

memoize-ext: Extensions for memoize

Clea F. Rees*

11744 2026-01-19

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
II Implementation	7
memoize-ext	7
memoize-ext-expl3	12
memoize-ext-l3draw	19
memoize-ext-sockets	21

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

memoize-ext-tag	22
memoize-ext-talk	32

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~csmname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
  auto~csmname={__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 `LATEX` 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 `lsockets`.

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 `tagsupport/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 `tagsupport/tikz/picture/begin` and `tagsupport/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?

tagsupport/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.

tagsupport/memoize/include/extern/after

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

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 }{%
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 11744 2026-03-06 21:12:42Z cfrees $ {Extensions for
17 <!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
18 <!debug> {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
```

⁶This follows memoize’s own practice.

```

19 <debug> \ProvidesExplPackage{\ExplFileName-debug}
20 <debug> {\ExplFileDate}{v0.3.1 \ExplFileVersion}{\ExplFileDescription}
21 %
22 \str_new:N \g__mmzx_name_str
23 \str_gset:N \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 }
62 \DeclareUnknownKeyHandler{
63 \PassOptionsToPackage{\CurrentOption}{memoize}
64 }

```

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

64327823#64327823

```

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

66 \IfFormatAtLeastTF { 2022-06-01 }
67 {
68   \ProcessKeyOptions [ memoize-ext ]
69 }{
70   \RequirePackage { l3keys2e }
71   \ProcessKeysOptions { memoize-ext }
72 }

73 \IfFormatAtLeastTF { 2020-10-01 }{
74 }{
75   \RequirePackage { xparse }
76   \providecommand \ExpandArgs [1]
77   { \cs_if_exist_use:c { exp_args:N #1 } }
78 }

```

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

```

79 \RequirePackage{memoize}
80 <debug>   \mmzset{
81 <debug>     trace,
82 <debug>     include context in ccmemo,
83 <debug>   }

```

temporary variables, quarks

```

84 \bool_new:N \l__mmzx_tmpa_bool
85 \fp_new:N \l__mmzx_tmpa_fp
86 \int_new:N \l__mmzx_tmpa_int
87 \quark_new:N \q__mmzx_stop
88 \tl_new:N \l__mmzx_tmpa_tl
89 \tl_new:N \l__mmzx_tmpb_tl
90 \tl_new:N \l__mmzx_tmpc_tl
91 \seq_new:N \l__mmzx_tmpa_seq
92 \str_new:N \l__mmzx_tmpa_str
93 \str_new:N \l__mmzx_tmpb_str
94 <*debug>
95 \cs_new_protected:Npn \__mmzx_debug:n #1
96 {
97   \iow_log:n {[mmzx debug]:: #1}
98 }
99 \cs_generate_variant:Nn \__mmzx_debug:n {e}
100 \cs_new_protected:Npn \__mmzx_debug:N #1
101 {
102   \__mmzx_debug:e {\cs_to_str:N #1: \exp_args:NV \exp_not:n #1}
103 }
104 </debug>

```

tag, expl, l3draw, talk loaded conditionally

```

105 \bool_if:NT \l__mmzx_opt_tag_bool
106 {
107 <!debug>   \RequirePackage{g__mmzx_name_str -tag}

```

```

108 <debug>      \RequirePackage{\g__mmzx_name_str -tag-debug}
109 \hook_gput_code:nnn {package/forest/after}{.}
110 {
111   \hook_gput_code:nnn {begindocument/before}{.}
112   {
113     \IfPackageLoadedF {forest-lib-ext.tagging}
114     {
115       \IfPackageLoadedF {forest-lib-ext.tagging-debug}
116       {
117         \msg_warning:nnnnnn {memoize-ext}{unsupported}{forest}
118         {forest-lib-ext.tagging.sty}{forest-ext}
119         {forest trees will not be correctly tagged and may cause fatal
120          compilation errors.}
121       }
122     }
123   }
124 }
125 }

```

memoize-ext-expl3[-debug]

```

126 \bool_if:NT \l__mmzx_opt_expl_bool
127 {
128 <!debug>      \RequirePackage{\g__mmzx_name_str -expl3}
129 <debug>       \RequirePackage{\g__mmzx_name_str -expl3-debug}
130 }

```

memoize-ext-l3draw[-debug]

```

131 \hook_gput_code:nnn {package/l3draw/after}{.}
132 {
133   \bool_if:NT \l__mmzx_opt_draw_bool
134   {
135 <!debug>      \RequirePackage {\g__mmzx_name_str -l3draw}
136 <debug>       \__mmzx_debug:n {Loading memoize-ext-l3draw-debug.}
137 <debug>       \RequirePackage {\g__mmzx_name_str -l3draw-debug}
138   }
139 }

```

memoize-ext-talk[-debug]

```

140 \hook_gput_code:nnn {class/ltx-talk/after} {.}
141 {
142   \bool_if:NT \l__mmzx_opt_talk_bool
143   {
144 <!debug>      \RequirePackage{memoize-ext-talk}
145 <debug>       \__mmzx_debug:n {Loading memoize-ext-talk-debug.}
146 <debug>       \RequirePackage{memoize-ext-talk-debug}
147   }
148 }

```

__mmzx_noop: Do nothing successfully.

```

\__mmzx_noop:n
149 \cs_new:Npn \__mmzx_noop: {}
150 \cs_new:Npn \__mmzx_noop:n {}

```

NaCl | halen | salt

```
151 \toksapp\mmzSalt{
152   Tagging status: \tag_if_active_p:
153 }

messages

154 \msg_new:nnnn {memoize-ext}{unsupported}
155 {
156   \msg_warning_text:n {memoize-ext}:
157   Non-existent or inappropriate version of #2 from #3 \msg_line_context:.
158   #4
159 } {
160   memoize-ext#1 requires an appropriate version of #2 from #3.
161 }

</sty>
```

memoize-ext-expl3

Clea F. Rees

11744 2026-01-19

Abstract

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

Contents

```
<@@=mmzx> <*common>
```

```
162 \GetIdInfo $Id: memoize-ext-expl3.dtx 11744 2026-03-06 21:12:42Z cfrees $ {Extension
163 (!debug) \ProvidesExplPackage{\ExplFileName-common}{\ExplFileDate}{v0.0 %
164 (!debug) \ExplFileVersion}{\ExplFileDescription}
165 (debug) \ProvidesExplPackage{\ExplFileName-common-debug}{\ExplFileDate}{v0.0 %
166 (debug) \ExplFileVersion}{\ExplFileDescription}
167 %
168 (!debug) \disable@package@load {memoize-ext-expl3-common-debug}
169 (debug) \disable@package@load {memoize-ext-expl3-common}
170 { Only one of memoize-ext-expl3-common and memoize-ext-expl3-common-debug
171 should be loaded.
172 Since
173 (!debug) memoize-ext-expl3-common
174 (debug) memoize-ext-expl3-common-debug
175 has been loaded,I will ignore your request for
176 (debug) memoize-ext-expl3-common
177 (!debug) memoize-ext-expl3-common-debug
178 .}

179 (!debug) \RequirePackage{memoize-ext}
180 (debug) \RequirePackage{memoize-ext-debug}
181 %
```

We don't want inconsistent names in hooks.

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

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

```
183 \bool_new:N \l__mmzx_replicating_bool
184 \bool_set_false:N \l__mmzx_replicating_bool
```

```

mmzx_if_replicating:TF (fn.)
mmzx_if_replicating_p: (fn.)
185 \prg_new_conditional:Npnn \_mmzx_if_replicating: {p,T,TF,F}
186 {
187   \if_bool:N \l_mmzx_replicating_bool
188   <debug> \_mmzx_debug:n {Replicating true.}
189   \prg_return_true:
190   \else:
191   <debug> \_mmzx_debug:n {Replicating false.}
192   \prg_return_false:
193   \fi:
194 }

```

`\AdviceRunIfNotReplicating` Run condition.

