

memoize-ext: Extensions for memoize

Clea F. Rees*

11663 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	6
	memoize-ext	7
	memoize-ext-expl3	11
	memoize-ext-l3draw	18
	memoize-ext-sockets	20

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

memoize-ext-tag	21
memoize-ext-talk	30

Part I

Usage

1 Basics

Usage is simple.

```
\usepackage{memoize-ext}
```

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

For use with `ltx-talk` either

```
\DocumentMetadata{}
\RequirePackage{memoize}
\documentclass{ltx-talk}
\usepackage{memoize-ext}
```

or

```
\DocumentMetadata{}
\RequirePackage{memoize-ext}
\documentclass{ltx-talk}
```

may be required.

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 `LATEX`'s tagging code is activated.

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.

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~cname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
  auto~cname={__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 ‘`beamer`’.

`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` macro¹.

5 Tagged pdf

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.

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`² (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.

¹The 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 L^AT_EX's shipout routine.

²This is not necessary if you use `prooftrees`, which will load the package automatically if required.

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)³.

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 correctly⁴.

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.

`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.

³I 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.

⁴It would be possible to disable tagging only if memoization succeeds, but I am not sure whether the structure will be right in this case?

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 memos⁵.

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 11663 2026-02-21 01:18:19Z cfrees $ {Extensions for
17 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
18 \!debug} {v0.1 \ExplFileVersion}{\ExplFileDescription}
19 \debug} \ProvidesExplPackage{\ExplFileName-debug}
20 \debug} {\ExplFileDate}{v0.1 \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
```

⁵This follows memoize’s own practice.

```

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 {}

```

messages

```

151 \msg_new:nnnn {memoize-ext}{unsupported}
152 {
153   \msg_warning_text:n {memoize-ext}:
154   Non-existent or inappropriate version of #2 from #3 \msg_line_context:.
155   #4
156 } {
157   memoize-ext#1 requires an appropriate version of #2 from #3.
158 }

```

</sty>

memoize-ext-expl3

Clea F. Rees

11663 2026-01-19

Abstract

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

Contents

<@@=mmzx> <*common>

```
159 \GetIdInfo $Id: memoize-ext-expl3.dtx 11663 2026-02-21 01:18:19Z cfrees $ {Extension
160 <!debug>      \ProvidesExplPackage{\ExplFileName-common}{\ExplFileDate}{v0.0 %
161 <!debug>      \ExplFileVersion}{\ExplFileDescription}
162 <debug>      \ProvidesExplPackage{\ExplFileName-common-debug}{\ExplFileDate}{v0.0 %
163 <debug>      \ExplFileVersion}{\ExplFileDescription}
164 %
165 <!debug>      \disable@package@load {memoize-ext-expl3-common-debug}
166 <debug>      \disable@package@load {memoize-ext-expl3-common}
167 { Only one of memoize-ext-expl3-common and memoize-ext-expl3-common-debug
168   should be loaded.
169   Since
170 <!debug>      memoize-ext-expl3-common
171 <debug>      memoize-ext-expl3-common-debug
172   has been loaded,I will ignore your request for
173 <debug>      memoize-ext-expl3-common
174 <!debug>      memoize-ext-expl3-common-debug
175 .}

176 <!debug>      \RequirePackage{memoize-ext}
177 <debug>      \RequirePackage{memoize-ext-debug}
178 %
```

We don't want inconsistent names in hooks.

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

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

```
180 \bool_new:N \l__mmzx_replicating_bool
181 \bool_set_false:N \l__mmzx_replicating_bool
```

```

mmzx_if_replicating:TF (fn.)
mmzx_if_replicating_p: (fn.)
182 \prg_new_conditional:Npnn \__mmzx_if_replicating: {p,T,TF,F}
183 {
184   \if_bool:N \l__mmzx_replicating_bool
185   (debug) \__mmzx_debug:n {Replicating true.}
186   \prg_return_true:
187   \else:
188   (debug) \__mmzx_debug:n {Replicating false.}
189   \prg_return_false:
190   \fi:
191 }

```

`\AdviceRunIfNotReplicating` Run condition.

