ext}
```

If for any reason it is necessary to load the package *prior* to the class in a tagged PDF, then tagging must be explicitly requested. For example,

```
\DocumentMetadata{tagging=on,lang=cy,pdfversion=2.0,pdfstandard=ua-2,}
\RequirePackage[tag]{memoize-ext}
\documentclass{book}
```

If necessary, a small number of package options are available to customise which code is loaded.

`expl3 (opt.) = true|false`

Loads code supporting `expl3` syntax.

Default is `true`. Initially `false`.

`l3draw (opt.) = true|false`

Loads code supporting `l3draw`, if the package is loaded.

Default is `true`. Initially `true`.

`tag (opt.) = true|false`

Loads code supporting tagged PDF, if L^AT_EX's tagging code is activated when the package is loaded. Note that this is *not* true prior to `\documentclass`.

Default is `true`. Initially `true` if tagging is activated; `false` otherwise.

`talk (opt.) = true|false`

Loads code supporting `ltx-talk`, if the class is loaded.

¹Note that `ltx-talk` diverges from `beamer` here, a point to which I was oblivious when I wrote the initial version of this documentation.

Default is `true`. Initially `true`.

Note that the additional code is not loaded if a different class is used, regardless of this setting. The option is provided in case it is necessary to disable support for the class, without disabling other parts of `memoize-ext`.

2 expl3

`replicate expl fn (pgfkey)` Sets up advice to ‘replicate’ an `expl3` function.

This works similarly to the builtin support for commands created with `\NewDocumentCommand` etc. This means that it is not necessary to specify `args`.

Functions with w-type arguments are NOT supported. Attempting to use this key with such a function will result in an error. Such cases require custom handling and can be configured using the standard `memoize` keys.

`memoize-ext-l3draw.sty` demonstrates use of `replicate expl fn`:

```
\mmzset{%
  auto~csname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
  auto~csname={__draw_record_origin:}{run~if~memoizing,replicate~expl~fn,},
}
```

This code sets up a custom ‘collector’ and installs ‘advice’ which memoizes `\draw_begin:`. It further advises `__draw_record_origin:` so that if this function is found during memoization, it will be replicated in the `ccmemo`. This ensures that the origin is recorded correctly when the memoized picture is utilised, since we do not want to record its position when memoized.

The net result of this is that auto-memoization of `l3draw` pictures should (hopefully) ‘just work’.

3 l3draw

`sec:draw «< l3draw` pictures are auto-memoized by default. `memoize-ext-l3draw.sty` mostly exists to demonstrate use of `replicate expl fn`. »> `sec:draw`

4 ltx-talk

The code is based on that provided by `memoize` for `beamer` and supports the same options, except that ‘`talk`’ is substituted for ‘`bemaer`’.

`per overlay (pgfkey)` Equivalent to the `beamer` option of the same name.

`talk mode to prefix` Equivalent to `beamer mode to prefix`.

(*pgfkey*) The code uses and/or changes internal code from both `ltx-talk` and `memoize`. While the public interface for `memoize` is fairly stable, the internals may not be, and `ltx-talk` is highly experimental. The latter also uses a large number of experimental packages and makes extensive use of experimental L^AT_EX features.

The justification for publishing this part of `memoize-ext` is essentially that anybody using `ltx-talk` and `memoize` is already playing with fire, so it is better to have an unreliable extinguisher to hand than none at all.

A few things you should know, even if you do not want to:

- the code uses an internal `ltx-talk` boolean to drive extern creation and utilisation;
- `talk mode to prefix` relies on an internal `ltx-talk` string;
- to workaround incompatibilities between `memoize` and `pdfmanagement`, the code redefines an internal `memoize` macro².

5 Tagged pdf

If using the package to produce tagged PDF, note that the tagging support

- redefines the internal macro `\mmzIncludeExtern` during utilisation;
- relies on an internal string variable in `ltsockets`.

Correct tagging *requires* that unmemoizable code be marked as such, either manually or automatically. The package does this automatically for `tikzpicture`s which use an unsupported tagging plug, but does nothing in any other case. So if your picture uses `remember picture`, for example, you *must* mark the code as unmemoizable or disable tagging for the affected code. The package will warn you about this, but that is all it does.

5.1 TikZ pictures

If the content you wish to memoize is a `TikZ` picture, you probably do not need to do anything special, but note that the default `latex-lab` plug is *not* supported. You must use one of `alt`, `actualtext` or `artifact`.

If you use `alt` or `actualtext` in the optional argument to `tikzpicture`, the value will be recorded in the `ccmemo` for use during utilisation. If you set the value outside the `tikzpicture`, this is not necessary. In the latter case, the *extern* will not depend on the value given (unless you request that specifically).

Note that if you change the selected plug *and* you set this *outside* the picture, you must manually tell `memoize` it should recompile the picture, since the plug is recorded in the `ccmemo`, but the hash will not have changed.

Note that tagging is disabled during memoization and additionally *disabled for content which has just been memoized*. So when a run produces an extern, the memoized code will not be tagged at all.

²The redefinition injects code into the box `memoize` ships out which resets opacity before and after the memoized code is executed. This is required because `memoize` relies on primitive `shipout`, whereas the implementation of opacity in `pdfmanagement` relies on \LaTeX 's `shipout` routine.

Note that this package does *not* support forest. If your document uses `forest` (Živanović 2017), you should either disable memoization for these pictures or load `forest-ext`³ (Rees 2026).

The TikZ support is implemented by replacing plugs provided by `latex-lab` with versions designed for memoized content (L^AT_EX Project 2025b). Code is also installed into the same hooks `latex-lab` uses with rules to ensure this package’s has priority.

`mmzx (plug)` Plug for `tagssupport/tikz/picture/init`

If memoization is not active, the plug executes the `latex-lab default` plug.

If some option for this package is specifically configured, it is used. Otherwise, the code initialisation code at the start of the picture attempts to find a match for any configured `latex-lab` plug. In effect, this means that you should not need to change anything in your document if you use one of the three supported plugs.

If memoization is enabled but no suitable plug is found, a warning is issued and memoization aborted. Otherwise, code is inserted into the `ccmemo` to emulate the appropriate `latex-lab` plug. In most cases, this code simply calls the relevant `latex-lab` plugs.

Plugs for `tagssupport/tikz/picture/begin` and `tagssupport/tikz/picture/end`:

`mmzx/actualtext (plug)` Sets up the `ccmemo` to use the `latex-lab actualtext` plugs.

`mmzx/artifact (plug)` Sets up the `ccmemo` to use the `latex-lab artifact` plugs.

`mmzx/alt (plug)` This pair of plugs is the exception. Rather than writing a `ccmemo` which will invoke the `latex-lab alt` plugs, these plugs write a `ccmemo` which uses an alternative implementation of those plugs. The reimplementation uses *properties* (provided by the L^AT_EX format) rather than *rememberpicture* (provided by PGF/TikZ)⁴.

If everything looks OK, tagging is disabled for the current picture. This is efficient if memoization is successful, but may be problematic if memoization is aborted or fails. In this case, it may be necessary to mark the content as unmemoizable or to disable memoization for particular pictures, in order to ensure content is tagged correctly⁵.

5.2 Other content

If the content you wish to memoize is *not* a TikZ picture, you may need to read the remainder of this section.

Generic support is provided in the form of two sockets which are used directly before and directly after an extern is included during utilisation. By default, the sockets do nothing, but they may be used to inject code which wraps the included extern in a suitable tagging structure.

Plugs may be assigned to the sockets either by writing suitable code to the `ccmemo` or in the document itself. The TikZ support, for example, writes commands to the `ccmemo` which assign plugs analogous to the `latex-lab` plugs available for non-memoized pictures.

³This is not necessary if you use `prooftrees`, which will load the package automatically if required.

⁴I considered using the support provided for `\includegraphic`, but this would require more intrusive changes to the internals of `memoize` and would essentially duplicate bounding box calculations already completed during memoization.

⁵It would be possible to disable tagging only if memoization succeeds, but I am not sure whether the structure will be right in this case?

More precisely, the TikZ support now uses one of the wrapper functions the package provides to assist users for the most common cases.

`tagssupport/memoize/include/extern/before`

(*socket*) This socket receives three arguments during extern utilisation: the width, height and depth of the memoized content. The `alt` plug for TikZ, for example, uses these values to calculate the bounding box required to create a `Figure` structure with `alt` text.

This socket is used just before the extern is included in the document.

`tagssupport/memoize/include/extern/after`

(*socket*) This socket absorbs no arguments. During extern utilisation, it is used immediately after inclusion of an extern.

5.2.1 expl3 functions

Three functions are provided to help setup code for these sockets when an extern is utilised. They should be used either during memoization or to configure defaults for use during memoization.

In pseudo-code, all three write the equivalent of the following to the *ccmemo* for execution during utilisation.

```
\mmzxtagtoks={<expansion of \the\mmzxtagtoks>}%
\AssignSocketPlug{tagssupport/memoize/include/extern/before}{<plug for before>}%
\AssignSocketPlug{tagssupport/memoize/include/extern/after}{<plug for after>}%
```

The effect is that the specified plugs will be used before and after the *extern* is utilised and these may use the *toks* register `\mmzxtagtoks`, if appropriate. If `expl3` syntax is preferred, `\mmzx_tag_get_recorded:N` may be used instead.

```
\mmzx_tag_socket_plug_record:nnf{<plug for before>} {<plug for after>} {<tokens>}
```

(*fn.*) Write code to the *ccmemo* which assigns

- `<plug for before>` to the socket `tagssupport/memoize/include/extern/before`,
- `<plug for after>` to the socket `tagssupport/memoize/include/extern/after`
- and `<tokens>` to the *toks* register `\mmzxtagtoks`

during utilisation.

```
\mmzx_tag_socket_plug_record:nn{<plug for before>} {<plug for after>}
```

(*fn.*) Write code to the *ccmemo* which assigns

- `<plug for before>` to the socket `tagssupport/memoize/include/extern/before`,
- `<plug for after>` to the socket `tagssupport/memoize/include/extern/after`
- and the current contents of `\mmzxtagtoks` to the *toks* register `\mmzxtagtoks`

during utilisation.

`\mmzx_tag_socket_plug_record:`

(*fn.*) Write code to the *ccmemo* which assigns

- the plug currently installed in the socket `tagsupport/memoize/include/extern/before` to the socket `tagsupport/memoize/include/extern/before`,
- the plug currently installed in the socket `tagsupport/memoize/include/extern/after` to the socket `tagsupport/memoize/include/extern/after`
- and the current contents of `\mmzxtagtoks` to the toks register `\mmzxtagtoks`

during utilisation.

A fourth function is provided to access the contents of the toks register `\mmzxtagtoks`, in case `expl3` syntax is preferred.

`\mmzx_tag_get_recorded:N` *<token list>*

(*fn.*) Recovers the value from the toks register for the current extern during utilisation and stores it in the specified (local) token list variable. May be used in the definitions of plugs, as explained above for `\mmzxtagtoks`.

Part II

Implementation

A double underscore (`__`) or an ‘at’ (`@`) indicates an internal macro or key. These are liable to change without notice and should not be used elsewhere. Some additional macros are categorised in the same way, but are named differently to simplify use in memos⁶.

memoize-ext

`<*sty> <@@=mmzx>`

```
1 \NeedsTeXFormat{LaTeX2e}[2021-11-15]%
```

copied verbatim, excepting format from Joseph Wright’s `siunitx.sty` under LPPL

```
2 \@ifundefined{ExplLoaderFileDate}{%
3   \RequirePackage{expl3}%
4 }{}
```

almost verbatim from `siunitx.sty`

should check date requirement (copied from `chronos`)

```
5 \@ifl@t@r\ExplLoaderFileDate{2022-02-24}{%
6 }{%
```

⁶This follows `memoize`’s own practice.


```

7  \PackageError{memoize-ext}{Support package expl3 too old}
8  {%
9    You need to update your installation of the bundles 'l3kernel' and
10   'l3packages'.\MessageBreak
11   Loading memoize-ext will abort!%
12 }%
13  \endinput
14}%
15%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 \GetIdInfo $Id: memoize-ext.dtx 11956 2026-06-04 03:13:27Z cfrees $ {Extensions for
17 <debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
18 <debug> {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
19 <debug> \ProvidesExplPackage{\ExplFileName-debug}
20 <debug> {\ExplFileDate}{v0.4.2 \ExplFileVersion}{\ExplFileDescription}
21 %
22 \str_new:N \g__mmzx_name_str
23 \str_gset:NV \g__mmzx_name_str \ExplFileName
24 %
25 <debug> \disable@package@load {memoize-ext-debug}
26 <debug> \disable@package@load {memoize-ext}
27 {
28   Only one of memoize-ext and memoize-ext-debug should be loaded.
29   Since
30 <debug> memoize-ext
31 <debug> memoize-ext-debug
32   had been loaded,I will ignore your request for
33 <debug> memoize-ext
34 <debug> memoize-ext-debug
35 .}
36 \SetDefaultHookLabel{memoize-ext}

```

`\l__mmzx_opt_tag_bool (var.)` Set according to activation status by default.