```

195 \cs_new:Npn \AdviceRunIfNotReplicating
196 {
197   \_mmzx_if_replicating:F
198   {
199   <debug> \_mmzx_debug:n {Not replicating,so proceeding.}
200   \ifmemoizing \AdviceRuntrue \fi
201   }
202 }

```

`if not replicating` (*pgfkey*) Style.

```

203 \mmzset{
204   auto/run if not replicating/.style = {
205     run conditions={\AdviceRunIfNotReplicating},
206   },
207 }

```

consts

```

208 \cctab_const:Nn \c_mmzx_expl_at_cctab {
209   \cctab_select:N \c_code_cctab
210   \makeatletter
211 }
212 \cctab_const:Nn \c_mmzx_nexpl_at_cctab {
213   \cctab_select:N \c_code_cctab
214   \makeatletter
215   \int_set:Nn \tex_endlinechar:D { 13 }
216   \char_set_catcode_space:n { 9 }
217   \char_set_catcode_space:n { 32 }
218   \char_set_catcode_active:n { 126 } % tilde
219 }

```

expl3, memoizing `côd` `ynddo`

hooks instead of [TeX 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.

```

220 \bool_new:N \l_mmzx_expl_bool

```

```

_restore_ccmemo_input: (fn.) Initialise.
_saved_mmzxExplAtBegin: (fn.)
x_saved_mmzxExplAtEnd: (fn.)
221 \cs_new_protected_nopar:Npn \__mmzx_restore_ccmemo_input: {}
222 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtBegin: {}
223 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtEnd: {}

```

Auto-switching for expl3 syntax.

```

224 \hook_gput_code:nnn {begindocument/end}{.}
225 {
226   \bool_set_false:N \l__mmzx_expl_bool
227 }

228 \hook_gput_code:nnn {begindocument/end}{.}
229 {
230 (debug)   \__mmzx_debug:n {Tracking expl3 syntax changes.}
231   \bool_set_false:N \l__mmzx_expl_bool
232   \hook_gput_code:nnn {cmd/ExplSyntaxOn/before} { . }
233   {
234     \bool_if:NF \l__mmzx_expl_bool
235     {
236       \bool_set_true:N \l__mmzx_expl_bool
237       \ifmmz@direct@ccmemo@input
238         \relax
239       \else
240         \cs_set_protected_nopar:Npn \__mmzx_restore_ccmemo_input:
241         {
242           \mmz@direct@ccmemo@inputfalse
243         }
244       \fi
245       \mmz@direct@ccmemo@inputtrue
246       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtBegin: \mmzxExplAtBegin
247       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtEnd: \mmzxExplAtEnd
248       \__mmzx_expl_at_start:
249     }
250   }

```

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

```

251 \hook_gput_code:nnn {cmd/ExplSyntaxOn/after} { . }
252 {
253   \hook_gput_code:nnn {cmd/ExplSyntaxOff/before} { . }
254   {
255     \bool_set_false:N \l__mmzx_expl_bool
256     \cs_set_eq:NN \mmzxExplAtBegin \__mmzx_saved_mmzxExplAtBegin:
257     \cs_set_eq:NN \mmzxExplAtEnd \__mmzx_saved_mmzxExplAtEnd:
258     \__mmzx_restore_ccmemo_input:
259   }
260 }
261 }

```

fns mewnlol

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

```

262 \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.) 263 \cs_new_protected_nopar:Npn \__mmzx_expl_at_begin:
\__mmzx_nexpl_at_start: (fn.) 264 {
\__mmzx_cctab_stop: (fn.) 265 \cctab_begin:N \c__mmzx_expl_at_cctab
266 }
267 \cs_new_protected_nopar:Npn \__mmzx_nexpl_at_begin:
268 {
269 \cctab_begin:N \c__mmzx_nexpl_at_cctab
270 }
271 \cs_new_nopar:Npn \__mmzx_expl_at_start:
272 {
273 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_expl_at_begin:}
274 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
275 }
276 \cs_new_nopar:Npn \__mmzx_nexpl_at_start:
277 {
278 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_nexpl_at_begin:}
279 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
280 }
281 \cs_new_nopar:Npn \__mmzx_cctab_stop:
282 {
283 \cs_set_nopar:Npn \mmzxExplAtBegin {relax}
284 \cs_set_nopar:Npn \mmzxExplAtEnd {relax}
285 }

```

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.

```

286 \bool_lazy_or:nnTF
287 {
288 \sys_if_engine_pdftex_p:
289 } {
290 \sys_if_engine_xetex_p:
291 } {
292 \appto\mmzAtBeginMemoization{
293 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
294 <debug> \string\mmzAtBeginMemoization.}
295 <debug> \__mmzx_debug:n {Working around pdfTeX/XeTeX peculiarity
296 <debug> which eats first noexpand.}
297 \xtoksapp\mmzCCMemo
298 {
299 \exp_not:N \exp_not:N
300 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \exp_not:N \exp_not:N \endcsname
301 }
302 <debug> \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
303 }
304 } {
305 \appto\mmzAtBeginMemoization{
306 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
307 <debug> \string\mmzAtBeginMemoization.}
308 <debug> \__mmzx_debug:n {LuaTeX doesn't eat first noexpand.}
309 \xtoksapp\mmzCCMemo
310 {
311 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname

```

```

312     }
313 <debug>     \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
314   }
315 }
316 \preto\mmzAtEndMemoization{
317 <debug>     \__mmzx_debug:n {Appending expl end to ccmemo at start
318 <debug>     \string\mmzAtEndMemoization.}
319 \xtoksapp\mmzCCMemo
320 {
321   \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
322 }
323 }

```

init

```

324 \hook_gput_code:nnn { begindocument/end } {mmzx}
325 {
326 % % % % % % % \__mmzx_cctab_stop:
327 % }

```

</common>

<*sty>

```

328 \GetIdInfo $Id: memoize-ext-expl3.dtx 11744 2026-03-06 21:12:42Z cfrees $ {Extension
329 <!debug>     \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
330 <!debug>     {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
331 <debug>     \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
332 <debug>     {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
333 %
334 <!debug>     \disable@package@load {memoize-ext-expl3-debug}
335 <debug>     \disable@package@load {memoize-ext-expl3}
336 { Only one of memoize-ext-expl3 and memoize-ext-expl3-debug
337   should be loaded.
338   Since
339 <!debug>     memoize-ext-expl3
340 <debug>     memoize-ext-expl3-debug
341   has been loaded,I will ignore your request for
342 <debug>     memoize-ext-expl3
343 <!debug>     memoize-ext-expl3-debug
344 .}

345 <!debug>     \RequirePackage{memoize-ext}
346 <debug>     \RequirePackage{memoize-ext-debug}
347 <!debug>     \RequirePackage{memoize-ext-expl3-common}
348 <debug>     \RequirePackage{memoize-ext-expl3-common-debug}
349 %

```

We don't want inconsistent names in hooks.