```

192 \cs_new:Npn \AdviceRunIfNotReplicating
193 {
194   \__mmzx_if_replicating:F
195   {
196   (debug) \__mmzx_debug:n {Not replicating,so proceeding.}
197   \ifmemoizing \AdviceRuntrue \fi
198   }
199 }

```

`if not replicating (pgfkey)` Style.

```

200 \mmzset{
201   auto/run if not replicating/.style = {
202     run conditions={\AdviceRunIfNotReplicating},
203   },
204 }

```

consts

```

205 \cctab_const:Nn \c__mmzx_expl_at_cctab {
206   \cctab_select:N \c_code_cctab
207   \makeatletter
208 }
209 \cctab_const:Nn \c__mmzx_nexpl_at_cctab {
210   \cctab_select:N \c_code_cctab
211   \makeatletter
212   \int_set:Nn \tex_endlinechar:D { 13 }
213   \char_set_catcode_space:n { 9 }
214   \char_set_catcode_space:n { 32 }
215   \char_set_catcode_active:n { 126 } % tilde
216 }

```

expl3, memoizing `côd ynddo`

hooks instead of [T_EX 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.

```

217 \bool_new:N \l__mmzx_expl_bool

```

```

_restore_ccmemo_input: (fn.) Initialise.
_saved_mmzxExplAtBegin: (fn.)
x_saved_mmzxExplAtEnd: (fn.)
218 \cs_new_protected_nopar:Npn \__mmzx_restore_ccmemo_input: {}
219 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtBegin: {}
220 \cs_new_protected_nopar:Npn \__mmzx_saved_mmzxExplAtEnd: {}

```

Auto-switching for expl3 syntax.

```

221 \hook_gput_code:nnn {begindocument/end}{.}
222 {
223   \bool_set_false:N \l__mmzx_expl_bool
224 }

225 \hook_gput_code:nnn {begindocument/end}{.}
226 {
227   (debug)   \__mmzx_debug:n {Tracking expl3 syntax changes.}
228   \bool_set_false:N \l__mmzx_expl_bool
229   \hook_gput_code:nnn {cmd/ExplSyntaxOn/before} { . }
230   {
231     \bool_if:NF \l__mmzx_expl_bool
232     {
233       \bool_set_true:N \l__mmzx_expl_bool
234       \ifmmz@direct@ccmemo@input
235         \relax
236       \else
237         \cs_set_protected_nopar:Npn \__mmzx_restore_ccmemo_input:
238         {
239           \mmz@direct@ccmemo@inputfalse
240         }
241       \fi
242       \mmz@direct@ccmemo@inputtrue
243       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtBegin: \mmzxExplAtBegin
244       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtEnd: \mmzxExplAtEnd
245       \__mmzx_expl_at_start:
246     }
247   }

```

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

```

248 \hook_gput_code:nnn {cmd/ExplSyntaxOn/after} { . }
249 {
250   \hook_gput_code:nnn {cmd/ExplSyntaxOff/before} { . }
251   {
252     \bool_set_false:N \l__mmzx_expl_bool
253     \cs_set_eq:NN \mmzxExplAtBegin \__mmzx_saved_mmzxExplAtBegin:
254     \cs_set_eq:NN \mmzxExplAtEnd \__mmzx_saved_mmzxExplAtEnd:
255     \__mmzx_restore_ccmemo_input:
256   }
257 }
258 }

```

fns mewnol

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