```

37 \bool_new:N \l__mmzx_opt_tag_bool
38 \tag_if_active:TF
39 { \bool_set_true:N \l__mmzx_opt_tag_bool }
40 { \bool_set_false:N \l__mmzx_opt_tag_bool }

```

`\l__mmzx_opt_draw_bool (var.)` Other bools.

`\l__mmzx_opt_expl_bool (var.)`

```

41 \keys_define:nn {memoize-ext}
42 {
43 <!*debug>
44   debug .code:n = {
45     \PackageWarning{memoize-ext}{
46       To load the debugging code,use memoize-ext-debug instead of this package.
47     }
48   },
49 </!debug>
50   expl3 .bool_set:N = \l__mmzx_opt_expl_bool,
51   expl3 .default:n = true,
52   expl3 .initial:n = false,
53   l3draw .bool_set:N = \l__mmzx_opt_draw_bool,
54   l3draw .default:n = true,
55   l3draw .initial:n = true,

```

```

56 tag      .bool_set:N = \l_mmzx_opt_tag_bool,
57 tag      .default:n = true,
58 talk     .bool_set:N = \l_mmzx_opt_talk_bool,
59 talk     .default:n = true,
60 talk     .initial:n = true,
61 }

```

Joseph Wright: <https://chat.stackexchange.com/transcript/message/69011532#69011532>.

```

62 \IfFormatAtLeastTF { 2026-06-01 }{
63   \IfFormatAtLeastF { 2026-11-01 }{
64     \DeclareDocumentCommand \DeclareUnknownKeyHandler { 0 { \@currname } +m }
65     {
66       \cs_set_protected:cpn { __keys_unknown_handler_ #1 :nn } ##1##2 {#2}
67       \__keys_options_expand_module:Nn \keys_define:ne {#1}
68       {
69         unknown .code:n =
70         \exp_not:N \exp_args:NV
71         \exp_not:c { __keys_unknown_handler_ #1 :nn }
72         \exp_not:N \l_keys_key_str {##1}
73       }
74     }
75   }
76 }{}

```

Pass unknown options to memoize, loaded below.

```

77 \DeclareUnknownKeyHandler[memoize-ext]{
78   \PassOptionsToPackage{\CurrentOption}{memoize}
79 }

```

\IfFormatAtLeastTF Joseph Wright: from siunitx.sty ; <https://chat.stackexchange.com/transcript/message/64327823#64327823>

```

80 \providecommand \IfFormatAtLeastTF { \@ifl@t@r \fmtversion }

81 \IfFormatAtLeastTF { 2022-06-01 }
82 {
83   \ProcessKeyOptions [ memoize-ext ]
84 }{}
85 \RequirePackage { l3keys2e }
86 \ProcessKeyOptions { memoize-ext }
87 }

88 \IfFormatAtLeastTF { 2020-10-01 }{
89 }{
90   \RequirePackage { xparse }
91   \providecommand \ExpandArgs [1]
92   { \cs_if_exist_use:c { exp_args:N #1 } }
93 }

```

Should specify next version here, most probably. Or conditionalise input switch for ccmemos?

^^ ????

```

94 \RequirePackage{memoize}
95 <debug> \mmzset{

```

```

96 <debug>      trace,
97 <debug>      include context in ccmemo,
98 <debug>      }

```

Fix for `\toksapp` and friends courtesy of Max Chernoff <https://github.com/sasozivanovic/memoize/pull/57/commits>.

```

99 \sys_if_engine luatex:F
100 {
101   \clist_map_inline:nn {\toksapp,\etoksapp,\gtoksapp,\xtoksapp}
102   {
103     \tl_if_head_eq_meaning:VNF #1 \protected
104     {
105       \expandafter\protected\expandafter\def\expandafter#1\expandafter{#1}
106     }
107   }
108 }

```

`\kernel@after@shipout@background` bug fix: Ulrike Fischer: <https://chat.stackexchange.com/transcript/41?m=68852988#68852988> ‘Less internal’ internal macro: Jasper Habicht [GITHUB cfr42/prooftrees/issues/8](https://github.com/jasperhabicht/prooftrees/issues/8).issue #8. Lua_{La}T_EX can’t tolerate this addition to the shipout hook, whereas pdf_{La}T_EX produces an invalid PDF without it. I have no clue about other engines, but probably better to err on the side of caution and leave possibly-well alone.

My worry here is that if the contents of `\@kernel@after@shipout@background` changes, I have zero confidence the same problem won’t arise for pdf_{La}T_EX. *However*, a combination of `\protected` and `\noexpand` seems to work even if I use `\special{}` rather than the kernel hook *and* it avoids looping with Lua_{La}T_EX. So, even though I have no clue what is happening and even though Ulrike said she can’t see how it would help, since it does help, it seems better to err on the side of caution.

Better create an invalid PDF than looping infinitely not creating one at all

```

109 \cs_new_protected_nopar:Npn \@mmzx@kernel@after@shipout@background
110 {
111   \use:c {\@kernel@after@shipout@background}
112 }
113 \FirstAidNeededT{memoize}{sty}{2024/12/02 v1.4.1 Fast and flexible externalization}
114 \sys_if_engine_pdftex:T {
115   \hook_gput_code:nnn {begindocument/end} {.} {
116     \hook_gput_code:nnn {cmd/mmz@shipout@extern/before} {.} {
117       \noexpand\@mmzx@kernel@after@shipout@background
118     }
119   }
120 }
121 }

```

temporary variables, quarks

```

122 \bool_new:N \l__mmzx_tmpa_bool
123 \fp_new:N \l__mmzx_tmpa_fp
124 \int_new:N \l__mmzx_tmpa_int
125 \quark_new:N \q__mmzx_stop
126 \tl_new:N \l__mmzx_tmpa_tl
127 \tl_new:N \l__mmzx_tmpb_tl

```

```

128 \tl_new:N \l__mmzx_tmpc_tl
129 \seq_new:N \l__mmzx_tmpa_seq
130 \str_new:N \l__mmzx_tmpa_str
131 \str_new:N \l__mmzx_tmpb_str
132 <*debug>
133 \cs_new_protected:Npn \__mmzx_debug:n #1
134 {
135   \iow_log:n {[mmzx debug]:: #1}
136 }
137 \cs_generate_variant:Nn \__mmzx_debug:n {e}
138 \cs_new_protected:Npn \__mmzx_debug:N #1
139 {
140   \__mmzx_debug:e {\cs_to_str:N #1: \exp_args:NV \exp_not:n #1}
141 }
142 </debug>

```

__mmzx_noop: Do nothing successfully.

```

\__mmzx_noop:n
143 \cs_new:Npn \__mmzx_noop: {}
144 \cs_new_eq:NN \__mmzx_noop:n \use_none:n

```

tag, expl, l3draw, talk loaded conditionally

```

145 \bool_if:NT \l__mmzx_opt_tag_bool
146 {
147   <!debug> \RequirePackage{\g__mmzx_name_str -tag}
148   <debug> \RequirePackage{\g__mmzx_name_str -tag-debug}
149   \hook_gput_code:nnn {package/forest/after}{.}
150   {
151     \hook_gput_code:nnn {begindocument/before}{.}
152     {
153       \IfPackageLoadedF {forest-lib-ext.tagging}
154       {
155         \IfPackageLoadedF {forest-lib-ext.tagging-debug}
156         {
157           \msg_warning:nnnnnn {memoize-ext}{unsupported}{forest}
158           {forest-lib-ext.tagging.sty}{forest-ext}
159           {forest trees will not be correctly tagged and may cause fatal
160            compilation errors.}
161         }
162       }
163     }
164   }
165 }

```

memoize-ext-expl3[-debug]

```

166 \bool_if:NT \l__mmzx_opt_expl_bool
167 {
168   <!debug> \RequirePackage{\g__mmzx_name_str -expl3}
169   <debug> \RequirePackage{\g__mmzx_name_str -expl3-debug}
170 }

```

memoize-ext-l3draw[-debug]

```

171 \hook_gput_code:nnn {package/l3draw/after}{.}
172 {

```

```

173 \bool_if:NT \l__mmzx_opt_draw_bool
174 {
175 <!debug> \RequirePackage {\g__mmzx_name_str -l3draw}
176 <debug> \__mmzx_debug:n {Loading memoize-ext-l3draw-debug.}
177 <debug> \RequirePackage {\g__mmzx_name_str -l3draw-debug}
178 }
179 }

```

memoize-ext-talk[-debug]

```

180 \hook_gput_code:nnn {class/ltx-talk/after} {..}
181 {
182 \bool_if:NT \l__mmzx_opt_talk_bool
183 {
184 <!debug> \RequirePackage{memoize-ext-talk}
185 <debug> \__mmzx_debug:n {Loading memoize-ext-talk-debug.}
186 <debug> \RequirePackage{memoize-ext-talk-debug}
187 }
188 }

```

NaCl | halen | salt

```

189 \toksapp\mmzSalt{
190 Tagging status: \tag_if_active_p:
191 }

```

messages

```

192 \msg_new:nnnn {memoize-ext}{unsupported}
193 {
194 \msg_warning_text:n {memoize-ext}:
195 Non-existent or inappropriate version of #2 from #3 \msg_line_context:.
196 #4
197 } {
198 memoize-ext#1 requires an appropriate version of #2 from #3.
199 }

```

</sty>

memoize-ext-expl3

Clea F. Rees

11956 2026-05-26

Abstract

Provides memoize-ext-expl3 and memoize-ext-expl3-common. Part of memoize-ext.

Contents

<@@=mmzx> <*common>

```
200 \GetIdInfo $Id: memoize-ext-expl3.dtx 11956 2026-06-04 03:13:27Z cfrees $ {Extension
201 \!debug} \ProvidesExplPackage{\ExplFileName-common}{\ExplFileDate}{v0.0 %
202 \!debug} \ExplFileVersion}{\ExplFileDescription}
203 \!debug} \ProvidesExplPackage{\ExplFileName-common-debug}{\ExplFileDate}{v0.0 %
204 \!debug} \ExplFileVersion}{\ExplFileDescription}
205 %
206 \!debug} \disable@package@load {memoize-ext-expl3-common-debug}
207 \!debug} \disable@package@load {memoize-ext-expl3-common}
208 { Only one of memoize-ext-expl3-common and memoize-ext-expl3-common-debug
209 should be loaded.
210 Since
211 \!debug} memoize-ext-expl3-common
212 \!debug} memoize-ext-expl3-common-debug
213 has been loaded,I will ignore your request for
214 \!debug} memoize-ext-expl3-common
215 \!debug} memoize-ext-expl3-common-debug
216 .}

217 \!debug} \RequirePackage{memoize-ext}
218 \!debug} \RequirePackage{memoize-ext-debug}
219 %
```

We don't want inconsistent names in hooks.

```
220 \SetDefaultHookLabel{memoize-ext}
```

mmzx_replicating_bool (*var.*) Internal variable to track whether currently replicating.

```
221 \bool_new:N \l__mmzx_replicating_bool
222 \bool_set_false:N \l__mmzx_replicating_bool
```

```

mmzx_if_replicating:TF (fn.)
mmzx_if_replicating_p: (fn.)
223 \prg_new_conditional:Npnn \__mmzx_if_replicating: {p,T,TF,F}
224 {
225   \if_bool:N \l__mmzx_replicating_bool
226   <debug> \__mmzx_debug:n {Replicating true.}
227   \prg_return_true:
228   \else:
229   <debug> \__mmzx_debug:n {Replicating false.}
230   \prg_return_false:
231   \fi:
232 }

```

\AdviceRunIfNotReplicating Run condition.

```

233 \cs_new:Npn \AdviceRunIfNotReplicating
234 {
235   \__mmzx_if_replicating:F
236   {
237   <debug> \__mmzx_debug:n {Not replicating,so proceeding.}
238   \ifmemoizing \AdviceRuntrue \fi
239   }
240 }

```

if not replicating (*pgfkey*) Style.

```

241 \mmzset{
242   auto/run if not replicating/.style = {
243     run conditions={\AdviceRunIfNotReplicating},
244   },
245 }

```

Constants

```

246 \cctab_const:Nn \c__mmzx_expl_at_cctab {
247   \cctab_select:N \c_code_cctab
248   \makeatletter
249 }
250 \cctab_const:Nn \c__mmzx_nexpl_at_cctab {
251   \cctab_select:N \c_code_cctab
252   \makeatletter
253   \int_set:Nn \tex_endlinechar:D { 13 }
254   \char_set_catcode_space:n { 9 }
255   \char_set_catcode_space:n { 32 }
256   \char_set_catcode_active:n { 126 } % tilde
257 }

```

expl3, memoizing c  d ynddo

hooks instead of [T  X SE: David Carlisle](#)

\l__mmzx_expl_bool (*var.*) A boolean to track whether expl3 syntax is active or not. yn lle ateb | in place of [748807](#) by David Carlisle.

```

258 \bool_new:N \l__mmzx_expl_bool

```

```

_restore_ccmemo_input: (fn.) Initialise.
saved_mmzxExplAtBegin: (fn.)
x_saved_mmzxExplAtEnd: (fn.)
259 \cs_new_protected_nopar:Npn \__mmzx_restore_ccmemo_input: {}
260 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtBegin: {}
261 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtEnd: {}

```

Auto-switching for expl3 syntax.

```

262 \hook_gput_code:nnn {begindocument/end}{.}
263 {
264   \bool_set_false:N \l__mmzx_expl_bool
265 }

266 \hook_gput_code:nnn {begindocument/end}{.}
267 {
268   <debug> \__mmzx_debug:n {Tracking expl3 syntax changes.}
269   \bool_set_false:N \l__mmzx_expl_bool
270   \hook_gput_code:nnn {cmd/ExplSyntaxOn/before} { . }
271   {
272     \bool_if:NF \l__mmzx_expl_bool
273     {
274       \bool_set_true:N \l__mmzx_expl_bool
275       \ifmmz@direct@ccmemo@input
276         \relax
277       \else
278         \cs_set_protected_nopar:Npn \__mmzx_restore_ccmemo_input:
279         {
280           \mmz@direct@ccmemo@inputfalse
281         }
282       \fi
283       \mmz@direct@ccmemo@inputtrue
284       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtBegin: \mmzxExplAtBegin
285       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtEnd: \mmzxExplAtEnd
286       \__mmzx_expl_at_start:
287     }
288   }

```

\ExplSyntaxOn rewrites \ExplSyntaxOff, so it doesn't work to add hook code to \ExplSyntaxOff directly.