```

350 \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.)

```

351 \tl_new:N \g__mmzx_expl_replicate__bb_tl
352 \tl_new:N \g__mmzx_expl_replicate__ba_tl
353 \tl_new:N \g__mmzx_expl_replicate__tb_tl
354 \tl_gput_right:NV \g__mmzx_expl_replicate__bb_tl \c_left_brace_str
355 \tl_gput_right:Nn \g__mmzx_expl_replicate__bb_tl {#}
356 \tl_gput_right:NV \g__mmzx_expl_replicate__ba_tl \c_right_brace_str
357 \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 `\l_expl_replicate__aux:n` (*fn.*) second delegates details according to the argument specification.

```

358 \cs_new:Npn \__mmzx_expl_replicate_fn_aux:nnN #1#2#3
359 {
360   \cs_if_exist:cF { __mmzx_rep_#1:#2 }
361   {
362     \int_zero:N \l__mmzx_tmpa_int
363     \tl_clear:N \l__mmzx_tmpa_tl
364     \tl_clear:N \l__mmzx_tmpc_tl
365     \tl_map_function:nN {#2} \__mmzx_expl_replicate__aux:n
366     \cs_if_exist:cF { __mmzx_rep_#1:\l__mmzx_tmpc_tl }
367     {
368       \cs_gset_protected:ce {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
369       {
370         \xtoksapp\mmzCCMemo{
371           \exp_after:wN \exp_not:N \cs:w #1:#2 \cs_end: \l__mmzx_tmpa_tl
372         }
373         \exp_not:N \expandonce \exp_not:N \AdviceOriginal \l__mmzx_tmpa_tl
374         \exp_not:N \group_end:
375       }
376     }
377     \str_if_eq:eeF {#2}{\l__mmzx_tmpc_tl}
378     { % ych!
379       \cs_new_eq:cc {__mmzx_rep_#1:#2} {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
380     }
381   }
382   \use:c { __mmzx_rep_#1:#2 }
383 }
384 \cs_new:Npn \__mmzx_expl_replicate__aux:n #1
385 {
386   \int_incr:N \l__mmzx_tmpa_int
387   \str_case:nnF { #1 }
388   {
389     {c} { \__mmzx_expl_replicate__b: }
390     {e} { \__mmzx_expl_replicate__b: }
391     {o} { \__mmzx_expl_replicate__b: }
392     {p} { }
393     {n} { \__mmzx_expl_replicate__b: }
394     {v} { \__mmzx_expl_replicate__b: }
395     {D} { }
396     {F} { \__mmzx_expl_replicate__b: }
397     {N} { \__mmzx_expl_replicate__t: }
398     {T} { \__mmzx_expl_replicate__b: }
399     {V} { \__mmzx_expl_replicate__t: }
400     {w} { \__mmzx_expl_replicate__e: }
401   }{
402     \__mmzx_expl_replicate__e:

```

```

403 }
404 }

```

`\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.

```

405 \cs_new_nopar:Npn \__mmzx_expl_replicate__b:
406 {
407   \tl_put_right:Nn \l__mmzx_tmpc_tl {n}
408   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__bb_tl
409   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
410   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__ba_tl
411 }
412 \cs_new_nopar:Npn \__mmzx_expl_replicate__t:
413 {
414   \tl_put_right:Nn \l__mmzx_tmpc_tl {N}
415   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__tb_tl
416   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
417 }
418 \cs_new_nopar:Npn \__mmzx_expl_replicate__e:
419 {
420   \PackageError{mmzx}{No do,sorry. Use replicate with args instead.}{}
421 }

```

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

```

\mmzx@expl@replicate@fn
422 \cs_new:Npn \__mmzx_expl_replicate_fn:
423 {
424   \group_begin:
425     \bool_set_true:N \l__mmzx_replicating_bool
426     \exp_last_unbraced:Ne \__mmzx_expl_replicate_fn_aux:nnN
427       { \exp_args:NV \cs_split_function:N \AdviceReplaced }
428 }
429 \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 `/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.

```

430 \mmzset{% config <<<
431   auto/replicate expl fn/.style={
432     run if not replicating,
433     outer handler=\mmzx@expl@replicate@fn,
434   },
435 }% >>>

```

</sty>

memoize-ext-l3draw

Clea F. Rees

11744 2026-01-19

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>

```
436 \GetIdInfo $Id: memoize-ext-l3draw.dtx 11744 2026-03-06 21:12:42Z cfrees $ {Extensi
437 \!debug) \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
438 \!debug) {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
439 \!debug) \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
440 \!debug) {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
441 %
442 \!debug) \disable@package@load {memoize-ext-l3draw-debug}
443 \!debug) \disable@package@load {memoize-ext-l3draw}
444 { Only one of memoize-ext-l3draw and memoize-ext-l3draw-debug
445 should be loaded.
446 Since
447 \!debug) memoize-ext-l3draw
448 \!debug) memoize-ext-l3draw-debug
449 has been loaded,I will ignore your request for
450 \!debug) memoize-ext-l3draw
451 \!debug) memoize-ext-l3draw-debug
452 .}
453 %
454 \!debug) \RequirePackage{memoize-ext}
455 \!debug) \RequirePackage{memoize-ext-debug}
456 \!debug) \RequirePackage{memoize-ext-expl3}
457 \!debug) \RequirePackage{memoize-ext-expl3-debug}
```

l3draw

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.

```
458 \cs_new:Npn \__mmzx_advice_collect_draw_args:w #1 \draw_end:
459 {
460 \toks0={ #1 \draw_end: }
461 \exp_args:No \AdviceInnerHandler {\the\toks0}
462 }
```

`AdviceCollectDrawArguments` Public wrapper.

```
463 \cs_new:Npn \AdviceCollectDrawArguments{
464   \toks0 = {}
465   \__mmzx_advice_collect_draw_args:w
466 }
```

Default configuration.

```
467 \mmzset{%
468   auto csname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
469   auto csname={_draw_record_origin:}{run if memoizing,replicate expl fn,},
470 }
```

</sty>

memoize-ext-sockets

Clea F. Rees

11744 2026-01-19

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.

```
471 \cs_new_nopar:Npn \__mmzx_socket_assigned_plug:n #1
472 { % rhybudd: fn mewno1
473   \str_use:c { l__socket_#1_plug_str }
474 }
```

</sty>

memoize-ext-tag

Clea F. Rees

11744 2026-01-19

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_actualext_text_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>