```

259 \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.) 260 \cs_new_protected_nopar:Npn \__mmzx_expl_at_begin:
\__mmzx_nexpl_at_start: (fn.) 261 {
\__mmzx_cctab_stop: (fn.) 262 \cctab_begin:N \c__mmzx_expl_at_cctab
263 }
264 \cs_new_protected_nopar:Npn \__mmzx_nexpl_at_begin:
265 {
266 \cctab_begin:N \c__mmzx_nexpl_at_cctab
267 }
268 \cs_new_nopar:Npn \__mmzx_expl_at_start:
269 {
270 \cs_set_nopar:Npn \mmzxExplAtBegin {__mmzx_expl_at_begin:}
271 \cs_set_nopar:Npn \mmzxExplAtEnd {__mmzx_cctab_end:}
272 }
273 \cs_new_nopar:Npn \__mmzx_nexpl_at_start:
274 {
275 \cs_set_nopar:Npn \mmzxExplAtBegin {__mmzx_nexpl_at_begin:}
276 \cs_set_nopar:Npn \mmzxExplAtEnd {__mmzx_cctab_end:}
277 }
278 \cs_new_nopar:Npn \__mmzx_cctab_stop:
279 {
280 \cs_set_nopar:Npn \mmzxExplAtBegin {relax}
281 \cs_set_nopar:Npn \mmzxExplAtEnd {relax}
282 }

```

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.

```

283 \bool_lazy_or:nnTF
284 {
285 \sys_if_engine_pdftex_p:
286 } {
287 \sys_if_engine_xetex_p:
288 } {
289 \appto\mmzAtBeginMemoization{
290 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
291 <debug> \string\mmzAtBeginMemoization.}
292 <debug> \__mmzx_debug:n {Working around pdfTeX/XeTeX peculiarity
293 <debug> which eats first noexpand.}
294 \xtoksapp\mmzCCMemo
295 {
296 \exp_not:N \exp_not:N
297 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \exp_not:N \exp_not:N \endcsname
298 }
299 <debug> \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
300 }
301 } {
302 \appto\mmzAtBeginMemoization{
303 <debug> \__mmzx_debug:n {Appending expl begin to ccmemo at end
304 <debug> \string\mmzAtBeginMemoization.}
305 <debug> \__mmzx_debug:n {LuaTeX doesn't eat first noexpand.}
306 \xtoksapp\mmzCCMemo
307 {
308 \exp_not:N \csname \mmzxExplAtBegin \exp_not:N \endcsname

```

```

309     }
310 <debug>    \exp_args:No \__mmzx_debug:n {\the\mmzCCMemo}
311   }
312 }
313 \preto\mmzAtEndMemoization{
314 <debug>    \__mmzx_debug:n {Appending expl end to ccmemo at start
315 <debug>    \string\mmzAtEndMemoization.}
316   \xtoksapp\mmzCCMemo
317   {
318     \exp_not:N \csname \mmzxExplAtEnd \exp_not:N \endcsname
319   }
320 }

```

init

```

321 \hook_gput_code:nnn { begindocument/end } {mmzx}
322 {
323 % % % % % %    \__mmzx_cctab_stop:
324 % }

```

</common>

<*sty>

```

325 \GetIdInfo $Id: memoize-ext-expl3.dtx 11663 2026-02-21 01:18:19Z cfrees $ {Extension
326 <!debug>    \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
327 <!debug>    {v0.1 \ExplFileVersion}{\ExplFileDescription}
328 <debug>    \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
329 <debug>    {v0.1 \ExplFileVersion}{\ExplFileDescription}
330 %
331 <!debug>    \disable@package@load {memoize-ext-expl3-debug}
332 <debug>    \disable@package@load {memoize-ext-expl3}
333 { Only one of memoize-ext-expl3 and memoize-ext-expl3-debug
334   should be loaded.
335   Since
336 <!debug>    memoize-ext-expl3
337 <debug>    memoize-ext-expl3-debug
338   has been loaded,I will ignore your request for
339 <debug>    memoize-ext-expl3
340 <!debug>    memoize-ext-expl3-debug
341 .}

342 <!debug>    \RequirePackage{memoize-ext}
343 <debug>    \RequirePackage{memoize-ext-debug}
344 <!debug>    \RequirePackage{memoize-ext-expl3-common}
345 <debug>    \RequirePackage{memoize-ext-expl3-common-debug}
346 %

```

We don't want inconsistent names in hooks.