```

289 \hook_gput_code:nnn {cmd/ExplSyntaxOn/after} { . }
290 {
291   \hook_gput_code:nnn {cmd/ExplSyntaxOff/before} { . }
292   {
293     \bool_set_false:N \l__mmzx_expl_bool
294     \cs_set_eq:NN \mmzxExplAtBegin \__mmzx_saved_mmzxExplAtBegin:
295     \cs_set_eq:NN \mmzxExplAtEnd \__mmzx_saved_mmzxExplAtEnd:
296     \__mmzx_restore_ccmemo_input:
297   }
298 }
299 }

```

fns mewnol

__mmzx_cctab_end: (fn.) just for symmetry ...

```

300 \cs_new_eq:NN \__mmzx_cctab_end: \cctab_end:

```



```

\__mmzx_expl_at_begin: (fn.) Convenience wrappers for cat code changes.
\__mmzx_nexpl_at_begin: (fn.)
\__mmzx_expl_at_start: (fn.) 301 \cs_new_protected_nopar:Npn \__mmzx_expl_at_begin:
\__mmzx_nexpl_at_start: (fn.) 302 {
\__mmzx_cctab_stop: (fn.) 303 \cctab_begin:N \c__mmzx_expl_at_cctab
304 }
305 \cs_new_protected_nopar:Npn \__mmzx_nexpl_at_begin:
306 {
307 \cctab_begin:N \c__mmzx_nexpl_at_cctab
308 }
309 \cs_new_nopar:Npn \__mmzx_expl_at_start:
310 {
311 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_expl_at_begin:}
312 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
313 }
314 \cs_new_nopar:Npn \__mmzx_nexpl_at_start:
315 {
316 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_nexpl_at_begin:}
317 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
318 }
319 \cs_new_protected_nopar:Npn \__mmzx_cctab_stop:
320 {
321 \cs_set_nopar:Npn \mmzxExplAtBegin {relax}
322 \cs_set_nopar:Npn \mmzxExplAtEnd {relax}
323 }

```

toks memoize (cyhoeddus yn unig)

The code here is constant but the meaning changes, so what is added to the memos reflects the configuration at the time.

```

324 \appto\mmzAtBeginMemoization{
325 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
326 <debug> \string\mmzAtBeginMemoization.}
327 \xtoksapp\mmzCCMemo
328 {
329 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname
330 }
331 <debug> \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
332 }
333 <debug> \cs_log:N \mmzAtBeginMemoization
334 \preto\mmzAtEndMemoization{
335 <debug> \__mmzx_debug:n {Appending expl end to ccmemo at start
336 <debug> \string\mmzAtEndMemoization.}
337 \xtoksapp\mmzCCMemo
338 {
339 \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
340 }
341 }
342 <debug> \cs_log:N \mmzAtEndMemoization

```

init

```

343 \hook_gput_code:nnn { begindocument/end } {mmzx}
344 {
345 % % % % % \__mmzx_cctab_stop:
346 % }

```

</common>

<*sty>

```

347 \GetIdInfo $Id: memoize-ext-expl3.dtx 11956 2026-06-04 03:13:27Z cfrees $ {Extension
348 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
349 \!debug} {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
350 \debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
351 \debug} {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
352 %
353 \!debug} \disable@package@load {memoize-ext-expl3-debug}
354 \debug} \disable@package@load {memoize-ext-expl3}
355 { Only one of memoize-ext-expl3 and memoize-ext-expl3-debug
356 should be loaded.
357 Since
358 \!debug} memoize-ext-expl3
359 \debug} memoize-ext-expl3-debug
360 has been loaded,I will ignore your request for
361 \debug} memoize-ext-expl3
362 \!debug} memoize-ext-expl3-debug
363 .}

364 \!debug} \RequirePackage{memoize-ext}
365 \debug} \RequirePackage{memoize-ext-debug}
366 \!debug} \RequirePackage{memoize-ext-expl3-common}
367 \debug} \RequirePackage{memoize-ext-expl3-common-debug}
368 %

```

We don't want inconsistent names in hooks.

```
369 \SetDefaultHookLabel{memoize-ext}
```

expl3 replicators

todo: figure out how to maintain correct grouping ...

expl_replicate__bb_tl (var.) Variables

expl_replicate__ba_tl (var.)

expl_replicate__tb_tl (var.)

```

370 \tl_new:N \g__mmzx_expl_replicate__bb_tl
371 \tl_new:N \g__mmzx_expl_replicate__ba_tl
372 \tl_new:N \g__mmzx_expl_replicate__tb_tl
373 \tl_gput_right:NV \g__mmzx_expl_replicate__bb_tl \c_left_brace_str
374 \tl_gput_right:Nn \g__mmzx_expl_replicate__bb_tl {#}
375 \tl_gput_right:NV \g__mmzx_expl_replicate__ba_tl \c_right_brace_str
376 \tl_gput_right:Nn \g__mmzx_expl_replicate__tb_tl {#}

```

l_replicate_fn_aux:nnN (fn.) Generic auxiliary functions for replication. The first does the actual replicating; the
_expl_replicate__aux:n (fn.) second delegates details according to the argument specification.

```

377 \cs_new:Npn \__mmzx_expl_replicate_fn_aux:nnN #1#2#3
378 {
379 \cs_if_exist:cF { __mmzx_rep_#1:#2 }
380 {
381 \int_zero:N \l__mmzx_tmpa_int
382 \tl_clear:N \l__mmzx_tmpa_tl
383 \tl_clear:N \l__mmzx_tmpc_tl
384 \tl_map_function:nN {#2} \__mmzx_expl_replicate__aux:n

```

```

385 \cs_if_exist:cF { __mmzx_rep_#1:\l__mmzx_tmpc_tl }
386 {
387   \cs_gset_protected:ce {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
388   {
389     \xtoksapp\mmzCCMemo{
390       \exp_after:wN \exp_not:N \cs:w #1:#2 \cs_end: \l__mmzx_tmpa_tl
391     }
392     \exp_not:N \expandonce \exp_not:N \AdviceOriginal \l__mmzx_tmpa_tl
393     \exp_not:N \group_end:
394     \__mmzx_tmp_cctab_stop:
395   }
396 }
397 \str_if_eq:eeF {#2}{\l__mmzx_tmpc_tl}
398 { % ych!
399   \cs_new_eq:cc {__mmzx_rep_#1:#2} {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
400 }
401 }
402 \use:c { __mmzx_rep_#1:#2 }
403 }
404 \cs_new:Npn \__mmzx_expl_replicate__aux:n #1
405 {
406   \int_incr:N \l__mmzx_tmpa_int
407   \str_case:nnF { #1 }
408   {
409     {c} { \__mmzx_expl_replicate__b: }
410     {e} { \__mmzx_expl_replicate__b: }
411     {o} { \__mmzx_expl_replicate__b: }
412     {p} { }
413     {n} { \__mmzx_expl_replicate__b: }
414     {v} { \__mmzx_expl_replicate__b: }
415     {D} { }
416     {F} { \__mmzx_expl_replicate__b: }
417     {N} { \__mmzx_expl_replicate__t: }
418     {T} { \__mmzx_expl_replicate__b: }
419     {V} { \__mmzx_expl_replicate__t: }
420     {w} { \__mmzx_expl_replicate__e: }
421   }{
422     \__mmzx_expl_replicate__e:
423   }
424 }

```

`__mmzx_expl_replicate__b:` (*fn.*) Type-specific auxiliaries. We don't need to be very specific here. We just need to distinguish argument specifiers which expect braced groups from those which expect single tokens and from those we cannot automate.

```

425 \cs_new_nopar:Npn \__mmzx_expl_replicate__b:
426 {
427   \tl_put_right:Nn \l__mmzx_tmpc_tl {n}
428   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__bb_tl
429   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
430   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__ba_tl
431 }
432 \cs_new_nopar:Npn \__mmzx_expl_replicate__t:
433 {
434   \tl_put_right:Nn \l__mmzx_tmpc_tl {N}
435   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__tb_tl

```

```

436 \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
437 }
438 \cs_new_nopar:Npn \__mmzx_expl_replicate_e:
439 {
440 \PackageError{mmzx}{No do,sorry. Use replicate with args instead.}{}
441 }

```

`\mmzx_expl_replicate_fn: (fn.)` For replicating `expl3` functions.

```

__mmzx_tmp_cctab_stop: (fn.)
\mmzx@expl@replicate@fn
442 \cs_new_eq:NN \__mmzx_tmp_cctab_stop: \__mmzx_noop:
443 \cs_new:Npn \__mmzx_expl_replicate_fn:
444 {
445 \bool_set_false:N \l__mmzx_tmpa_bool
446 \str_if_eq:eeT {\mmzxExplAtEnd} {relax}
447 {
448 \bool_set_true:N \l__mmzx_tmpa_bool
449 \__mmzx_nexpl_at_start:
450 \xtoksapp\mmzCCMemo {
451 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname
452 }
453 \cs_set:Npn \__mmzx_tmp_cctab_stop:
454 {
455 \xtoksapp\mmzCCMemo {
456 \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
457 }
458 \__mmzx_cctab_stop:
459 }
460 }{
461 \cs_set_eq:NN \__mmzx_tmp_cctab_stop: \__mmzx_noop:
462 }
463 \group_begin:
464 \bool_set_true:N \l__mmzx_replicating_bool
465 \exp_last_unbraced:Ne \__mmzx_expl_replicate_fn_aux:nnN
466 { \exp_args:NV \cs_split_function:N \AdviceReplaced }
467 }
468 \cs_new_eq:NN \mmzx@expl@replicate@fn \__mmzx_expl_replicate_fn:

```

`\o/replicate expl fn (pgfkey)` By default we handle `\tex_savepos:D` since this commonly requires replication in `\o/replicate expl var (pgfkey)` the kinds of environments typically subject to memoization. Another candidate is `\int_gincr:N`, but that results in a large number of additions and it is not at all clear these are generally required or desirable. Don't do this. It breaks stuff.

```

469 \mmzset{% config <<<
470 auto/replicate expl fn/.style={
471 run if not replicating,
472 outer handler=\mmzx@expl@replicate@fn,
473 },
474 }% >>>

```

</sty>

memoize-ext-l3draw

Clea F. Rees

11956 2026-05-26

Abstract

Support for auto-memoization of content created with l3draw (L^AT_EX Project 2025a).
memoize-ext-l3draw is part of memoize-ext.

Contents

<*sty> <@@=mmzx>

```
475 \GetIdInfo $Id: memoize-ext-l3draw.dtx 11956 2026-06-04 03:13:27Z cfrees $ {Extensi
476 <!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
477 <!debug> {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
478 <debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
479 <debug> {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
480 %
481 <!debug> \disable@package@load {memoize-ext-l3draw-debug}
482 <debug> \disable@package@load {memoize-ext-l3draw}
483 { Only one of memoize-ext-l3draw and memoize-ext-l3draw-debug
484 should be loaded.
485 Since
486 <!debug> memoize-ext-l3draw
487 <debug> memoize-ext-l3draw-debug
488 has been loaded,I will ignore your request for
489 <debug> memoize-ext-l3draw
490 <!debug> memoize-ext-l3draw-debug
491 .}
492 %
493 <!debug> \RequirePackage{memoize-ext}
494 <debug> \RequirePackage{memoize-ext-debug}
495 <!debug> \RequirePackage{memoize-ext-expl3}
496 <debug> \RequirePackage{memoize-ext-expl3-debug}
```

l3draw

mmzx_draw_id_begin_int (*var.*) Temporary int.

```
497 \int_new:N \g__mmzx_draw_id_begin_int
```

ce_collect_draw_args:w (*fn.*) The l3draw picture environment is essentially a function with a weird argument specification, so we use a custom collector.

```

498 \cs_new:Npn \__mmzx_advice_collect_draw_args:w #1 \draw_end:
499 {
500   \toks0={ #1 \draw_end: }
501   \exp_args:No \AdviceInnerHandler {\the\toks0}
502 }

```

`\AdviceCollectDrawArguments` Public wrapper.

```

503 \cs_new:Npn \AdviceCollectDrawArguments{
504   \toks0 = {}
505   \__mmzx_advice_collect_draw_args:w
506 }

```

Default configuration. This replicates changes to `\g_draw_id_int`, but does so by executing code at the start and end of *every* memoization. There must be a more efficient way to do this

```

507 \mmzset{%
508   gincr draw id/.style={
509     at begin memoization={
510       \gtoksapp\mmzCCMemo
511       {
512         \UseName{int_gincr:c} {g_draw_id_int}
513       }
514     },
515   },
516   auto csname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,gincr
draw id,},
517   auto csname={__draw_record_origin:}{run if memoizing,replicate expl fn,},
518 }

```

</sty>

memoize-ext-sockets

Clea F. Rees

11956 2026-05-26

Abstract

memoize-ext-sockets is part of memoize-ext.

Contents

<*sty> <@@=mmzx>

ltsockets

socket_assigned_plug:n (fn.) Returns the name of the plug assigned to the specified socket. This ought not use a variable internal to the format’s code, but there does not seem to be a public interface. So it may break, but for now it works.