475 \GetIdInfo \$Id: memoize-ext-tag.dtx 11744 2026-03-06 21:12:42Z cfrees \$ {Extensions

```

476 <!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
477 <!debug> {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
478 <debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
479 <debug> {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
480 %
481 <!debug> \disable@package@load {memoize-ext-tag-debug}
482 <debug> \disable@package@load {memoize-ext-tag}
483 { Only one of memoize-ext-tag and memoize-ext-tag-debug
484 should be loaded.
485 Since
486 <!debug> memoize-ext-tag
487 <debug> memoize-ext-tag-debug
488 has been loaded,I will ignore your request for
489 <debug> memoize-ext-tag
490 <!debug> memoize-ext-tag-debug
491 .}

492 <!debug> \RequirePackage{memoize-ext}
493 <debug> \RequirePackage{memoize-ext-debug}
494 %

```

We don't want inconsistent names in hooks.

```

495 \SetDefaultHookLabel{memoize-ext}
496 %
497 \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.)
\g__mmzx_plug_orig_str (var.)
  \mmzxtagtoks (toks)
501 \tl_new:N \l__mmzx_toks_tl
502 \newtoks\mmzxtagtoks

```

```

\include/extern/before (socket) New sockets for extern utilisation.
\include/extern/after (socket)
503 \socket_new:nn {tagsupport/memoize/include/extern/before} {3}
504 \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.

```

505 \hook_gput_code:nnn {package/tikz/after} {.}
506 {
507   \pgfqkeys{/tikz}{
508     alt/.forward to=/mmz/alt,
509     actualtext/.forward to=/mmz/actualtext,
510     artifact/.forward to=/mmz/artifact,
511     tagging-setup/.forward to=/mmz/tagging-setup,
512     /mmzx/tagging@setup/.is family,
513     alt/.belongs to family=/mmzx/tagging@setup,
514     artifact/.belongs to family=/mmzx/tagging@setup,
515     actualtext/.belongs to family=/mmzx/tagging@setup,
516     tagging-setup/.belongs to family=/mmzx/tagging@setup,
517   }
518   \mmzset{
519     alt/.code={
520       \keys_set:nn {tikz/tagging}{alt={#1}}
521       \bool_set_true:N \l__mmzx_ok_bool
522       \str_gset:Nn \g__mmzx_plug_orig_str {alt}
523       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
524       \socket_assign_plug:nn
525         {tagsupport/memoize/include/extern/before} {mmzx/alt}
526       \socket_assign_plug:nn
527         {tagsupport/memoize/include/extern/after} {mmzx/alt}
528 (debug) \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
529     },
530     actualtext/.code={
531       \keys_set:nn {tikz/tagging}{actualtext={#1}}
532       \bool_set_true:N \l__mmzx_ok_bool
533       \str_gset:Nn \g__mmzx_plug_orig_str {actualtext}
534       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
535       \socket_assign_plug:nn
536         {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
537       \socket_assign_plug:nn
538         {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
539 (debug) \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
540     },
541     artifact/.code={
542       \keys_set:nn {tikz/tagging}{artifact}
543       \bool_set_true:N \l__mmzx_ok_bool
544       \str_gset:Nn \g__mmzx_plug_orig_str {artifact}
545       \mmzxtagtoks {}
546       \socket_assign_plug:nn
547         {tagsupport/memoize/include/extern/before} {mmzx/artifact}
548       \socket_assign_plug:nn
549         {tagsupport/memoize/include/extern/after} {mmzx/artifact}
550 (debug) \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
551     },
552     tagging-setup/.is choice,
553     tagging-setup/alt/.forward to=/mmz/alt,
554     tagging-setup/actualtext/.forward to=/mmz/actualtext,
555     tagging-setup/artifact/.forward to=/mmz/artifact,
556     tagging-setup/.unknown/.code =

```

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

```

557   {
558     \exp_args:Nno
559     \keys_set:nn {tikz/tagging}{\pgfkeyscurrentname={#1}}
560   },
561   alt/.belongs to family=/mmzx/tagging@setup,
562   actualtext/.belongs to family=/mmzx/tagging@setup,
563   artifact/.belongs to family=/mmzx/tagging@setup,
564   tagging-setup/.belongs to family=/mmzx/tagging@setup,
565   tagging-setup/alt/.belongs to family=/mmzx/tagging@setup,
566   tagging-setup/actualtext/.belongs to family=/mmzx/tagging@setup,
567   tagging-setup/artifact/.belongs to family=/mmzx/tagging@setup,
568 }
569 }

```

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 redefine 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.

```

570 \appto\mmzAtBeginMemoization{
571   \gtoksapp\mmzCCMemo{
572     \let\mmzxIncludeExternOrig\mmzIncludeExtern
573     \let\mmzIncludeExtern\mmzxIncludeExtern
574   }
575 }

```

`_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

But does that lessen my guilt? :-)

```

576 \cs_new_protected_nopar:Npn
577   \_mmzx_noop:nNnnnnnnn #1#2#3#4#5#6#7#8#9 {}

```

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.

```

578 \cs_new_protected_nopar:Npn \__mmzx_include_extern:nNnnnnnnn #1#2#3#4#5#6#7#8#9
579 {
580   \int_gincr:N \g__mmzx_tagpic_int
581   <debug>   \__mmzx_debug:n {Args: #1; #2; #3; #4; #5; #6; #7; #8; #9}
582   \socket_use:nnnn {tagsupport/memoize/include/extern/before}
583     {#3}{#4}{#5}
584   <debug>   \__mmzx_debug:n {Executing original inclusion macro.}
585   \mmzxIncludeExternOrig {#1}#2{#3}{#4}{#5}{#6}{#7}{#8}{#9}
586   <debug>   \__mmzx_debug:n {Closing tagging structures.}
587   \socket_use:n {tagsupport/memoize/include/extern/after}
588 }
589 \cs_new_eq:NN \mmzxIncludeExtern \__mmzx_include_extern:nNnnnnnnn
590 \cs_new_eq:NN \mmzxIncludeExternOrig \__mmzx_noop:nNnnnnnnn

```

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

```

591 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/alt}
592 {
593   <debug>   \__mmzx_debug:n {Executing plug mmzx/alt in socket
594   <debug>     tagsupport/memoize/include/extern/before.}
595   \mode_if_vertical:T
596   {
597     \if@inlabel
598     \mode_leave_vertical:
599     \else
600     \tag_socket_use:n {para/begin}
601     \fi
602   }
603   \tag_mc_end_push:
604   \tl_set:No \l__mmzx_toks_tl {\the\mmzxtagtoks}
605   \tag_struct_begin:n
606   {
607     tag=Figure,
608     alt=\l__mmzx_toks_tl,
609   }
610   \tag_mc_begin:n {tag=Figure}
611   \cs_new:cpe {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
612   {
613     \__mmzx_pgftikz_tag_bbox:ennn {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
614     {#1}{#2}{#3}
615   }
616   <debug>   \cs_log:c {mmzx@tag@tikz@mark@pos@\int_to_arabic:n
617   <debug>     {\g__mmzx_tagpic_int}}
618   \tag_struct_gput:ene
619   {\tag_get:n {struct_num}}
620   {attribute}
621   {
622     /O /Layout /BBox
623     [
624     \use:c
625     {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
626     ]
627   }
628   <debug>   \__mmzx_debug:e {Recording xpos,
629   <debug>     ypos of mmxc-id\int_to_arabic:n {\g__mmzx_tagpic_int}.}
630   \tex_savepos:D

```