```

347 \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.)

```

348 \tl_new:N \g__mmzx_expl_replicate__bb_tl
349 \tl_new:N \g__mmzx_expl_replicate__ba_tl
350 \tl_new:N \g__mmzx_expl_replicate__tb_tl
351 \tl_gput_right:NV \g__mmzx_expl_replicate__bb_tl \c_left_brace_str
352 \tl_gput_right:Nn \g__mmzx_expl_replicate__bb_tl {#}
353 \tl_gput_right:NV \g__mmzx_expl_replicate__ba_tl \c_right_brace_str
354 \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.

```

355 \cs_new:Npn \__mmzx_expl_replicate_fn_aux:nnN #1#2#3
356 {
357   \cs_if_exist:cF { __mmzx_rep_#1:#2 }
358   {
359     \int_zero:N \l__mmzx_tmpa_int
360     \tl_clear:N \l__mmzx_tmpa_tl
361     \tl_clear:N \l__mmzx_tmpc_tl
362     \tl_map_function:nN {#2} \__mmzx_expl_replicate__aux:n
363     \cs_if_exist:cF { __mmzx_rep_#1:\l__mmzx_tmpc_tl }
364     {
365       \cs_gset_protected:ce {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
366       {
367         \xtoksapp\mmzCCMemo{
368           \exp_after:wN \exp_not:N \cs:w #1:#2 \cs_end: \l__mmzx_tmpa_tl
369         }
370         \exp_not:N \expandonce \exp_not:N \AdviceOriginal \l__mmzx_tmpa_tl
371         \exp_not:N \group_end:
372       }
373     }
374     \str_if_eq:eeF {#2}{\l__mmzx_tmpc_tl}
375     { % ych!
376       \cs_new_eq:cc {__mmzx_rep_#1:#2} {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
377     }
378   }
379   \use:c { __mmzx_rep_#1:#2 }
380 }
381 \cs_new:Npn \__mmzx_expl_replicate__aux:n #1
382 {
383   \int_incr:N \l__mmzx_tmpa_int
384   \str_case:nnF { #1 }
385   {
386     {c} { \__mmzx_expl_replicate__b: }
387     {e} { \__mmzx_expl_replicate__b: }
388     {o} { \__mmzx_expl_replicate__b: }
389     {p} { }
390     {n} { \__mmzx_expl_replicate__b: }
391     {v} { \__mmzx_expl_replicate__b: }
392     {D} { }
393     {F} { \__mmzx_expl_replicate__b: }
394     {N} { \__mmzx_expl_replicate__t: }
395     {T} { \__mmzx_expl_replicate__b: }
396     {V} { \__mmzx_expl_replicate__t: }
397     {w} { \__mmzx_expl_replicate__e: }
398   }{
399     \__mmzx_expl_replicate__e:

```



```

400 }
401 }

```

`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.

```

402 \cs_new_nopar:Npn \__mmzx_expl_replicate__b:
403 {
404   \tl_put_right:Nn \l__mmzx_tmpc_tl {n}
405   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__bb_tl
406   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
407   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__ba_tl
408 }
409 \cs_new_nopar:Npn \__mmzx_expl_replicate__t:
410 {
411   \tl_put_right:Nn \l__mmzx_tmpc_tl {N}
412   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__tb_tl
413   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
414 }
415 \cs_new_nopar:Npn \__mmzx_expl_replicate__e:
416 {
417   \PackageError{mmzx}{No do, sorry. Use replicate with args instead.}{}
418 }

```

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

```

\mmzx@expl@replicate@fn
419 \cs_new:Npn \__mmzx_expl_replicate_fn:
420 {
421   \group_begin:
422   \bool_set_true:N \l__mmzx_replicating_bool
423   \exp_last_unbraced:Ne \__mmzx_expl_replicate_fn_aux:nnN
424   { \exp_args:NV \cs_split_function:N \AdviceReplaced }
425 }
426 \cs_new_eq:NN \mmzx@expl@replicate@fn \__mmzx_expl_replicate_fn:

```

`auto/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.