```
519 \cs_new_nopar:Npn \__mmzx_socket_assigned_plug:n #1
520 { % rhybudd: fn mewno1
521   \str_use:c { l__socket_#1_plug_str }
522 }
```

</sty>

memoize-ext-tag

Clea F. Rees

11956 2026-05-26

Abstract

`memoize-ext-tag` is part of `memoize-ext`. It supports tagging memoized content/the memoization of tagged content.

Contents

Uses and/or redefines the following internals, primitives and other nefarious methods:

- `\l__socket_assigned_plug:n` thing
 - If a public interface for retrieving the plug is provided at some point, I will see if I can use that. However, they do not wish it to be expandable. This is manageable, but it is very nice having an expandable function here, so I will see if I realise, remember and can do it not-too-painfully, I guess.
- `latex-lab` variables, keys etc.
 - This is fairly fragile: patching patches to make them work with patched patches
 - But I guess the `l3keys` and `pgfkeys` are public. I'm not sure about the sockets/plugs. Are these supposed to be used by external code?
 - The use of `l__tikz_tagging_alt_tl` and `l__tikz_tagging_actualextext_tl` is definitely ungood, but I do not currently see a better way. Hopefully most people will use the `pgfkeys` interface, as that's much more natural here?
- `\tex_savepos:D`
 - This is more-or-less routine and actually documented, so no worries really here?
- `\mmzIncludeExtern`
 - Documented as internal, certainly not something to redefine, but maybe I can persuade Sašo to add the sockets I need here?

<*sty> <@@=mmzx>

523 \GetIdInfo \$Id: memoize-ext-tag.dtx 11956 2026-06-04 03:13:27Z cfrees \$ {Extensions


```

524 <!debug>    \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
525 <!debug>    {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
526 <debug>    \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
527 <debug>    {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
528 %
529 <!debug>    \disable@package@load {memoize-ext-tag-debug}
530 <debug>    \disable@package@load {memoize-ext-tag}
531 { Only one of memoize-ext-tag and memoize-ext-tag-debug
532  should be loaded.
533  Since
534 <!debug>    memoize-ext-tag
535 <debug>    memoize-ext-tag-debug
536  has been loaded,I will ignore your request for
537 <debug>    memoize-ext-tag
538 <!debug>    memoize-ext-tag-debug
539 .}

540 <!debug>    \RequirePackage{memoize-ext}
541 <debug>    \RequirePackage{memoize-ext-debug}
542 %

```

We don't want inconsistent names in hooks.

```

543 \SetDefaultHookLabel{memoize-ext}
544 %
545 \cs_generate_variant:Nn \socket_assign_plug:nn {nV}

```

```

\g_mmzx_tagpic_int (var.) Tracking & storage variables.
\l_mmzx_toks_tl (var.)
\l_mmzx_ok_bool (var.) 546 \bool_new:N \l_mmzx_ok_bool
g_mmzx_plug_orig_str (var.) 547 \int_new:N \g_mmzx_tagpic_int
\mmzxtagtoks (toks) 548 \str_new:N \g_mmzx_plug_orig_str
549 \tl_new:N \l_mmzx_toks_tl
550 \newtoks\mmzxtagtoks

```

\include/extern/before (socket) New sockets for extern utilisation.

```

\include/extern/after (socket)
551 \socket_new:nn {tagsupport/memoize/include/extern/before} {3}
552 \socket_new:nn {tagsupport/memoize/include/extern/after} {0}

```

Ulrike's pgf/tikz keys don't do anything with the arguments they receive? It only seems they do because `init` passes them also to the `l3keys`?

And so we have to pass them from `tikz` to `l3keys` and back to `tikz`

This creates a problem of synchronisation. It would be nice to use module-specific names to track `latex-lab` internal variables for ease of maintenance, but I don't think this is possible. If I let a new name to a token list variable, I'll just get the current value.

On the one hand, if users use `pgfkeys` to set the values for `alt` and `actualtext`, we can easily avoid `latex-lab` internals, at least. But we do that by introducing additional variables and then have the problem of synchronisation.

On the other hand, we could avoid the synchronisation problems by using `latex-lab` internals more consistently, but then even the `pgfkeys` interface will be dependent on the internal implementation¹.

¹I get the prohibition on internals. I really do. But there are cases where it just makes things much

Note: will need revising when the `pgfkeys` version of the `latex-lab` code gets published.

Keys can only belong to a single family, so I'm surprised it works still at all Given I wrote the `latex-lab` code I cannot really complain here.

```

553 \hook_gput_code:nnn {package/tikz/after} {.}
554 {
555   \pgfqkeys{/tikz}{
556     alt/.forward to=/mmz/alt,
557     actualtext/.forward to=/mmz/actualtext,
558     artifact/.forward to=/mmz/artifact,
559     tagging-setup/.forward to=/mmz/tagging-setup,
560   }
561   \mmzset{
562     alt/.code={
563       \pgfqkeys{/tikz/tagging-setup}{alt={#1}}
564       \bool_set_true:N \l__mmzx_ok_bool
565       \str_gset:Nn \g__mmzx_plug_orig_str {alt}
566       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
567       \socket_assign_plug:nn
568         {tagsupport/memoize/include/extern/before} {mmzx/alt}
569       \socket_assign_plug:nn
570         {tagsupport/memoize/include/extern/after} {mmzx/alt}
571       <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
572     },
573     actualtext/.code={
574       \pgfqkeys{/tikz/tagging-setup}{actualtext={#1}}
575       \bool_set_true:N \l__mmzx_ok_bool
576       \str_gset:Nn \g__mmzx_plug_orig_str {actualtext}
577       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
578       \socket_assign_plug:nn
579         {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
580       \socket_assign_plug:nn
581         {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
582       <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
583     },
584     artifact/.code={
585       \pgfqkeys{/tikz/tagging-setup}{artifact}
586       \bool_set_true:N \l__mmzx_ok_bool
587       \str_gset:Nn \g__mmzx_plug_orig_str {artifact}
588       \mmzxtagtoks {}
589       \socket_assign_plug:nn
590         {tagsupport/memoize/include/extern/before} {mmzx/artifact}
591       \socket_assign_plug:nn
592         {tagsupport/memoize/include/extern/after} {mmzx/artifact}
593       <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
594     },

```

Originally, I used a `choice` key here because `latex-lab` used one. Now it doesn't, so I wonder if a simple `code` key would make more sense.

```

595   tagging-setup/.is choice,
596   tagging-setup/alt/.forward to=/mmz/alt,
597   tagging-setup/actualtext/.forward to=/mmz/actualtext,

```

more complicated and error-prone. And sometimes it just doesn't not seem worth the additional fragility that introduces, even at the cost of some additional fragility due to the use of internals.

```

598     tagging-setup/artifact/.forward to=/mmz/artifact,
599     tagging-setup/.unknown/.code =
600     {

```

I guess this still does something useful?

```

601     \exp_args:Nno
602     \pgfqkeys{/tikz/tagging-setup}{\pgfkeyscurrentname={#1}}
603   },
604   alt/.belongs to family=/tikz/tagging-setup,
605   actualtext/.belongs to family=/tikz/tagging-setup,
606   artifact/.belongs to family=/tikz/tagging-setup,
607   tagging-setup/.belongs to family=/tikz/tagging-setup,
608   tagging-setup/alt/.belongs to family=/tikz/tagging-setup,
609   tagging-setup/actualtext/.belongs to family=/tikz/tagging-setup,
610   tagging-setup/artifact/.belongs to family=/tikz/tagging-setup,
611 }
612 }

```

So it is possible to do without this, but it is *much* more straightforward to do with.

The basic idea is this:

- At the start of memoization, we add to the utilisation code which redefines the (internal) command `\mmzIncludeExtern`.
- In its place, we use simple wrapper around a copy of the original.
- The wrapper defines a socket before and after executing the original.

This lets us wrap utilisation of the extern in an appropriate tagging structure. At least, that's the idea.

It would be nice to assign the plug here just based on whichever plugs are installed, but it is too early, while `\mmzAtEndMemoization` is too late.

```

613 \appto\mmzAtBeginMemoization{
614   \gtoksapp\mmzCCMemo{
615     \let\mmzxIncludeExternOrig\mmzIncludeExtern
616     \let\mmzIncludeExtern\mmzxIncludeExtern
617   }
618 }

```

`_mmzx_noop:nNnnnnnnn (fn.) \mmzIncludeExtern` only seems to be defined during utilisation, so we set up a command `include_extern:nNnnnnnnn (fn.) \mmzxIncludeExternOrig` and set it to noop here, then redefine it locally when the `\mmzxIncludeExtern` extern is utilised. `\mmzIncludeExtern` is then redefined to `\mmzxIncludeExtern`, which `\mmzxIncludeExternOrig` wraps `\mmzxIncludeExternOrig` in a tagging structure².

Note this not only uses, but redefines an internal command which the manual says users should never need to even use

On the other hand, this is exactly the kind of thing `memoize` does to *other* packages' internals and, indeed, to the L^AT_EX Project's. Which is more than can be said to defend my use of their internals

²Note to self: check output of tests visually if you do not want documents to be inaccessible to that tiny group of people who rely on vision to read.

But does that lessen my guilt? :-)

```

619 \cs_new_protected_nopar:Npn
620   \__mmzx_noop:nNnnnnnnn #1#2#3#4#5#6#7#8#9 {}
621 \cs_new_protected_nopar:Npn \__mmzx_include_extern:nNnnnnnnn #1#2#3#4#5#6#7#8#9
622 {
623   \int_gincr:N \g__mmzx_tagpic_int
624   <debug>   \__mmzx_debug:n {Args: #1; #2; #3; #4; #5; #6; #7; #8; #9}
625   \socket_use:nnnn {tagsupport/memoize/include/extern/before}
626   {#3}{#4}{#5}
627   <debug>   \__mmzx_debug:n {Executing original inclusion macro.}
628   \mmzxIncludeExternOrig {#1}#2{#3}{#4}{#5}{#6}{#7}{#8}{#9}
629   <debug>   \__mmzx_debug:n {Closing tagging structures.}
630   \socket_use:n {tagsupport/memoize/include/extern/after}
631 }
632 \cs_new_eq:NN \mmzxIncludeExtern \__mmzx_include_extern:nNnnnnnnn
633 \cs_new_eq:NN \mmzxIncludeExternOrig \__mmzx_noop:nNnnnnnnn

```

extern/before mmzx/alt (*plug*) pgf/tikz addaswyd o latex-lab-testphase-tika.sty «<

```

634 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/alt}
635 {
636   <debug>   \__mmzx_debug:n {Executing plug mmzx/alt in socket
637   <debug>   tagsupport/memoize/include/extern/before.}
638   \mode_if_vertical:T
639   {
640     \if@inlabel
641       \mode_leave_vertical:
642     \else
643       \tag_socket_use:n {para/begin}
644     \fi
645   }
646   \tag_mc_end_push:
647   \tl_set:No \l__mmzx_toks_tl {\the\mmzxtagtoks}
648   \tag_struct_begin:n
649   {
650     tag=Figure,
651     alt=\l__mmzx_toks_tl,
652   }
653   \tag_mc_begin:n {tag=Figure}
654   \cs_new:cpe {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
655   {
656     \__mmzx_pgftikz_tag_bbox:ennn {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
657     {#1}{#2}{#3}
658   }
659   <debug>   \cs_log:c {mmzx@tag@tikz@mark@pos@\int_to_arabic:n
660   <debug>   {\g__mmzx_tagpic_int}}
661   \tag_struct_gput:ene
662   {\tag_get:n {struct_num}}
663   {attribute}
664   {
665     /O /Layout /BBox
666     [
667       \use:c
668       {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
669     ]

```

```

670 }
671 <debug>    \_mmzx_debug:e {Recording xpos,
672 <debug>      ypos of mmxc-id\int_to_arabic:n {\g_mmzx_tagpic_int}.}
673 \tex_savepos:D
674 \_mmzx_property_record_orig:ee
675 {mmzx-id\int_to_arabic:n {\g_mmzx_tagpic_int}}
676 {xpos,ypos}
677 \tex_savepos:D
678 }

```

»>

extern/after mmzx/alt (*plug*) pgf/tikz addaswyd o latex-lab-testphase-tika.sty bod yn onest, cafodd ei ddwyn o latex-lab yn hollol «<

```

679 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/alt}
680 {
681 <debug>    \_mmzx_debug:n {Executing plug mmzx/alt in socket
682 <debug>      tagsupport/memoize/include/extern/after.}
683 \tag_mc_end:
684 \tag_struct_end:
685 \tag_mc_begin_pop:n {}
686 }

```

»>

before mmzx/actualtext (*plug*) «< addaswyd o latex-lab-testphase-tika.sty

```

after mmzx/actualtext (plug)
687 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
688 {
689 <debug>    \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
690 <debug>      tagsupport/memoize/include/extern/before.}
691 \pgfqkeys{/tikz/tagging-setup}{actualtext/.expand once = {\the\mmzxtagtoks},}
692 \socket_use:n {tagsupport/tikz/picture/begin}
693 }
694 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
695 {
696 <debug>    \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
697 <debug>      tagsupport/memoize/include/extern/after.}
698 \socket_use:n {tagsupport/tikz/picture/end}
699 }

```

»>

/before mmzx/artifact (*plug*) «< addaswyd o latex-lab-testphase-tika.sty

```

n/after mmzx/artifact (plug)
700 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/artifact}
701 {
702 <debug>    \_mmzx_debug:n {Executing plug mmzx/artifact in socket
703 <debug>      tagsupport/memoize/include/extern/before.}
704 \pgfqkeys{/tikz/tagging-setup}{artifact,}
705 \socket_use:n {tagsupport/tikz/picture/begin}
706 }
707 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/artifact}
708 {
709 <debug>    \_mmzx_debug:n {Executing plug mmzx/artifact in socket

```

```

710 <debug>      tagsupport/memoize/include/extern/after.}
711 \socket_use:n {tagsupport/tikz/picture/end}
712 }

```

»>

_pgftikz_tag_bbox:nnnn (fn.) A memoize-friendly alternative to get the bounding box for the alt plug.

_pgftikz_tag_bbox:ennn (fn.)