```

631 \_mmzx_property_record_orig:ee
632   {mmzx-id\int_to_arabic:n {\g_mmzx_tagpic_int}}
633   {xpos,ypos}
634 \tex_savepos:D
635 }

```

»>

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 «<

```

636 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/alt}
637 {
638 (debug) \_mmzx_debug:n {Executing plug mmzx/alt in socket
639 (debug)   tagsupport/memoize/include/extern/after.}
640 \tag_mc_end:
641 \tag_struct_end:
642 \tag_mc_begin_pop:n { }
643 }

```

»>

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

after mmzx/actualtext (*plug*)

```

644 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
645 {
646 (debug) \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
647 (debug)   tagsupport/memoize/include/extern/before.}
648 \keys_set:nn {tikz/tagging} {actualtext:o = {\the\mmzxtagtoks},}
649 \socket_use:n {tagsupport/tikz/picture/begin}
650 }
651 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
652 {
653 (debug) \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
654 (debug)   tagsupport/memoize/include/extern/after.}
655 \socket_use:n {tagsupport/tikz/picture/end}
656 }

```

»>

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

n/after mmzx/artifact (*plug*)

```

657 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/artifact}
658 {
659 (debug) \_mmzx_debug:n {Executing plug mmzx/artifact in socket
660 (debug)   tagsupport/memoize/include/extern/before.}
661 \keys_set:nn {tikz/tagging} {artifact,}
662 \socket_use:n {tagsupport/tikz/picture/begin}
663 }
664 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/artifact}
665 {
666 (debug) \_mmzx_debug:n {Executing plug mmzx/artifact in socket
667 (debug)   tagsupport/memoize/include/extern/after.}
668 \socket_use:n {tagsupport/tikz/picture/end}
669 }

```

»>

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

```
_pgftikz_tag_bbox:ennn (fn.)
670 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox:nmnn #1#2#3#4
671 {
672   \__mmzx_pgftikz_tag_bbox_aux:ennn
673   {
674     \mmzx_property_ref_orig:ee {#1}{xpos}
675   }
676   {
677     \mmzx_property_ref_orig:ee {#1}{ypos}
678   }
679   {#2}{#3}{#4}
680 }
681 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox:nmnn {ennn}
```

`ikz_tag_bbox_aux:nmnn` (*fn.*) Auxiliary.

```
ikz_tag_bbox_aux:ennn (fn.)
682 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox_aux:nmnn #1#2#3#4#5
683 {
684   \dim_to_decimal_in_bp:n {#1sp}
685   \c_space_tl
686   \dim_to_decimal_in_bp:n {#2sp-#5}
687   \c_space_tl
688   \dim_to_decimal_in_bp:n {#1sp+#3}
689   \c_space_tl
690   \dim_to_decimal_in_bp:n {#2sp+#4+#5}
691 }
692 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox_aux:nmnn {ennn}

693 \hook_gput_code_with_args:nnn {cmd/tikz@picture/before} {mmzx}
694 {
695   \tag_if_active:T
696   {
697     (debug) \__mmzx_debug:n {Executing mmzx code in hook cmd/tikz@picture/before.}
698     \socket_assign_plug:nn {tagsupport/tikz/picture/init} {mmzx}
699   }
700 }
```

»>

`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).

```

701 \socket_new_plug:nnn {tagsupport/tikz/picture/init} {mmzx}
702 {
703 <debug> \__mmzx_debug:n {Executing plug mmzx in socket
704 <debug> tagsupport/tikz/picture/init.}
705 \bool_set_false:N \l__mmzx_ok_bool
706 \legacy_if:nT {memoizing}
707 {
708 <debug> \__mmzx_debug:n {Arg to tagsupport/tikz/picture/init is #1.}
709 \pgfkeyssavekeyfilterstateto\mmzx@tagging@setup@saved@filterstate
710 \pgfkeysinstallkeyfilter{/pgf/key filters/active families}{}
711 \pgfqkeysactivatesinglefamilyandfilteroptions{/mmzx/tagging@setup}{/tikz}{#1}
712 <debug> \__mmzx_debug:e { Restoring filter state: \exp_not:V
713 <debug> \mmzx@tagging@setup@saved@filterstate }
714 \mmzx@tagging@setup@saved@filterstate
715 \bool_if:NF \l__mmzx_ok_bool
716 {
717 <debug> \__mmzx_debug:n {
718 <debug> No plug set for utilisation. Trying to match latex-lab plug.
719 <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.

```

720 \xtoksapp\mmzContextExtra {
721 tagging plug=\__mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}
722 }
723 \str_case_e:nn
724 {\__mmzx_socket_assigned_plug:n {tagsupport/tikz/picture/begin}}
725 {
726 {alt} {
727 \exp_args:Ne \mmzset{
728 alt = \exp_not:V \l__tikz_tagging_alt_tl,
729 }
730 <debug> \__mmzx_debug:n {Using alt plug.}
731 }
732 {actualtext} {
733 \exp_args:Ne \mmzset{
734 actualtext = \exp_not:V \l__tikz_tagging_actualtext_tl,
735 }
736 <debug> \__mmzx_debug:n {Using actualtext plug.}
737 }
738 {artifact} {
739 \mmzset{artifact,}
740 <debug> \__mmzx_debug:n {Using artifact plug.}
741 }
742 {text} {
743 \bool_set_false:N \l__mmzx_ok_bool
744 <debug> \__mmzx_debug:e {Found unsupported
745 <debug> text
746 <debug> {tagsupport/tikz/picture/begin} plug.}
747 \PackageWarning{memoize-ext}{Unsupported tag config for tikz picture.
748 Please use alt,actualtext,artifact or another supported plug.
749 Note that text (the latex-lab default) is NOT supported.