```

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

</sty>

```

memoize-ext-l3draw

Clea F. Rees

11663 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>

```
433 \GetIdInfo $Id: memoize-ext-l3draw.dtx 11663 2026-02-21 01:18:19Z cfrees $ {Extensi
434 \!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
435 \!debug> {v0.1 \ExplFileVersion}{\ExplFileDescription}
436 \!debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
437 \!debug> {v0.1 \ExplFileVersion}{\ExplFileDescription}
438 %
439 \!debug> \disable@package@load {memoize-ext-l3draw-debug}
440 \!debug> \disable@package@load {memoize-ext-l3draw}
441 { Only one of memoize-ext-l3draw and memoize-ext-l3draw-debug
442 should be loaded.
443 Since
444 \!debug> memoize-ext-l3draw
445 \!debug> memoize-ext-l3draw-debug
446 has been loaded,I will ignore your request for
447 \!debug> memoize-ext-l3draw
448 \!debug> memoize-ext-l3draw-debug
449 .}
450 %
451 \!debug> \RequirePackage{memoize-ext}
452 \!debug> \RequirePackage{memoize-ext-debug}
453 \!debug> \RequirePackage{memoize-ext-expl3}
454 \!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.

```
455 \cs_new:Npn \__mmzx_advice_collect_draw_args:w #1 \draw_end:
456 {
457 \toks0={ #1 \draw_end: }
458 \exp_args:No \AdviceInnerHandler {\the\toks0}
459 }
```

AdviceCollectDrawArguments Public wrapper.

```
460 \cs_new:Npn \AdviceCollectDrawArguments{
461   \toks0 = {}
462   \__mmzx_advice_collect_draw_args:w
463 }
```

Default configuration.

```
464 \mmzset{%
465   auto csname={draw_begin:}{memoize,collector=\AdviceCollectDrawArguments,},
466   auto csname={__draw_record_origin:}{run if memoizing,replicate expl fn,},
467 }
```

</sty>

memoize-ext-sockets

Clea F. Rees

11663 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.

```
468 \cs_new_nopar:Npn \__mmzx_socket_assigned_plug:n #1
469 { % rhybudd: fn mewno1
470   \str_use:c { l__socket_#1_plug_str }
471 }
```

</sty>

memoize-ext-tag

Clea F. Rees

11663 2026-01-19

Abstract

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

Contents

Should probably use unique prefix ...?

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 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>

```

472 \GetIdInfo $Id: memoize-ext-tag.dtx 11663 2026-02-21 01:18:19Z cfrees $ {Extensions
473 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
474 \!debug} {v0.1 \ExplFileVersion}{\ExplFileDescription}
475 \debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
476 \debug} {v0.1 \ExplFileVersion}{\ExplFileDescription}
477 %
478 \!debug} \disable@package@load {memoize-ext-tag-debug}
479 \debug} \disable@package@load {memoize-ext-tag}
480 { Only one of memoize-ext-tag and memoize-ext-tag-debug
481 should be loaded.
482 Since
483 \!debug} memoize-ext-tag
484 \debug} memoize-ext-tag-debug
485 has been loaded,I will ignore your request for
486 \debug} memoize-ext-tag
487 \!debug} memoize-ext-tag-debug
488 .}

489 \!debug} \RequirePackage{memoize-ext}
490 \debug} \RequirePackage{memoize-ext-debug}
491 %

```

We don't want inconsistent names in hooks.

```

492 \SetDefaultHookLabel{memoize-ext}