```

713 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox:nnnn #1#2#3#4
714 {
715   \__mmzx_pgftikz_tag_bbox_aux:eennn
716   {
717     \mmzx_property_ref_orig:ee {#1}{xpos}
718   }
719   {
720     \mmzx_property_ref_orig:ee {#1}{ypos}
721   }
722   {#2}{#3}{#4}
723 }
724 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox:nnnn {ennn}

```

ikz_tag_bbox_aux:nnnnn (fn.) Auxiliary.

ikz_tag_bbox_aux:eennn (fn.)

```

725 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox_aux:nnnnn #1#2#3#4#5
726 {
727   \dim_to_decimal_in_bp:n {#1sp}
728   \c_space_tl
729   \dim_to_decimal_in_bp:n {#2sp-#5}
730   \c_space_tl
731   \dim_to_decimal_in_bp:n {#1sp+#3}
732   \c_space_tl
733   \dim_to_decimal_in_bp:n {#2sp+#4+#5}
734 }
735 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox_aux:nnnnn {eennn}

736 \hook_gput_code_with_args:nnn {cmd/tikz@picture/before} {mmzx}
737 {
738   \tag_if_active:T
739   {
740     <debug>      \__mmzx_debug:n {Executing mmzx code in hook cmd/tikz@picture/before.}
741     \socket_assign_plug:nn {tagsupport/tikz/picture/init} {mmzx}
742   }
743 }

```

»>

ikz/picture/init mmzx (plug) «< Originally, I made something much more complicated, which worked, but meh. Here the idea is that *if we are memoizing, we do not tag at all*. There's no real downside to this: a document which includes code memoized during the current run is not usable anyway and tagging the memoized code is pointless and inefficient. (Actually, so is executing the original code for use during the current run, which is, I believe, how it works. But that is not my fault.)

Instead, we tag *only the utilisation of memos*. We don't need to get the bounding box — memoize already does that. All we need do is get the position on the page during utilisation. Everything else is for free.

Note that this code uses multiple format/latex-lab internals. If the support for tikz used exclusively pgfkeys, this would be easily avoided: we could stick to the public interface for all but one of these usages. But because it supports also l3keys, I don't see a way to avoid depending on internal variables there, too. l3keys does not have a `.forward_to:n` or similar. There is `.inherit:n`, but that does not work at all in the same way. Filtering and family code (below) is copied from the code I suggested for latex-lab's TikZ support ([#2004](#)).

```

744 \socket_new_plug:nnn {tagsupport/tikz/picture/init} {mmzx}
745 {
746   <debug>    \_mmzx_debug:n {Executing plug mmzx in socket
747   <debug>      tagsupport/tikz/picture/init.}
748   \bool_set_false:N \l__mmzx_ok_bool
749   \legacy_if:nT {memoizing}
750   {
751     <debug>    \_mmzx_debug:n {Arg to tagsupport/tikz/picture/init is #1.}
752     \pgfkeyssavekeyfilterstateto\mmzx@temp@tagging@saved@filterstate
753     \pgfkeysinstallkeyfilter{/pgf/key filters/active families or no family}
754     {{/pgf/key filters/false}{/pgf/key filters/false}}
755     \pgfqkeysactivatesinglefamilyandfilteroptions{/tikz/tagging-setup}{/tikz}{#1}
756
757     <debug>    \_mmzx_debug:e { Restoring filter state: \exp_not:V
758     <debug>      \mmzx@temp@tagging@saved@filterstate }
759     \mmzx@temp@tagging@saved@filterstate
760     \bool_if:NF \l__mmzx_ok_bool
761     {
762       <debug>    \_mmzx_debug:n {
763       <debug>      No plug set for utilisation. Trying to match latex-lab plug.
764       <debug>    }

```

If the argument to the picture set the plug, it is already in the code hash. Otherwise, we add the value of the latex-lab plug to the context, using `\mmzContextExtra` and appending globally as we are memoizing by hypothesis.

```

765     \xtoksapp\mmzContextExtra {
766       tagging plug=\_mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}
767     }
768     \str_case_e:nn
769     {\_mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}}
770     {
771       {alt} {
772         \exp_args:Ne \mmzset{
773           alt = \exp_not:V \l__tikz_tagging_alt_tl,
774         }
775       <debug>    \_mmzx_debug:n {Using alt plug.}
776       }
777       {actualtext} {
778         \exp_args:Ne \mmzset{
779           actualtext = \exp_not:V \l__tikz_tagging_actualtext_tl,
780         }
781       <debug>    \_mmzx_debug:n {Using actualtext plug.}
782       }
783       {artifact} {
784         \mmzset{artifact,}
785       <debug>    \_mmzx_debug:n {Using artifact plug.}
786     }

```

```

787     {text} {
788         \bool_set_false:N \l__mmzx_ok_bool
789 <debug>         \__mmzx_debug:e {Found unsupported
790 <debug>         text
791 <debug>         {tagsupport/tikz/picture/begin} plug.}
792         \PackageWarning{memoize-ext}{Unsupported tag config for tikz picture.
793         Please use alt,actualtext,artifact or another supported plug.
794         Note that text (the latex-lab default) is NOT supported.
795         Aborting memoization and marking code unmemoizable.
796     }
797     \mmzUnmemoizable
798 }
799 }
800 }
801 }
802 \bool_if:NT \l__mmzx_ok_bool
803 {
804     %         \__mmzx_tag_socket_plug_record:
805 <debug>     \__mmzx_debug:n {Disabling tagging for current tikz picture during
806 <debug>     current run.}
807     \socket_assign_plug:nn {tagsupport/tikz/picture/begin} {noop}
808     \socket_assign_plug:nn {tagsupport/tikz/picture/end} {noop}
809     \socket_assign_plug:nn {tagsupport/tikz/picture/text/begin} {noop}
810     \socket_assign_plug:nn {tagsupport/tikz/picture/text/end} {noop}
811 }
812 }

813 \hook_gput_code:nnn {cmd/mmzAbort/after} {.}
814 {
815     \legacy_if:nT {memoizing} {
816         \PackageWarning{memoize-ext}{
817             Memoization has been aborted. If something like a reference needs
818             resolving, just compile again. If the content is unmemoizable --
819             e.g. contains remember picture, a breakable tcolorbox or suchlike,
820             you must mark the content unmemoizable or the tagging will be
821             incorrect. You may do this automatically or manually. Aborted
822         }
823     }
824 }

»>

```

socket_plug_record:nnn (fn.) Wrappers for the most common cases of recording plugs for use in utilisation. «<

_socket_plug_record:nn (fn.)

_socket_plug_record:ee (fn.)

ag_socket_plug_record: (fn.)

```

825 \cs_new_protected:Npn \__mmzx_tag_socket_plug_record:nn #1#2
826 {
827     \xtoksapp\mmzCCMemo{
828         \exp_not:N \mmzxtagtoks
829         =
830         \c_left_brace_str
831         \the\mmzxtagtoks
832         \c_right_brace_str
833         \exp_not:N \AssignSocketPlug
834         \c_left_brace_str
835         tagsupport/memoize/include/extern/before
836         \c_right_brace_str

```



```

837 \c_left_brace_str
838 #1
839 \c_right_brace_str
840 \exp_not:N \AssignSocketPlug
841 \c_left_brace_str
842 tagsupport/memoize/include/extern/after
843 \c_right_brace_str
844 \c_left_brace_str
845 #2
846 \c_right_brace_str
847 }
848 }
849 \cs_generate_variant:Nn \_mmzx_tag_socket_plug_record:nn {ee}
850 \cs_new_protected:Npn \_mmzx_tag_socket_plug_record:
851 {
852 \_mmzx_tag_socket_plug_record:ee
853 {\_mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/before}}
854 {\_mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/after}}
855 }
856 \cs_new_protected:Npn \_mmzx_tag_socket_plug_record:nnn #1#2#3
857 {
858 \mmzxtagtoks { \text_purify:n {#3} }
859 \_mmzx_tag_socket_plug_record:nn {#1} {#2}
860 }

»>

```

`\mmzx_tag_get_recorded:N` (*fn.*) Intended for use in plugs.

#1 should be a local token list variable. «<

```

861 \cs_new_protected_nopar:Npn \mmzx_tag_get_recorded:N #1
862 {
863 \tl_set:No #1 {\the\mmzxtagtoks}
864 }

»>

```

`\socket_plug_record:nnn` (*fn.*) Public names for recording and auto-recording plugs. Note `\mmzxtagtoks` is available here and is auto-recorded regardless. «<

`_socket_plug_record:nn` (*fn.*) here and is auto-recorded regardless. «<

```

865 \cs_new_eq:NN \mmzx_tag_socket_plug_record:nnn \_mmzx_tag_socket_plug_record:nnn
866 \cs_new_eq:NN \mmzx_tag_socket_plug_record:nn \_mmzx_tag_socket_plug_record:nn
867 \cs_new_eq:NN \mmzx_tag_socket_plug_record: \_mmzx_tag_socket_plug_record:

```

»> pgf/tikz addaswyd o latex-lab-testphase-tika.sty

Hook rules to ensure our substitutes override the format's. «<

```

868 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{latex-lab-testphase-tikz}{>}{.}
869 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tagpdf}{>}{.}
870 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tikz}{>}{.}
871 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{latex-lab-testphase-tikz}{>}{.}
872 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tikz}{>}{.}
873 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tagpdf}{>}{.}
874 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {latex-lab-testphase-tikz}
875 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tagpdf}
876 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tikz}

```

```

877 <debug>          \hook_log:n {package/tikz/after}
878 \hook_gput_code:nnn {begindocument/end} {.}
879 {
880 <debug>          \hook_log:n {cmd/tikz@picture/before}
881 <debug>          \hook_log:n {cmd/endpgfpicture/after}

```

»>

x_property_ref_orig:nn (*fn.*) «< Avoid dependency on memoize-ext-properties.

```

882 \cs_if_free:NT \mmzx_property_ref_orig:nn
883 {
884   \cs_new_eq:NN \mmzx_property_ref_orig:nn \property_ref:nn
885   \cs_generate_variant:Nn \mmzx_property_ref_orig:nn {ee}
886 }

```

»>

property_record_orig:nn (*fn.*) «< Avoid dependency on memoize-ext-properties.

```

887 \cs_if_free:NT \__mmzx_property_record_orig:nn
888 {
889   \cs_new_eq:NN \__mmzx_property_record_orig:nn \property_record:nn
890   \cs_generate_variant:Nn \__mmzx_property_record_orig:nn {ee}
891 }

```

»>

892 }

</sty>

memoize-ext-talk

Clea F. Rees

11956 2026-05-26

Abstract

memoize-ext-talk enables memoization of ltx-talk presentations (Wright 2026). The package is part of memoize-ext.

memoize-ext-talk is essentially a ‘translation’ of memoize’s support for beamer, together with modifications for differences in the way the classes implement overlays and changes to the implementation of opacity when pdfmanagement-testphase (L^AT_EX Project 2026) is loaded.

The package tries to avoid utilising the internals of either memoize or ltx-talk, but it was not entirely possible to do so without making the code both more fragile and unduly convoluted. See the main manual for memoize-ext for details.

The user interface is identical to that provided by memoize for beamer (Živanović 2024).

Contents

<*sty> <@@=mmzx>

```
893 \GetIdInfo $Id: memoize-ext-talk.dtx 11956 2026-06-04 03:13:27Z cfrees $ {Extension
894 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
895 \!debug} {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
896 \!debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
897 \!debug} {v0.4.2 \ExplFileVersion}{\ExplFileDescription}
898 %
899 \!debug} \disable@package@load {memoize-ext-talk-debug}
900 \!debug} \disable@package@load {memoize-ext-talk}
901 { Only one of memoize-ext-talk and memoize-ext-talk-debug
902 should be loaded.
903 Since
904 \!debug} memoize-ext-talk
905 \!debug} memoize-ext-talk-debug
906 has been loaded,I will ignore your request for
907 \!debug} memoize-ext-talk
908 \!debug} memoize-ext-talk-debug
909 .}
```

We have to load this (I think) prior to \documentclass, so cannot use tag_if_active:TF here. We could test for \DocumentMetadata, but ltx-talk already requires that. But maybe I should be testing that anyway?

```
910 \!debug} \RequirePackage{memoize-ext-tag}
911 \!debug} \RequirePackage{memoize-ext-tag-debug}
```

We don't want inconsistent names in hooks.