```

```

750         Aborting memoization and marking code unmemoizable.
751     }
752     \mmzUnmemoizable
753 }
754 }
755 }
756 }
757 \bool_if:NT \l__mmzx_ok_bool
758 {
759     \xtoksapp\mmzCCMemo{
760         \exp_not:N \mmzxtagtoks
761         =
762         \c_left_brace_str
763         \the\mmzxtagtoks
764         \c_right_brace_str
765         \exp_not:N \AssignSocketPlug
766         \c_left_brace_str
767         tagsupport/memoize/include/extern/before
768         \c_right_brace_str
769         \c_left_brace_str
770         \__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/before}
771         \c_right_brace_str
772         \exp_not:N \AssignSocketPlug
773         \c_left_brace_str
774         tagsupport/memoize/include/extern/after
775         \c_right_brace_str
776         \c_left_brace_str
777         \__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/after}
778         \c_right_brace_str
779     }
780 <debug>     \__mmzx_debug:n {Disabling tagging for current tikz picture during
781 <debug>         current run.}
782     \socket_assign_plug:nn {tagsupport/tikz/picture/begin} {noop}
783     \socket_assign_plug:nn {tagsupport/tikz/picture/end} {noop}
784     \socket_assign_plug:nn {tagsupport/tikz/picture/text/begin} {noop}
785     \socket_assign_plug:nn {tagsupport/tikz/picture/text/end} {noop}
786 }
787 }

788 \hook_gput_code:nnn {cmd/mmzAbort/after} {..}
789 {
790     \legacy_if:nT {memoizing} {
791         \PackageWarning{memoize-ext}{
792             Memoization has been aborted. If something like a reference needs
793             resolving, just compile again. If the content is unmemoizable --
794             e.g. contains remember picture, a breakable tcolorbox or suchlike,
795             you must mark the content unmemoizable or the tagging will be
796             incorrect. You may do this automatically or manually. Aborted
797         }
798     }
799 }

```

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

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

```
800 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{latex-lab-testphase-tikz}{>}{..}
```

```

801 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tagpdf}{>}{.}
802 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tikz}{>}{.}
803 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{latex-lab-testphase-tikz}{>}{.}
804 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tikz}{>}{.}
805 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tagpdf}{>}{.}
806 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {latex-lab-testphase-tikz}
807 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tagpdf}
808 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tikz}
809 <debug> \hook_log:n {package/tikz/after}
810 \hook_gput_code:nnn {begindocument/end} {.}
811 {
812 <debug> \hook_log:n {cmd/tikz@picture/before}
813 <debug> \hook_log:n {cmd/endpgfpicture/after}

```

»>

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

```

814 \cs_if_free:NT \mmzx_property_ref_orig:nn
815 {
816 \cs_new_eq:NN \mmzx_property_ref_orig:nn \property_ref:nn
817 \cs_generate_variant:Nn \mmzx_property_ref_orig:nn {ee}
818 }

```

»>

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

```

819 \cs_if_free:NT \_mmzx_property_record_orig:nn
820 {
821 \cs_new_eq:NN \_mmzx_property_record_orig:nn \property_record:nn
822 \cs_generate_variant:Nn \_mmzx_property_record_orig:nn {ee}
823 }

```

»>

824 }

</sty>

memoize-ext-talk

Clea F. Rees

11744 2026-01-19

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>

```
825 \GetIdInfo $Id: memoize-ext-talk.dtx 11744 2026-03-06 21:12:42Z cfrees $ {Extension
826 \!debug) \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
827 \!debug) {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
828 \debug) \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
829 \debug) {v0.3.1 \ExplFileVersion}{\ExplFileDescription}
830 %
831 \!debug) \disable@package@load {memoize-ext-talk-debug}
832 \debug) \disable@package@load {memoize-ext-talk}
833 { Only one of memoize-ext-talk and memoize-ext-talk-debug
834 should be loaded.
835 Since
836 \!debug) memoize-ext-talk
837 \debug) memoize-ext-talk-debug
838 has been loaded,I will ignore your request for
839 \debug) memoize-ext-talk
840 \!debug) memoize-ext-talk-debug
841 .}
```

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?

```
842 \!debug) \RequirePackage{memoize-ext-tag}
843 \debug) \RequirePackage{memoize-ext-tag-debug}
```

We don't want inconsistent names in hooks.

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

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

```
845 \hook_gput_code:nnn {class/ltx-talk/after}{.}
846 {
847 <debug> \__mmzx_debug:n {Enabling ltx-talk support.}
848 \mmzset{
849   per overlay/.code={},
850   talk mode to prefix/.style={
851     prefix=\mmz@prefix@dir\mmz@prefix@name\l__mmzx_talk_mode_str -,
852   },
853 }
```

`mmzx_talk_opacity_seq` (*var.*)

`talk_saved_opacity_seq` (*var.*)

`__mmzx_talk_opacity_tl` (*var.*)

```
854 \seq_new:N \g__mmzx_talk_opacity_seq
855 \seq_new:N \g__mmzx_talk_saved_opacity_seq
856 \tl_new:N \l__mmzx_talk_opacity_tl
```

`\g__mmzx_talk_frame_int`

`\g__mmzx_talk_slide_int`

`\g__mmzx_talk_pauses_int`

`\l__mmzx_talk_mode_str`

```
857
858 \cs_new_eq:NN \g__mmzx_talk_frame_int \c@frame
859 \cs_new_eq:NN \g__mmzx_talk_slide_int \c@slide
860 \cs_new_eq:NN \g__mmzx_talk_pauses_int \c@pauses
861 \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)

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

Initialise sequence.

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

`mmzx_talk_opacity_save:` (*fn.*)

```
864 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_save:
865 {
866   \seq_gset_eq:NN \g__mmzx_talk_opacity_saved_seq \g__mmzx_talk_opacity_seq
867   \seq_get:NN \g__mmzx_talk_opacity_seq \l__mmzx_talk_opacity_tl
868   \exp_args:NV \tl_if_eq:NNTF \l__mmzx_talk_opacity_tl \q_no_value
869   {
870     \seq_gpush:Nn \g__mmzx_talk_opacity_saved_seq {1}
871     \opacity_select:n {1}
872   } {
873     \seq_gpush:NV \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
874     \opacity_select:V \l__mmzx_talk_opacity_tl
875   }
876 }
```

_talk_opacity_restore: (fn.)

```

877 \cs_new_protected_nopar:Npn \_mmzx_talk_opacity_restore:
878 {
879   \seq_gpop:NN \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
880   \opacity_select:V \l__mmzx_talk_opacity_tl
881   \seq_gset_eq:NN \g__mmzx_talk_opacity_seq \g__mmzx_talk_opacity_saved_seq
882 }

```

\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 ...

```

883 \long\def\mmzSingleExternDriver#1{
884   \xtoksapp\mmzCCMemo{\mmz@maybe@quitvmode}
885   \setbox\mmz@box\mmz@capture{
886     \_mmzx_talk_opacity_save:
887     #1
888     \_mmzx_talk_opacity_restore:
889   }
890   \mmzExternalizeBox\mmz@box\mmz@temptoks
891   \xtoksapp\mmzCCMemo{\the\mmz@temptoks}
892   \mmz@maybe@quitvmode\box\mmz@box
893 }

```

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

```

894 \hook_gset_rule:nnnn {begin:document} {ltx-talk} {<} {.}
895 \hook_gput_code_with_args:nnn {cmd/opacity_select:n/before} {..}
896 {
897   \seq_gpush:Nn \g__mmzx_talk_opacity_seq {#1}
898 }
899 \hook_gput_code:nnn {cmd/opacity_end:/after}{..}
900 {
901 (debug)   \_mmzx_debug:n{Opacity after}
902   \seq_gpop:NN \g__mmzx_talk_opacity_seq \l_tmpa_tl
903 (debug)   \seq_log:N \g__mmzx_talk_opacity_seq
904 }
905 \mmzset{
906   at begin memoization={
907     \socket_use:n {mmzx/talk/memoization/begin}
908   },
909   at end memoization={
910     \socket_use:n {mmzx/talk/memoization/end}
911   },
912   per overlay/.style={
913     /mmz/context={
914       overlay=\csname theslide\endcsname,
915       pauses=\ifmemoizing
916         \mmzxTalkPauses
917       \else
918         \expandafter\the\csname c@pauses\endcsname
919       \fi
920     },
921     /utils/exec={
922       \str_if_eq:eeF {peroverlay}
923       {\_mmzx_socket_assigned_plug:n {mmzx/talk/memoization/begin}}

```