```

```

\g__mmzx_tagpic_int (var.) Tracking & storage variables.
  \l__mmzx_toks_tl (var.)
  \l__mmzx_ok_bool (var.)
    \mmzxtagtoks
493 \bool_new:N \l__mmzx_ok_bool
494 \int_new:N \g__mmzx_tagpic_int
495 \tl_new:N \l__mmzx_toks_tl
496 \newtoks\mmzxtagtoks

```

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

```

include/extern/after (socket)
497 \socket_new:nn {tagsupport/memoize/include/extern/before} {3}
498 \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 valuse 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

```

499 \hook_gput_code:nnn {package/tikz/after} {..}
500 {
501   \tikzset{
502     alt/.forward to=/mmz/alt,
503     actualtext/.forward to=/mmz/actualtext,
504     artifact/.forward to=/mmz/artifact,
505   }
506   \mmzset{
507     alt/.code={
508       \keys_set:nn {tikz/tagging}{alt={#1}}
509       \bool_set_true:N \l__mmzx_ok_bool
510       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
511       \socket_assign_plug:nn
512         {tagsupport/memoize/include/extern/before} {mmzx/alt}
513       \socket_assign_plug:nn
514         {tagsupport/memoize/include/extern/after} {mmzx/alt}
515 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
516     },
517     actualtext/.code={
518       \keys_set:nn {tikz/tagging}{actualtext={#1}}
519       \bool_set_true:N \l__mmzx_ok_bool
520       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
521       \socket_assign_plug:nn
522         {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
523       \socket_assign_plug:nn
524         {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
525 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
526     },
527     artifact/.code={
528       \keys_set:nn {tikz/tagging}{artifact}
529       \bool_set_true:N \l__mmzx_ok_bool
530       \mmzxtagtoks {}
531       \socket_assign_plug:nn
532         {tagsupport/memoize/include/extern/before} {mmzx/artifact}
533       \socket_assign_plug:nn
534         {tagsupport/memoize/include/extern/after} {mmzx/artifact}
535 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
536     },
537   }
538 }

```

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.

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

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.

```

539 \appto\mmzAtBeginMemoization{
540   \gtoksapp\mmzCCMemo{
541     \let\mmzxIncludeExternOrig\mmzIncludeExtern
542     \let\mmzIncludeExtern\mmzxIncludeExtern
543   }
544 }
```

`_mmzx_noop:nNnnnnnnn` (*fn.*) `\mmzIncludeExtern` only seems to be defined during utilisation, so we set up a command `\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? :-)

```

545 \cs_new_protected_nopar:Npn
546   \_mmzx_noop:nNnnnnnnn #1#2#3#4#5#6#7#8#9 {}
547 \cs_new_protected_nopar:Npn \_mmzx_include_extern:nNnnnnnnn #1#2#3#4#5#6#7#8#9
548 {
549   \int_gincr:N \g__mmzx_tagpic_int
550   <debug>   \_mmzx_debug:n {Args: #1; #2; #3; #4; #5; #6; #7; #8; #9}
551   \socket_use:nnnn {tagsupport/memoize/include/extern/before}
552     {#3}{#4}{#5}
553   <debug>   \_mmzx_debug:n {Executing original inclusion macro.}
554   \mmzxIncludeExternOrig {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}
555   <debug>   \_mmzx_debug:n {Closing tagging structures.}
556   \socket_use:n {tagsupport/memoize/include/extern/after}
557 }
558 \cs_new_eq:NN \mmzxIncludeExtern \_mmzx_include_extern:nNnnnnnnn
559 \cs_new_eq:NN \mmzxIncludeExternOrig \_mmzx_noop:nNnnnnnnn
```

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

```

560 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/alt}
561 {
562   <debug>   \_mmzx_debug:n {Executing plug mmzx/alt in socket
563   <debug>     tagsupport/memoize/include/extern/before.}
564   \mode_if_vertical:T
565   {
566     \if@inlabel
567       \mode_leave_vertical:
568     \else
569       \tag_socket_use:n {para/begin}
570     \fi
```

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.