```
912 \SetDefaultHookLabel{memoize-ext}
```

BEGIN ltx-talk addaswyd o memoize-beamer.code.tex & other memoize code

```
913 \hook_gput_code:nnn {class/ltx-talk/after}{.}
914 {
915   <debug>      \__mmzx_debug:n {Enabling ltx-talk support.}
916   \mmzset{
917     per overlay/.code={},
918     talk mode to prefix/.style={
919       prefix=\mmz@prefix@dir\mmz@prefix@name\l__mmzx_talk_mode_str -,
920     },
921   }
```

```
mmzx_talk_opacity_seq (var.)
talk_saved_opacity_seq (var.)
__mmzx_talk_opacity_tl (var.) 922 \seq_new:N \g__mmzx_talk_opacity_seq
923 \seq_new:N \g__mmzx_talk_saved_opacity_seq
924 \tl_new:N \l__mmzx_talk_opacity_tl

\g__mmzx_talk_frame_int
\g__mmzx_talk_slide_int
\g__mmzx_talk_pauses_int 925
\l__mmzx_talk_mode_str 926 \cs_new_eq:NN \g__mmzx_talk_frame_int \c@frame
927 \cs_new_eq:NN \g__mmzx_talk_slide_int \c@slide
928 \cs_new_eq:NN \g__mmzx_talk_pauses_int \c@pauses
929 \cs_new_eq:NN \l__mmzx_talk_mode_str \l__talk_mode_str
```

\opacity_select:V (fn.) ltx-talk does this too late (at least, I get an error)

```
930 \cs_generate_variant:Nn \opacity_select:n {V}
```

Initialise sequence.

```
931 \seq_gpush:Nn \g__mmzx_talk_opacity_seq {1}
```

mmzx_talk_opacity_save: (fn.)

```
932 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_save:
933 {
934   \seq_gset_eq:NN \g__mmzx_talk_opacity_saved_seq \g__mmzx_talk_opacity_seq
935   \seq_get:NN \g__mmzx_talk_opacity_seq \l__mmzx_talk_opacity_tl
936   \exp_args:NV \tl_if_eq:NNTF \l__mmzx_talk_opacity_tl \q_no_value
937   {
938     \seq_gpush:Nn \g__mmzx_talk_opacity_saved_seq {1}
939     \opacity_select:n {1}
940   } {
941     \seq_gpush:NV \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
942     \opacity_select:V \l__mmzx_talk_opacity_tl
943   }
944 }
```

_talk_opacity_restore: (fn.)

```

945 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_restore:
946 {
947   \seq_gpop:NN \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
948   \opacity_select:V \l__mmzx_talk_opacity_tl
949   \seq_gset_eq:NN \g__mmzx_talk_opacity_seq \g__mmzx_talk_opacity_saved_seq
950 }

```

`\mmzSingleExternDriver` Redefine driver Even if this does not have an internal name, the redefinition uses internals in spades We could define a new one & I guess that's what should be done ...

```

951 \long\def\mmzSingleExternDriver#1{
952   \xtoksapp\mmzCCMemo{\mmz@maybe@quitvmode}
953   \setbox\mmz@box\mmz@capture{
954     \__mmzx_talk_opacity_save:
955     #1
956     \__mmzx_talk_opacity_restore:
957   }
958   \mmzExternalizeBox\mmz@box\mmz@temptoks
959   \xtoksapp\mmzCCMemo{\the\mmz@temptoks}
960   \mmz@maybe@quitvmode\box\mmz@box
961 }

```

The code below is iffy, I guess, since the begin/end thing here is rather illusory ...

```

962 \hook_gset_rule:nnnn {begindocument} {ltx-talk} {<} {..}
963 \hook_gput_code_with_args:nnn {cmd/opacity_select:n/before} {..}
964 {
965   \seq_gpush:Nn \g__mmzx_talk_opacity_seq {#1}
966 }
967 \hook_gput_code:nnn {cmd/opacity_end:/after}{..}
968 {
969 <debug>   \__mmzx_debug:n{0opacity after}
970   \seq_gpop:NN \g__mmzx_talk_opacity_seq \l_tmpa_tl
971 <debug>   \seq_log:N \g__mmzx_talk_opacity_seq
972 }
973 \mmzset{
974   at begin memoization={
975     \socket_use:n {mmzx/talk/memoization/begin}
976   },
977   at end memoization={
978     \socket_use:n {mmzx/talk/memoization/end}
979   },
980   per overlay/.style={
981     /mmz/context={
982       overlay=\csname theslide\endcsname,
983       pauses=\ifmemoizing
984         \mmzxTalkPauses
985       \else
986         \expandafter\the\csname c@pauses\endcsname
987       \fi
988     },
989     /utils/exec={
990       \str_if_eq:eeF {peroverlay}
991       {\__mmzx_socket_assigned_plug:n {mmzx/talk/memoization/begin}}

```

```

992      {
993        \socket_assign_plug:nn {mmzx/talk/memoization/begin}{peroverlay}
994        \socket_assign_plug:nn {mmzx/talk/memoization/end}{peroverlay}

```

This is required to allow per overlay to be used in the options for tikzpictures etc.

```

995        \legacy_if:nT {memoizing} {
996          \socket_use:n {mmzx/talk/memoization/begin}
997        }
998      }
999    },

```

Is resetting the style meant to prevent duplication in memos etc.? Because it obviously won't ...?

```

1000    /mmz/per overlay/.code={},
1001  },
1002 }

```

k/memoization/begin (*socket*) Sockets.

alk/memoization/end (*socket*)

```

1003 \socket_new:nn {mmzx/talk/memoization/begin}{0}
1004 \socket_new:nn {mmzx/talk/memoization/end}{0}

```

tion/begin peroverlay (*plug*) Plugs.

zation/end peroverlay (*plug*)

```

1005 \socket_new_plug:nnn {mmzx/talk/memoization/begin}{peroverlay}
1006 {
1007 <debug>   \_mmzx_debug:n {Executing plug peroverlay in socket
1008 <debug>     mmzx/talk/memoization/begin.}
1009   \xdef\mmzxTalkPauses{
1010     \int_to_arabic:n {\g__mmzx_talk_pauses_int}
1011   }
1012   \xtoksapp\mmzCMemo{
1013     \noexpand\mmzxSetTalkOverlays{\mmzxTalkPauses}{
1014       \int_to_arabic:n {\g__mmzx_talk_slide_int}
1015     }
1016   }
1017   \gtoksapp\mmzCCMemo{
1018     \only<all:\mmzxTalkOverlays>{}
1019   }
1020   \seq_get:NNTF \g__mmzx_talk_opacity_seq \l__mmzx_opacity_tl
1021   {
1022     \fp_gset:N \l__mmzx_tmpa_fp \l__mmzx_opacity_tl
1023   } {
1024     \fp_gset:Nn \l__mmzx_tmpa_fp {1}
1025   }
1026   \xtoksapp\mmzContextExtra{
1027     opacity=\fp_to_decimal:N \l__mmzx_tmpa_fp
1028   }
1029 }
1030 \socket_new_plug:nnn {mmzx/talk/memoization/end}{peroverlay}
1031 {
1032 <debug>   \_mmzx_debug:n {Executing plug peroverlay in socket
1033 <debug>     mmzx/talk/memoization/end.}
1034   \xtoksapp\mmzCCMemo{
1035     \exp_not:N \setcounter{pauses}

```

```

1036      {\int_to_arabic:n {\g__mmzx_talk_pauses_int}}
1037    }
1038  }

```

`\mmzxSetTalkOverlays` Note the addition of code to set `g__talk_slide_continue_bool`. This isn't necessary for `beamer` and is not 'allowed' for `ltx-talk`, but is necessary for frames with overlay specifications in memoized content.

```

1039  \cs_new_nopar:Npn \mmzxSetTalkOverlays#1#2{
1040  <debug>    \__mmzx_debug:e {Executing \cs_to_str:N \mmzxSetTalkOverlays}
1041    \int_compare:nNnTF {\g__mmzx_talk_pauses_int} = {#1}
1042    {
1043      \gdef\mmzxTalkOverlays{#2}
1044      \int_compare:nNnTF {\g__mmzx_talk_slide_int} < {#2}
1045      {
1046        \bool_set_true:N \l__mmzx_tmpa_bool
1047      }{
1048        \bool_set_false:N \l__mmzx_tmpa_bool
1049      }
1050    }{
1051      \bool_set_true:N \l__mmzx_tmpa_bool
1052    }
1053    \bool_if:NT \l__mmzx_tmpa_bool
1054    {
1055      \bool_gset_true:N \g__talk_slide_continue_bool
1056      \appto\mmzAtBeginMemoization{
1057        \gtoksapp\mmzCMemo{\mmzxSetTalkOverlays{#1}{#2}}
1058      }
1059    }
1060  }