```

924     {
925         \socket_assign_plug:nn {mmzx/talk/memoization/begin}{peroverlay}
926         \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.

```

927         \legacy_if:nT {memoizing} {
928             \socket_use:n {mmzx/talk/memoization/begin}
929         }
930     }
931 },

```

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

```

932     /mmz/per overlay/.code={},
933 },
934 }

```

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

alk/memoization/end (*socket*)

```

935 \socket_new:nn {mmzx/talk/memoization/begin}{0}
936 \socket_new:nn {mmzx/talk/memoization/end}{0}

```

tion/begin peroverlay (*plug*) Plugs.

zation/end peroverlay (*plug*)

```

937 \socket_new_plug:nnn {mmzx/talk/memoization/begin}{peroverlay}
938 {
939 <debug> \__mmzx_debug:n {Executing plug peroverlay in socket
940 <debug> mmzx/talk/memoization/begin.}
941 \xdef\mmzxTalkPauses{
942     \int_to_arabic:n {\g__mmzx_talk_pauses_int}
943 }
944 \xtoksapp\mmzCMemo{
945     \noexpand\mmzxSetTalkOverlays{\mmzxTalkPauses}{
946     \int_to_arabic:n {\g__mmzx_talk_slide_int}
947     }
948 }
949 \gtoksapp\mmzCCMemo{
950     \only<all:\mmzxTalkOverlays>{}
951 }
952 \seq_get:NNTF \g__mmzx_talk_opacity_seq \l__mmzx_opacity_tl
953 {
954     \fp_gset:NV \l__mmzx_tmpa_fp \l__mmzx_opacity_tl
955 } {
956     \fp_gset:Nn \l__mmzx_tmpa_fp {1}
957 }
958 \xtoksapp\mmzContextExtra{
959     opacity=\fp_to_decimal:N \l__mmzx_tmpa_fp
960 }
961 }
962 \socket_new_plug:nnn {mmzx/talk/memoization/end}{peroverlay}
963 {
964 <debug> \__mmzx_debug:n {Executing plug peroverlay in socket
965 <debug> mmzx/talk/memoization/end.}
966 \xtoksapp\mmzCCMemo{
967     \exp_not:N \setcounter{pauses}