```

571 }
572 \tag_mc_end_push:
573 \tl_set:No \l__mmzx_toks_tl {\the\mmzxtagtoks}
574 \tag_struct_begin:n
575 {
576   tag=Figure,
577   alt=\l__mmzx_toks_tl,
578 }
579 \tag_mc_begin:n {tag=Figure}
580 \cs_new:cpe {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
581 {
582   \__mmzx_pgftikz_tag_bbox:ennn {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
583   {#1}{#2}{#3}
584 }
585 <debug> \cs_log:c {mmzx@tag@tikz@mark@pos@\int_to_arabic:n
586 <debug> {\g__mmzx_tagpic_int}}
587 \tag_struct_gput:ene
588 {\tag_get:n {struct_num}}
589 {attribute}
590 {
591   /O /Layout /BBox
592   [
593     \use:c
594     {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
595   ]
596 }
597 <debug> \__mmzx_debug:e {Recording xpos,
598 <debug> ypos of mmxc-id\int_to_arabic:n {\g__mmzx_tagpic_int}.}
599 \tex_savepos:D
600 \__mmzx_property_record_orig:ee
601 {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
602 {xpos,ypos}
603 \tex_savepos:D
604 }

```

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

```

605 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/alt}
606 {
607 <debug> \__mmzx_debug:n {Executing plug mmzx/alt in socket
608 <debug> tagsupport/memoize/include/extern/after.}
609 \tag_mc_end:
610 \tag_struct_end:
611 \tag_mc_begin_pop:n {}
612 }

```

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

after mmzx/actualtext (*plug*)

```

613 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
614 {
615 <debug> \__mmzx_debug:n {Executing plug mmzx/actualtext in socket
616 <debug> tagsupport/memoize/include/extern/before.}
617 \keys_set:nn {tikz/tagging} {actualtext:o = {\the\mmzxtagtoks},}
618 \socket_use:n {tagsupport/tikz/picture/begin}
619 }

```

```

620 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
621 {
622 <debug>      \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
623 <debug>      tagsupport/memoize/include/extern/after.}
624 \socket_use:n {tagsupport/tikz/picture/end}
625 }

```

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

n/after mmzx/artifact (*plug*)

```

626 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/artifact}
627 {
628 <debug>      \_mmzx_debug:n {Executing plug mmzx/artifact in socket
629 <debug>      tagsupport/memoize/include/extern/before.}
630 \keys_set:nn {tikz/tagging} {artifact,}
631 \socket_use:n {tagsupport/tikz/picture/begin}
632 }
633 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/artifact}
634 {
635 <debug>      \_mmzx_debug:n {Executing plug mmzx/artifact in socket
636 <debug>      tagsupport/memoize/include/extern/after.}
637 \socket_use:n {tagsupport/tikz/picture/end}
638 }

```

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

_pgftikz_tag_bbox:ennn (*fn.*)

```

639 \cs_new_nopar:Npn \_mmzx_pgftikz_tag_bbox:nnnn #1#2#3#4
640 {
641 \_mmzx_pgftikz_tag_bbox_aux:eennn
642 {
643 \mmzx_property_ref_orig:ee {#1}{xpos}
644 }
645 {
646 \mmzx_property_ref_orig:ee {#1}{ypos}
647 }
648 {#2}{#3}{#4}
649 }
650 \cs_generate_variant:Nn \_mmzx_pgftikz_tag_bbox:nnnn {ennn}

```

ikz_tag_bbox_aux:nnnn (*fn.*) Auxiliary.

ikz_tag_bbox_aux:eennn (*fn.*)

```

651 \cs_new_nopar:Npn \_mmzx_pgftikz_tag_bbox_aux:nnnn #1#2#3#4#5
652 {
653 \dim_to_decimal_in_bp:n {#1sp}
654 \c_space_tl
655 \dim_to_decimal_in_bp:n {#2sp-#5}
656 \c_space_tl
657 \dim_to_decimal_in_bp:n {#1sp+#3}
658 \c_space_tl
659 \dim_to_decimal_in_bp:n {#2sp+#4+#5}
660 }
661 \cs_generate_variant:Nn \_mmzx_pgftikz_tag_bbox_aux:nnnn {eennn}