1061 }

</sty>

```

References

- L^AT_EX Project (2025a). *The l3draw Package: Core Drawing Support*. 2025-10-09. 9th Oct. 2025. CTAN: [l3experimental](#).
 — (2025b). *The latex-lab-tikz Package: Support for the Tagging of TikZ Pictures*. v0.80d. 27th Sept. 2025. CTAN: [latex-lab](#).
 — (2026). *The L^AT_EX PDF Management Bundle*. 0.96y. 23rd Jan. 2026. CTAN: [pdfmanagement-testphase](#).
 Rees, Clea F. (2026). *forest-ext: A Collection of forest Libraries*. 0.2. 20th Feb. 2026. CTAN: [forest-ext](#).
 Wright, Joseph (2026). *ltx-talk: A Class for Typesetting Presentations*. 0.4.4. 10th Feb. 2026. CTAN: [ltx-talk](#).
 Živanović, Sašo (2017). *Forest: A PGF/TikZ-Based Package for Drawing Linguistic Trees*. 2.1.5. 14th July 2017. CTAN: [forest](#).
 — (2024). *Memoize*. 1.4.1. 2nd Dec. 2024. CTAN: [memoize](#).

Change History

| | | | |
|--|----|--------|---|
| vo.2 | | | |
| General: See <code>init</code> . | 25 | | <code>_mmzx_tag_socket_plug_record::</code> Add fns. |
| <code>tagssupport/tikz/picture/init_mmzx</code> : Avoid processing non-tagging keys too early and too often. Save and restore existing filter state. | 31 | | <code>_mmzx_tag_socket_plug_record:nnn</code> , <code>_mmzx_tag_socket_plug_record:nn</code> , <code>_mmzx_tag_socket_plug_record:ee</code> and <code>_mmzx_tag_socket_plug_record:.</code> 32 |
| vo.3 | | | <code>\g_mmzx_draw_id_begin_int</code> : Save and compare l3draw id. 21 |
| General: Add tagging status to salt. | 13 | | <code>\mmzx_tag_get_recorded:N</code> : New expl3 fn. to get recorded text. 33 |
| Correct and update usage details for <code>ltx-talk</code> . | 3 | | <code>\mmzx_tag_socket_plug_record::</code> Add fns. |
| Load <code>memoize-ext-tag/memoize-ext-tag-debug</code> and hope <code>\DocumentMetadata</code> is sufficient in case tagging is off. | 35 | | <code>\mmzx_tag_socket_plug_record:nn</code> and <code>\mmzx_tag_socket_plug_record:.</code> 33 |
| <code>mmzx/talk/memoization/end</code> : Use sockets to try to keep memos cleaner. | 38 | vo.3.5 | <code>\@mmzx@kernel@after@shipout@background</code> : Fix for GitHub sasozivanovic/memoize/issues/60 .issue #60. 11 |
| <code>mmzx/talk/memoization/end_peroverlay</code> : What is a socket without a plug? | 38 | | |
| vo.3.1 | | | |
| <code>tagssupport/tikz/picture/init_mmzx</code> : Default to OK. | 31 | vo.3.6 | <code>\@mmzx@kernel@after@shipout@background</code> : Fix fix for Fix for GitHub sasozivanovic/memoize/issues/60 .issue #60. 11 |
| vo.3.2 | | | Try to be a bit more cautious, even on pdfL ^A T _E X. 11 |
| General: Fix <code>\toksapp</code> and friends courtesy Max Chernoff. | 11 | | Use ‘less internal’ internal <code>2e</code> macro, which does not need metadata test. 11 |
| Modified <code>\xtoksapp</code> means all engines can use the same code here. | 17 | | |
| vo.3.3 | | | |
| <code>_mmzx_noop:n</code> : Need this defined earlier. I’m starting to understand why libraries are a thing. | 12 | vo.4 | General: Use <code>pgfkeys</code> interface now it’s available, which is a little bit nicer. 27 |
| <code>\mmzx@expl@replicate@fn</code> : Switch cat codes if necessary. | 20 | | Use new <code>/tikz/tagging-setup</code> rather than <code>/mmzx/tagging@setup</code> ; switch to <code>pgfkeys</code> interface. 26 |
| vo.3.4 | | | <code>_mmzx_noop:n</code> : Fix <code>_mmzx_noop:n</code> , though currently unused. 12 |
| General: Hopefully slightly saner code for incrementing l3draw id. | 22 | | |
| Replicate changes to l3draw’s id. | 22 | | |

`\l_mmzx_opt_expl_bool`: Fix for GitHub
[latex3/latex2e/issues/2110](https://github.com/latex3/latex2e/issues/2110). L^AT_EX 2_ε issue
 #2110. 10

Fix non-recognition of memoize pkg options
 in `memoize-ext-debug`. 9

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

| | | | |
|---|---|--|------------------------------|
| <code>\currname</code> | 64 | <code>_mmzx_property_record_orig:ee</code> | 674 |
| <code>\@ifl@t@r</code> | 5, 80 | <code>_mmzx_property_record_orig:nn</code> (fn.) . . . | 887 |
| <code>\@ifundefined</code> | 2 | <code>_mmzx_restore_ccmemo_input</code> : (fn.) | 259, 278, 296 |
| <code>\@mmzx@kernel@after@shipout@background</code> . | 109 | <code>_mmzx_saved_mmzxExplAtBegin</code> : (fn.) | 259, 284, 294 |
| <code>_keys_options_expand_module:Nn</code> | 67 | <code>_mmzx_saved_mmzxExplAtEnd</code> : (fn.) | 259, 285, 295 |
| <code>_mmzx_advice_collect_draw_args:w</code> (fn.) . . | 498, 505 | <code>_mmzx_socket_assigned_plug:n</code> (fn.) | 519, 766, 769, 853, 854, 991 |
| <code>_mmzx_cctab_end</code> : (fn.) | 300 | <code>_mmzx_tag_socket_plug_record</code> : (fn.) | 804, 825, 867 |
| <code>_mmzx_cctab_stop</code> : (fn.) | 301, 345, 458 | <code>_mmzx_tag_socket_plug_record:ee</code> (fn.) . | 825 |
| <code>_mmzx_debug:N</code> | 138 | <code>_mmzx_tag_socket_plug_record:nn</code> (fn.) . . | 825, 866 |
| <code>_mmzx_debug:e</code> | 140, 671, 757, 789, 1040 | <code>_mmzx_tag_socket_plug_record:nnn</code> (fn.) . . | 825, 865 |
| <code>_mmzx_debug:n</code> | 133, 137, 176, 185, 226, 229, 237, 268, 325, 331, 335, 624, 627, 629, 636, 681, 689, 696, 702, 709, 740, 746, 751, 762, 775, 781, 785, 805, 915, 969, 1007, 1032 | <code>_mmzx_talk_opacity_restore</code> : (fn.) . | 945, 956 |
| <code>_mmzx_expl_at_begin</code> : (fn.) | 301 | <code>_mmzx_talk_opacity_save</code> : (fn.) . . . | 932, 954 |
| <code>_mmzx_expl_at_start</code> : (fn.) | 286, 301 | <code>_mmzx_tmp_cctab_stop</code> : (fn.) | 394, 442 |
| <code>_mmzx_expl_replicate_aux:n</code> (fn.) | 377 | | |
| <code>_mmzx_expl_replicate_b</code> : (fn.) | 409, 410, 411, 413, 414, 416, 418, 425 | | |
| <code>_mmzx_expl_replicate_e</code> : (fn.) | 420, 422, 425 | | |
| <code>_mmzx_expl_replicate_t</code> : (fn.) | 417, 419, 425 | | |
| <code>_mmzx_expl_replicate_fn</code> : (fn.) | 442 | | |
| <code>_mmzx_expl_replicate_fn_aux:nnN</code> (fn.) . . | 377, 465 | | |
| <code>_mmzx_if_replicating</code> : | 223 | | |
| <code>_mmzx_if_replicating:F</code> | 235 | | |
| <code>_mmzx_if_replicating:TF</code> (fn.) | 223 | | |
| <code>_mmzx_if_replicating_p</code> : (fn.) | 223 | | |
| <code>_mmzx_include_extern:nNnnnnnnn</code> (fn.) . . | 619 | | |
| <code>_mmzx_nexpl_at_begin</code> : (fn.) | 301 | | |
| <code>_mmzx_nexpl_at_start</code> : (fn.) | 301, 449 | | |
| <code>_mmzx_noop</code> : | 143, 442, 461 | | |
| <code>_mmzx_noop:n</code> | 143 | | |
| <code>_mmzx_noop:nNnnnnnnn</code> (fn.) | 619 | | |
| <code>_mmzx_pgftikz_tag_bbox:ennn</code> (fn.) . . | 656, 713 | | |
| <code>_mmzx_pgftikz_tag_bbox:nnnn</code> (fn.) | 713 | | |
| <code>_mmzx_pgftikz_tag_bbox_aux:ennn</code> (fn.) . . | 715, 725 | | |
| <code>_mmzx_pgftikz_tag_bbox_aux:nnnn</code> (fn.) . | 725 | | |

A

| | |
|---|----------------|
| <code>\AdviceCollectDrawArguments</code> | 503, 516 |
| <code>\AdviceInnerHandler</code> | 501 |
| <code>\AdviceOriginal</code> | 392 |
| <code>\AdviceReplaced</code> | 466 |
| <code>\AdviceRunIfNotReplicating</code> | 233, 243 |
| <code>\AdviceRuntrue</code> | 238 |
| <code>\appto</code> | 324, 613, 1056 |
| <code>\AssignSocketPlug</code> | 833, 840 |
| <code>auto/run if not replicating</code> (pgfkey) | 241 |

B

| | |
|--|---|
| <code>\bool_gset_true:N</code> | 1055 |
| <code>\bool_if:NF</code> | 272, 760 |
| <code>\bool_if:NT</code> | 145, 166, 173, 182, 802, 1053 |
| <code>\bool_new:N</code> | 37, 122, 221, 258, 546 |
| <code>\bool_set_false:N</code> | 40, 222, 264, 269, 293, 445, 748, 788, 1048 |
| <code>\bool_set_true:N</code> | 39, 274, 448, 464, 564, 575, 586, 1046, 1051 |
| <code>\box</code> | 960 |

C

| | |
|---------------------------------|-----|
| <code>\c@frame</code> | 926 |
|---------------------------------|-----|

| | | | |
|--|---|--|--|
| <code>\c@pauses</code> | 928 | <code>\else:</code> | 228 |
| <code>\c@slide</code> | 927 | <code>\endcsname</code> | 329, 339, 451, 456, 982, 986 |
| <code>\c__mmzx_expl_at_cctab</code> | 246, 303 | <code>\endinput</code> | 13 |
| <code>\c__mmzx_nexpl_at_cctab</code> | 250, 307 | <code>\etoksapp</code> | 101 |
| <code>\c_code_cctab</code> | 247, 251 | <code>\exp_after:wN</code> | 390 |
| <code>\c_left_brace_str</code> | 373, 830, 834, 837, 841, 844 | <code>\exp_args:Ne</code> | 566, 577, 772, 778 |
| <code>\c_right_brace_str</code> | 375, 832, 836, 839, 843, 846 | <code>\exp_args:Nno</code> | 601 |
| <code>\c_space_tl</code> | 728, 730, 732 | <code>\exp_args:No</code> | 331, 501, 571, 582, 593 |
| <code>\cctab_begin:N</code> | 303, 307 | <code>\exp_args:NV</code> | 70, 140, 466, 936 |
| <code>\cctab_const:Nn</code> | 246, 250 | <code>\exp_last_unbraced:Ne</code> | 465 |
| <code>\cctab_end:</code> | 300 | <code>\exp_not:c</code> | 71 |
| <code>\cctab_select:N</code> | 247, 251 | <code>\exp_not:N</code> | 70, 72, 329, 339, 390, 392, 393, 451, 456, 828, 833, 840, 1035 |
| <code>\char_set_catcode_active:n</code> | 256 | <code>\exp_not:n</code> | 140, 571, 582, 593 |
| <code>\char_set_catcode_space:n</code> | 254, 255 | <code>\exp_not:V</code> | 757, 773, 779 |
| <code>\clist_map_inline:nn</code> | 101 | <code>\expandafter</code> | 105, 986 |
| <code>\cs:w</code> | 390 | <code>\ExpandArgs</code> | 91 |
| <code>\cs_end:</code> | 390 | <code>\expandonce</code> | 392 |
| <code>\cs_generate_variant:Nn</code> | 137, 545, 724, 735, 849, 885, 890, 930 | <code>expl3 (opt.)</code> | 3 |
| <code>\cs_gset_protected:ce</code> | 387 | <code>expl3 functions:</code> | |
| <code>\cs_if_exist:cF</code> | 379, 385 | <code>__mmzx_advice_collect_draw_args:w</code> | 498, 505 |
| <code>\cs_if_exist_use:c</code> | 92 | <code>__mmzx_cctab_end:</code> | 300 |
| <code>\cs_if_free:NT</code> | 882, 887 | <code>__mmzx_cctab_stop:</code> | 301, 345, 458 |
| <code>\cs_log:c</code> | 659 | <code>__mmzx_expl_at_begin:</code> | 301 |
| <code>\cs_log:N</code> | 333, 342 | <code>__mmzx_expl_at_start:</code> | 286, 301 |
| <code>\cs_new:cpe</code> | 654 | <code>__mmzx_expl_replicate_aux:n</code> | 377 |
| <code>\cs_new:Npn</code> | 143, 233, 377, 404, 443, 498, 503 | <code>__mmzx_expl_replicate_b:</code> | 409, 410, 411, 413, 414, 416, 418, 425 |
| <code>\cs_new_eq:cc</code> | 399 | <code>__mmzx_expl_replicate_e:</code> | 420, 422, 425 |
| <code>\cs_new_eq:NN</code> | 144, 300, 442, 468, 632, 633, 865, 866, 867, 884, 889, 926, 927, 928, 929 | <code>__mmzx_expl_replicate_t:</code> | 417, 419, 425 |
| <code>\cs_new_nopar:Npn</code> | 309, 314, 425, 432, 438, 519, 713, 725, 1039 | <code>__mmzx_expl_replicate_fn:</code> | 442 |
| <code>\cs_new_protected:Npn</code> | 133, 138, 825, 850, 856 | <code>__mmzx_expl_replicate_fn_aux:nnN</code> | 377, 465 |
| <code>\cs_new_protected_nopar:Npn</code> | 261, 301, 305, 319, 619, 621, 861, 932, 945 | <code>__mmzx_if_replicating:TF</code> | 223 |
| <code>\cs_set:Npn</code> | 453 | <code>__mmzx_if_replicating_p:</code> | 223 |
| <code>\cs_set_eq:NN</code> | 284, 285, 294, 295, 461 | <code>__mmzx_include_extern:nNnnnnnnn</code> | 619 |
| <code>\cs_set_nopar:Npn</code> | 311, 312, 316, 317, 321, 322 | <code>__mmzx_nexpl_at_begin:</code> | 301 |
| <code>\cs_set_protected:cpn</code> | 66 | <code>__mmzx_nexpl_at_start:</code> | 301, 449 |
| <code>\cs_set_protected_nopar:Npn</code> | 278 | <code>__mmzx_noop:nNnnnnnnnn</code> | 619 |
| <code>\cs_split_function:N</code> | 466 | <code>__mmzx_pgftikz_tag_bbox:ennn</code> | 656, 713 |