```

```

968     {\int_to_arabic:n {\g__mmzx_talk_pauses_int}}
969   }
970 }

```

`\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.

```

971 \cs_new_nopar:Npn \mmzxSetTalkOverlays#1#2{
972 <debug>   \__mmzx_debug:e {Executing \cs_to_str:N \mmzxSetTalkOverlays}
973   \int_compare:nNnTF {\g__mmzx_talk_pauses_int} = {#1}
974   {
975     \gdef\mmzxTalkOverlays{#2}
976     \int_compare:nNnTF {\g__mmzx_talk_slide_int} < {#2}
977     {
978       \bool_set_true:N \l__mmzx_tmpa_bool
979     }{
980       \bool_set_false:N \l__mmzx_tmpa_bool
981     }
982   }{
983     \bool_set_true:N \l__mmzx_tmpa_bool
984   }
985   \bool_if:NT \l__mmzx_tmpa_bool
986   {
987     \bool_gset_true:N \g__talk_slide_continue_bool
988     \appto\mmzAtBeginMemoization{
989       \gtoksapp\mmzCMemo{\mmzxSetTalkOverlays{#1}{#2}}
990     }
991   }
992 }

993 }

</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

v0.2		and hope \DocumentMetadata is sufficient in case tagging is off.	32
General: See <code>init</code>	23		
<code>tagsupport/tikz/picture/init_mmzx</code> : Avoid processing non-tagging keys too early and too often. Save and restore existing filter state.	28	<code>mmzx/talk/memoization/end</code> : Use sockets to try to keep memos cleaner.	35
v0.3		<code>mmzx/talk/memoization/end_peroverlay</code> : What is a socket without a plug?	35
General: Add tagging status to <code>salt</code>	10	v0.3.1	
Correct and update usage details for <code>ltx-talk</code>	3	<code>tagsupport/tikz/picture/init_mmzx</code> : Default to OK.	29
Load <code>memoize-ext-tag/memoize-ext-tag-debug</code>			

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>\@ifl@t@r</code>	5, 65	<code>__mmzx_expl_at_start</code> : (fn.)	248, <u>263</u>
<code>\@ifundefined</code>	2	<code>__mmzx_expl_replicate_aux:n</code> (fn.)	<u>358</u>
<code>__mmzx_advice_collect_draw_args:w</code> (fn.)	<u>458</u> , <u>465</u>	<u>389</u> , <u>390</u> , <u>391</u> , <u>393</u> , <u>394</u> , <u>396</u> , <u>398</u> , <u>405</u>
<code>__mmzx_cctab_end</code> : (fn.)	<u>262</u>	<code>__mmzx_expl_replicate_e</code> : (fn.)	<u>400</u> , <u>402</u> , <u>405</u>
<code>__mmzx_cctab_stop</code> : (fn.)	<u>263</u> , <u>326</u>	<code>__mmzx_expl_replicate_t</code> : (fn.)	<u>397</u> , <u>399</u> , <u>405</u>
<code>__mmzx_debug:N</code>	100	<code>__mmzx_expl_replicate_fn</code> : (fn.)	<u>422</u>
<code>__mmzx_debug:e</code>	102, 628, 712, 744, 972	<code>__mmzx_expl_replicate_fn_aux:nnN</code> (fn.)	<u>358</u> , <u>426</u>
<code>__mmzx_debug:n</code>	95, 99, 136, 145, 188, 191, 199, 230, 293, 295, 302, 306, 308, 313, 317, 581, 584, 586, 593, 638, 646, 653, 659, 666, 697, 703, 708, 717, 730, 736, 740, 780, 847, 901, 939, 964	<code>__mmzx_if_replicating</code> :	185
<code>__mmzx_expl_at_begin</code> : (fn.)	<u>263</u>	<code>__mmzx_if_replicating:F</code>	197
		<code>__mmzx_if_replicating:TF</code> (fn.)	185
		<code>__mmzx_if_replicating_p</code> : (fn.)	185
		<code>__mmzx_include_extern:nNnnnnnnn</code> (fn.)	<u>576</u>

\hook_gput_code_with_args:nnn 693, 895
 \hook_gset_rule:nnnn 800,
 801, 802, 803, 804, 805, 806, 807, 808, 894
 \hook_log:n 809, 812, 813

I

\if@inlabel 597
 \if_bool:N 187
 \IfFormatAtLeastTF 65, 66, 73
 \ifmemoizing 200, 915
 \ifmmz@direct@ccmemo@input 237
 \IfPackageLoadedF 113, 115
 \int_compare:nNnTF 973, 976
 \int_gincr:N 580
 \int_incr:N 386
 \int_new:N 86, 499
 \int_set:Nn 215
 \int_to_arabic:n 409, 416,
 611, 613, 616, 625, 629, 632, 942, 946, 968
 \int_zero:N 362
 \iow_log:n 97

K

\keys_define:nn 41
 \keys_set:nn 520, 531, 542, 559, 648, 661

L

l3draw (opt.) 3
 \l__mmzx_expl_bool (var)
 220, 226, 231, 234, 236, 255
 \l__mmzx_ok_bool (var)
 498, 521, 532, 543, 705, 715, 743, 757
 \l__mmzx_opacity_tl 952, 954
 \l__mmzx_opt_draw_bool (var) 41, 133
 \l__mmzx_opt_expl_bool (var) 41, 126
 \l__mmzx_opt_tag_bool (var) 37, 56, 105
 \l__mmzx_opt_talk_bool 58, 142
 \l__mmzx_replicating_bool (var) 183, 187, 425
 \l__mmzx_talk_mode_str 851, 857
 \l__mmzx_talk_opacity_tl (var)
 854, 867, 868, 873, 874, 879, 880
 \l__mmzx_tmpa_bool 84, 978, 980, 983, 985
 \l__mmzx_tmpa_fp 85, 954, 956, 959
 \l__mmzx_tmpa_int 86, 362, 386, 409, 416
 \l__mmzx_tmpa_seq 91
 \l__mmzx_tmpa_str 92
 \l__mmzx_tmpa_tl
 88, 363, 371, 373, 408, 409, 410, 415, 416
 \l__mmzx_tmpb_str 93
 \l__mmzx_tmpb_tl 89
 \l__mmzx_tmpc_tl
 90, 364, 366, 368, 377, 379, 407, 414
 \l__mmzx_toks_tl (var) 498, 604, 608
 \l_talk_mode_str 861
 \l_tikz_tagging_actualtext_tl 734
 \l_tikz_tagging_alt_tl 728

\l_tmpa_tl 902
 \legacy_if:nT 706, 790, 927
 \let 572, 573
 \long 883

M

\makeatletter 210, 214
 \MessageBreak 10
 mmz/auto/replicate expl fn (pgfkey) 430
 mmz/auto/replicate expl var (pgfkey) 430
 \mmz@box 885, 890, 892
 \mmz@capture 885
 \mmz@direct@ccmemo@inputfalse 242
 \mmz@direct@ccmemo@inputtrue 245
 \mmz@maybe@quitvmode 884, 892
 \mmz@prefix@dir 851
 \mmz@prefix@name 851
 \mmz@temptoks 890, 891
 \mmzAtBeginMemoization
 292, 294, 305, 307, 570, 988
 \mmzAtEndMemoization 316, 318
 \mmzCCMemo 297, 302, 309, 313, 319, 370,
 528, 539, 550, 571, 759, 884, 891, 949, 966
 \mmzCMemo 944, 989
 \mmzContextExtra 720, 958
 \mmzExternalizeBox 890
 \mmzIncludeExtern 572, 573
 \mmzSalt 151
 \mmzset 80,
 203, 430, 467, 518, 727, 733, 739, 848, 905
 \mmzSingleExternDriver 883
 \mmzUnmemoizable 752
 mmzx (plug) 6
 mmzx/actualtext (plug) 6
 mmzx/alt (plug) 6
 mmzx/artifact (plug) 6
 mmzx/talk/memoization/begin (socket) 935
 mmzx/talk/memoization/begin peroverlay
 (plug) 937
 mmzx/talk/memoization/end (socket) 935
 mmzx/talk/memoization/end peroverlay
 (plug) 937
 \mmzx@expl@replicate@fn 422, 433
 \mmzx@tagging@setup@saves@filterstate
 709, 713, 714
 \mmzx_property_ref_orig:ee 674, 677
 \mmzx_property_ref_orig:nn (fn.) 814
 \mmzxExplAtBegin 246, 256, 273, 278, 283, 300, 311
 \mmzxExplAtEnd 247, 257, 274, 279, 284, 321
 \mmzxIncludeExtern 573, 576
 \mmzxIncludeExternOrig 572, 576
 \mmzxSetTalkOverlays 945, 971
 \mmzxtagtoks (toks)
 498, 523, 534, 545, 604, 648, 760, 763
 \mmzxTalkOverlays 950, 975

tagsupport/memoize/include/extern/after	tagsupport/memoize/include/extern/before
..... 7, 503	mmzx/actualtext (plug) 644
tagsupport/memoize/include/extern/before	tagsupport/memoize/include/extern/before
..... 7, 503	mmzx/alt (plug) 591
\str_case:mnF 387	tagsupport/memoize/include/extern/before
\str_case_e:nn 723	mmzx/artifact (plug) 657
\str_gset:Nn 522, 533, 544	tagsupport/tikz/picture/init mmzx (plug) 701
\str_gset:NV 23	talk (opt.) 3
\str_if_eq:eeF 377, 922	talk mode to prefix (pgfkey) 4
\str_new:N 22, 92, 93, 500	\tex_endlinechar:D 215
\str_use:c 473	\tex_savepos:D 630, 634
\string 294, 307, 318	\text_purify:n 523, 534
\sys_if_engine_pdftex_p: 288	\the 302, 313,
\sys_if_engine_xetex_p: 290	461, 528, 539, 550, 604, 648, 763, 891, 918
T	
tag (opt.) 3	\tl_clear:N 363, 364
\tag_get:n 619	\tl_gput_right:Nn 355, 357
\tag_if_active:T 695	\tl_gput_right:NV 354, 356
\tag_if_active:TF 38	\tl_if_eq:NNTF 868
\tag_if_active_p: 152	\tl_log:e 528, 539, 550
\tag_mc_begin:n 610	\tl_map_function:nN 365
\tag_mc_begin_pop:n 642	\tl_new:N ... 88, 89, 90, 351, 352, 353, 501, 856
\tag_mc_end: 640	\tl_put_right:Ne 409, 416
\tag_mc_end_push: 603	\tl_put_right:Nn 407, 414
\tag_socket_use:n 600	\tl_put_right:NV 408, 410, 415
\tag_struct_begin:n 605	\tl_set:No 604
\tag_struct_end: 641	token registers:
\tag_struct_gput:ene 618	\mmzxtagtoks
tagsupport/memoize/include/extern/after 498, 523, 534, 545, 604, 648, 760, 763
(socket) 7, 503	\toks 460, 461, 464
tagsupport/memoize/include/extern/after	\toksapp 151
mmzx/actualtext (plug) 644	
tagsupport/memoize/include/extern/after	U
mmzx/alt (plug) 636	\use:c 382, 624
tagsupport/memoize/include/extern/after	
mmzx/artifact (plug) 657	X
tagsupport/memoize/include/extern/before	\xdef 941
(socket) 7, 503	\xtoksapp 297, 309,
	319, 370, 720, 759, 884, 891, 944, 958, 966