| <code>\cs_to_str:N</code> | 140, 1040 | <code>__mmzx_pgftikz_tag_bbox:nnnn</code> | 713 |
| <code>\csname</code> | 329, 339, 451, 456, 982, 986 | <code>__mmzx_pgftikz_tag_bbox_aux:eennn</code> | 715, 725 |
| <code>\CurrentOption</code> | 78 | <code>__mmzx_pgftikz_tag_bbox_aux:nnnnn</code> | 725 |
| D | | | |
| <code>\DeclareDocumentCommand</code> | 64 | <code>__mmzx_property_record_orig:nn</code> | 887 |
| <code>\DeclareUnknownKeyHandler</code> | 64, 77 | <code>__mmzx_restore_ccmemo_input:</code> | 259, 278, 296 |
| <code>\def</code> | 105, 951 | <code>__mmzx_saved_mmzxExplAtBegin:</code> | 259, 284, 294 |
| <code>\dim_to_decimal_in_bp:n</code> | 727, 729, 731, 733 | <code>__mmzx_saved_mmzxExplAtEnd:</code> | 259, 285, 295 |
| <code>\disable@package@load</code> | 25, 26, 206, 207, 353, 354, 481, 482, 529, 530, 899, 900 | <code>__mmzx_socket_assigned_plug:n</code> | 519, 766, 769, 853, 854, 991 |
| <code>\draw_end:</code> | 498, 500 | <code>__mmzx_tag_socket_plug_record:</code> | 804, 825, 867 |
| E | | | |
| <code>\else</code> | 277, 642, 985 | <code>__mmzx_tag_socket_plug_record:ee</code> | 825 |

| | | | |
|--------------------------------------|--|----------|--|
| _mmzx_tag_socket_plug_record:nn | 825, 866 | | |
| _mmzx_tag_socket_plug_record:nnn | ... | | |
| | 825, 865 | | |
| _mmzx_talk_opacity_restore: | .. 945, 956 | | |
| _mmzx_talk_opacity_save: | 932, 954 | | |
| _mmzx_tmp_cctab_stop: | 394, 442 | | |
| \mmzx_property_ref_orig:nn | 882 | | |
| \mmzx_tag_get_recorded:N | 8, 861 | | |
| \mmzx_tag_socket_plug_record: | 8, 865 | | |
| \mmzx_tag_socket_plug_record:nn | .. 7, 865 | | |
| \mmzx_tag_socket_plug_record:nnn | .. 7, 865 | | |
| \opacity_select:V | 930, 942, 948 | | |
| expl3 variables: | | | |
| \g_mmzx_draw_id_begin_int | 497 | | |
| \g_mmzx_expl_replicate__ba_tl | .. 370, 430 | | |
| \g_mmzx_expl_replicate__bb_tl | .. 370, 428 | | |
| \g_mmzx_expl_replicate__tb_tl | .. 370, 435 | | |
| \g_mmzx_plug_orig_str | .. 546, 565, 576, 587 | | |
| \g_mmzx_tagpic_int | | | |
| | 546, 623, 654, 656, 660, 668, 672, 675 | | |
| \g_mmzx_talk_opacity_seq | 922, | | |
| | 931, 934, 935, 949, 965, 970, 971, 1020 | | |
| \g_mmzx_talk_saved_opacity_seq | 922 | | |
| \l_mmzx_expl_bool | | | |
| | 258, 264, 269, 272, 274, 293 | | |
| \l_mmzx_ok_bool | | | |
| | 546, 564, 575, 586, 748, 760, 788, 802 | | |
| \l_mmzx_opt_draw_bool | 41, 173 | | |
| \l_mmzx_opt_expl_bool | 41, 166 | | |
| \l_mmzx_opt_tag_bool | 37, 56, 145 | | |
| \l_mmzx_replicating_bool | ... 221, 225, 464 | | |
| \l_mmzx_talk_opacity_tl | | | |
| | 922, 935, 936, 941, 942, 947, 948 | | |
| \l_mmzx_toks_tl | 546, 647, 651 | | |
| \ExplFileDate | 17, 20, 201, | | |
| | 203, 348, 350, 476, 478, 524, 526, 894, 896 | | |
| \ExplFileDescription | 18, 20, 202, | | |
| | 204, 349, 351, 477, 479, 525, 527, 895, 897 | | |
| \ExplFileName | 17, 19, 23, 201, | | |
| | 203, 348, 350, 476, 478, 524, 526, 894, 896 | | |
| \ExplFileVersion | 18, 20, 202, | | |
| | 204, 349, 351, 477, 479, 525, 527, 895, 897 | | |
| \ExplLoaderFileDate | 5 | | |
| F | | | |
| \fi | 238, 282, 644, 987 | | |
| \fi: | 231 | | |
| \FirstAidNeededT | 113 | | |
| \fmtversion | 80 | | |
| \fp_gset:Nn | 1024 | | |
| \fp_gset:Nv | 1022 | | |
| \fp_new:N | 123 | | |
| \fp_to_decimal:N | 1027 | | |
| | | G | |
| \g_mmzx_draw_id_begin_int (var) | 497 | | |
| \g_mmzx_expl_replicate__ba_tl (var) | 370, 430 | | |
| \g_mmzx_expl_replicate__bb_tl (var) | .. 370, 428 | | |
| \g_mmzx_expl_replicate__tb_tl (var) | .. 370, 435 | | |
| \g_mmzx_name_str | | | |
| | 22, 23, 147, 148, 168, 169, 175, 177 | | |
| \g_mmzx_plug_orig_str (var) | 546, 565, 576, 587 | | |
| \g_mmzx_tagpic_int (var) | | | |
| | 546, 623, 654, 656, 660, 668, 672, 675 | | |
| \g_mmzx_talk_frame_int | 925 | | |
| \g_mmzx_talk_opacity_saved_seq | | | |
| | 934, 938, 941, 947, 949 | | |
| \g_mmzx_talk_opacity_seq (var) | 922, | | |
| | 931, 934, 935, 949, 965, 970, 971, 1020 | | |
| \g_mmzx_talk_pauses_int | 925, 1010, 1036, 1041 | | |
| \g_mmzx_talk_saved_opacity_seq (var) | ... 922 | | |
| \g_mmzx_talk_slide_int | 925, 1014, 1044 | | |
| \g_talk_slide_continue_bool | 1055 | | |
| \gdef | 1043 | | |
| \GetIdInfo | 16, 200, 347, 475, 523, 893 | | |
| \group_begin: | 463 | | |
| \group_end: | 393 | | |
| \gtoksapp | 101, 510, 614, 1017, 1057 | | |
| | | H | |
| \hook_gput_code:nnn | | | |
| | 115, 116, 149, 151, 171, 180, 262, 266, | | |
| | 270, 289, 291, 343, 553, 813, 878, 913, 967 | | |
| \hook_gput_code_with_args:nnn | 736, 963 | | |
| \hook_gset_rule:nnnn | 868, | | |
| | 869, 870, 871, 872, 873, 874, 875, 876, 962 | | |
| \hook_log:n | 877, 880, 881 | | |
| | | I | |
| \if@inlabel | 640 | | |
| \if_bool:N | 225 | | |
| \IfFormatAtLeastF | 63 | | |
| \IfFormatAtLeastTF | 62, 80, 81, 88 | | |
| \ifmemoizing | 238, 983 | | |
| \ifmmz@direct@ccmemo@input | 275 | | |
| \IfPackageLoadedF | 153, 155 | | |
| \int_compare:nNnTF | 1041, 1044 | | |
| \int_gincr:N | 623 | | |
| \int_incr:N | 406 | | |
| \int_new:N | 124, 497, 547 | | |
| \int_set:Nn | 253 | | |
| \int_to_arabic:n | 429, 436, | | |
| | 654, 656, 659, 668, 672, 675, 1010, 1014, 1036 | | |
| \int_zero:N | 381 | | |
| \iow_log:n | 135 | | |
| | | K | |
| \keys_define:ne | 67 | | |
| \keys_define:nn | 41 | | |

L

l3draw (opt.) 3
 \l__mmzx_expl_bool (var)
 258, 264, 269, 272, 274, 293
 \l__mmzx_ok_bool (var)
 546, 564, 575, 586, 748, 760, 788, 802
 \l__mmzx_opacity_tl 1020, 1022
 \l__mmzx_opt_draw_bool (var) 41, 173
 \l__mmzx_opt_expl_bool (var) 41, 166
 \l__mmzx_opt_tag_bool (var) 37, 56, 145
 \l__mmzx_opt_talk_bool 58, 182
 \l__mmzx_replicating_bool (var) . 221, 225, 464
 \l__mmzx_talk_mode_str 919, 925
 \l__mmzx_talk_opacity_tl (var)
 922, 935, 936, 941, 942, 947, 948
 \l__mmzx_tmpa_bool
 122, 445, 448, 1046, 1048, 1051, 1053
 \l__mmzx_tmpa_fp 123, 1022, 1024, 1027
 \l__mmzx_tmpa_int 124, 381, 406, 429, 436
 \l__mmzx_tmpa_seq 129
 \l__mmzx_tmpa_str 130
 \l__mmzx_tmpa_tl
 126, 382, 390, 392, 428, 429, 430, 435, 436
 \l__mmzx_tmpb_str 131
 \l__mmzx_tmpb_tl 127
 \l__mmzx_tmpc_tl
 128, 383, 385, 387, 397, 399, 427, 434
 \l__mmzx_toks_tl (var) 546, 647, 651
 \l__talk_mode_str 929
 \l__tikz_tagging_actualtext_tl 779
 \l__tikz_tagging_alt_tl 773
 \l__keys_key_str 72
 \l__tmpa_tl 970
 \legacy_if:nT 749, 815, 995
 \let 615, 616
 \long 951

M

\makeatletter 248, 252
 \MessageBreak 10
 mmz/auto/replicate expl fn (pgfkey) 469
 mmz/auto/replicate expl var (pgfkey) 469
 \mmz@box 953, 958, 960
 \mmz@capture 953
 \mmz@direct@ccmemo@inputfalse 280
 \mmz@direct@ccmemo@inputtrue 283
 \mmz@maybe@quitvmode 952, 960
 \mmz@prefix@dir 919
 \mmz@prefix@name 919
 \mmz@temptoks 958, 959
 \mmzAtBeginMemoization 324, 326, 333, 613, 1056
 \mmzAtEndMemoization 334, 336, 342
 \mmzCCMemo 327, 331, 337, 389, 450, 455, 510,
 571, 582, 593, 614, 827, 952, 959, 1017, 1034
 \mmzCMemo 1012, 1057

\mmzContextExtra 765, 1026
 \mmzExternalizeBox 958
 \mmzIncludeExtern 615, 616
 \mmzSalt 189
 \mmzset 95,
 241, 469, 507, 561, 772, 778, 784, 916, 973
 \mmzSingleExternDriver 951
 \mmzUnmemoizable 797
 mmzx (plug) 6
 mmzx/actualtext (plug) 6
 mmzx/alt (plug) 6
 mmzx/artifact (plug) 6
 mmzx/talk/memoization/begin (socket) ... 1003
 mmzx/talk/memoization/begin peroverlay
 (plug) 1005
 mmzx/talk/memoization/end (socket) 1003
 mmzx/talk/memoization/end peroverlay
 (plug) 1005
 \mmzx@expl@replicate@fn 442, 472
 \mmzx@temp@tagging@saved@filterstate
 752, 758, 759
 \mmzx_property_ref_orig:ee 717, 720
 \mmzx_property_ref_orig:nn (fn.) 882
 \mmzx_tag_get_recorded:N (fn.) 8, 861
 \mmzx_tag_socket_plug_record: (fn.) ... 8, 865
 \mmzx_tag_socket_plug_record:nn (fn.) . 7, 865
 \mmzx_tag_socket_plug_record:nnn (fn.) 7, 865
 \mmzxExplAtBegin 284, 294, 311, 316, 321, 329, 451
 \mmzxExplAtEnd
 285, 295, 312, 317, 322, 339, 446, 456
 \mmzxIncludeExtern 616, 619
 \mmzxIncludeExternOrig 615, 619
 \mmzxSetTalkOverlays 1013, 1039
 \mmzxtagtoks (toks) 546,
 566, 577, 588, 647, 691, 828, 831, 858, 863
 \mmzxTalkOverlays 1018, 1043
 \mmzxTalkPauses 984, 1009, 1013
 \mode_if_vertical:T 638
 \mode_leave_vertical: 641
 \msg_line_context: 195
 \msg_new:nnnn 192
 \msg_warning:nnnnnn 157
 \msg_warning_text:n 194

N

\NeedsTeXFormat 1
 \newtoks 550
 \noexpand 117, 1013

O

\only 1018
 \opacity_select:n 930, 939
 \opacity_select:V (fn.) 930, 942, 948
 options:
 expl3 3

| | | | |
|---|---|--|---|
| l3draw | 3 | | |
| tag | 3 | | |
| talk | 3 | | |
| P | | Q | |
| \PackageError | 7, 440 | \q_mmzx_stop | 125 |
| \PackageWarning | 45, 792, 816 | \q_no_value | 936 |
| \PassOptionsToPackage | 78 | \quark_new:N | 125 |
| per overlay (pgfkey) | 4 | | |
| pgfkeys: | | R | |
| auto/run if not replicating | 241 | \relax | 276 |
| mmz/auto/replicate expl fn | 469 | replicate expl fn (pgfkey) | 4 |
| mmz/auto/replicate expl var | 469 | \RequirePackage . . . | 3, 85, 90, 94, 147, 148, 168, 169, 175, 177, 184, 186, 217, 218, 364, 365, 366, 367, 493, 494, 495, 496, 540, 541, 910, 911 |
| per overlay | 4 | | |
| replicate expl fn | 4 | S | |
| talk mode to prefix | 4 | \seq_get:NN | 935 |
| \pgfkeyscurrentname | 602 | \seq_get:NNTF | 1020 |
| \pgfkeysinstallkeyfilter | 753 | \seq_gpop:NN | 947, 970 |
| \pgfkeyssavekeyfilterstateto | 752 | \seq_gpush:Nn | 931, 938, 965 |
| \pgfqkeys | 555, 563, 574, 585, 602, 691, 704 | \seq_gpush:NV | 941 |
| \pgfqkeysactivatesinglefamilyandfilteroptions | | \seq_gset_eq:NN | 934, 949 |
| | 755 | \seq_log:N | 971 |
| plugins: | | \seq_new:N | 129, 922, 923 |
| mmzx | 6 | \setbox | 953 |
| mmzx/actualtext | 6 | \setcounter | 1035 |
| mmzx/alt | 6 | \SetDefaultHookLabel ... | 36, 220, 369, 543, 912 |
| mmzx/artifact | 6 | \socket_assign_plug:nn | |
| mmzx/talk/memoization/begin peroverlay | | | 545, 567, 569, 578, 580, 589, 591, 741, 807, 808, 809, 810, 993, 994 |
| | 1005 | \socket_new:nn | 551, 552, 1003, 1004 |
| mmzx/talk/memoization/end peroverlay | 1005 | \socket_new_plug:nnn | 634, 679, 687, 694, 700, 707, 744, 1005, 1030 |
| tagsupport/memoize/include/extern/after | | \socket_use:n | |
| mmzx/actualtext | 687 | | 630, 692, 698, 705, 711, 975, 978, 996 |
| tagsupport/memoize/include/extern/after | | \socket_use:nnnn | 625 |
| mmzx/alt | 679 | sockets: | |
| tagsupport/memoize/include/extern/after | | mmzx/talk/memoization/begin | 1003 |
| mmzx/artifact | 700 | mmzx/talk/memoization/end | 1003 |
| tagsupport/memoize/include/extern/before | | tagsupport/memoize/include/extern/after | |
| mmzx/actualtext | 687 | | 7, 551 |
| tagsupport/memoize/include/extern/before | | tagsupport/memoize/include/extern/before | |
| mmzx/alt | 634 | | 7, 551 |
| tagsupport/memoize/include/extern/before | | \str_case:nnF | 407 |
| mmzx/artifact | 700 | \str_case_e:nn | 768 |
| tagsupport/tikz/picture/init mmzx | 744 | \str_gset:Nn | 565, 576, 587 |
| \preto | 334 | \str_gset:NV | 23 |
| \prg_new_conditional:Npnn | 223 | \str_if_eq:eeF | 397, 990 |
| \prg_return_false: | 230 | \str_if_eq:eeT | 446 |
| \prg_return_true: | 227 | \str_new:N | 22, 130, 131, 548 |
| \ProcessKeyOptions | 83 | \str_use:c | 521 |
| \ProcessKeysOptions | 86 | \string | 326, 336 |
| \property_record:nn | 889 | \sys_if_engine luatex:F | 99 |
| \property_ref:nn | 884 | \sys_if_engine pdftex:T | 114 |
| \protected | 103, 105 | | |
| \providecommand | 80, 91 | T | |
| \ProvidesExplPackage | 17, 19, 201, 203, 348, 350, 476, 478, 524, 526, 894, 896 | tag (opt.) | 3 |
| | | \tag_get:n | 662 |
| | | \tag_if_active:T | 738 |

[illegible]