662 \hook_gput_code_with_args:nnn {cmd/tikz@picture/before} {mmzx}
663 {
664 \tag_if_active:T
665 {

```

```

666 <debug>      \_mmzx_debug:n {Executing mmzx code in hook cmd/tikz@picture/before.}
667      \socket_assign_plug:nn {tagssupport/tikz/picture/init} {mmzx}
668    }
669 }

```

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.

```

670 \socket_new_plug:nnn {tagssupport/tikz/picture/init} {mmzx}
671 {
672 <debug>      \_mmzx_debug:n {Executing plug mmzx in socket
673 <debug>      tagssupport/tikz/picture/init.}
674      \bool_set_false:N \l__mmzx_ok_bool
675      \legacy_if:nT {memoizing}
676    {
677 <debug>      \_mmzx_debug:n {Arg to tagssupport/tikz/picture/init is #1.}
678      \pgfqkeys{/pgf}{%
679      key filters/defined/.install key filter,
680      }%
681      % \pgfkeysfiltered{/mmz/.cd,#1}%
682      \bool_if:NF \l__mmzx_ok_bool
683    {
684 <debug>      \_mmzx_debug:n {
685 <debug>      No plug set for utilisation. Trying to match latex-lab plug.
686 <debug>      }
687      \str_case_e:nnF
688      {\_mmzx_socket_assigned_plug:n {tagssupport/tikz/picture/begin}}
689      {
690      {alt} {
691      \exp_args:Ne \mmzset{
692      alt = \exp_not:V \l__tikz_tagging_alt_tl,
693      }
694 <debug>      \_mmzx_debug:n {Using alt plug.}
695      }
696      {actualtext} {
697      \exp_args:Ne \mmzset{
698      actualtext = \exp_not:V \l__tikz_tagging_actualtext_tl,
699      }
700 <debug>      \_mmzx_debug:n {Using actualtext plug.}
701      }
702      {artifact} {

```

```

703         \mmzset{artifact,}
704 <debug>         \_mmzx_debug:n {Using artifact plug.}
705     }
706 }{
707 <debug>         \_mmzx_debug:e {Found unsupported
708 <debug>         \_mmzx_socket_assigned_plug:n
709 <debug>         {tagsupport/tikz/picture/begin} plug.}
710     \PackageWarning{memoize-ext}{Unsupported tag config for tikz picture.
711         Please use one of alt,actualtext and artifact.
712         Note that text (the latex-lab default) is NOT supported.
713         Aborting memoization.
714     }
715     \mmzAbort
716 }
717 }
718 }
719 \bool_if:NTF \l__mmzx_ok_bool
720 {
721     \xtoksapp\mmzCCMemo{
722         \exp_not:N \mmzxtagtoks
723         =
724         \c_left_brace_str
725         \the\mmzxtagtoks
726         \c_right_brace_str
727         \exp_not:N \AssignSocketPlug
728         \c_left_brace_str
729         tagsupport/memoize/include/extern/before
730         \c_right_brace_str
731         \c_left_brace_str
732         \_mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/before}
733         \c_right_brace_str
734         \exp_not:N \AssignSocketPlug
735         \c_left_brace_str
736         tagsupport/memoize/include/extern/after
737         \c_right_brace_str
738         \c_left_brace_str
739         \_mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/after}
740         \c_right_brace_str
741     }
742 <debug>         \_mmzx_debug:n {Disabling tagging for current tikz picture during
743 <debug>         current run.}
744     \socket_assign_plug:nn {tagsupport/tikz/picture/begin} {noop}
745     \socket_assign_plug:nn {tagsupport/tikz/picture/end} {noop}
746     \socket_assign_plug:nn {tagsupport/tikz/picture/text/begin} {noop}
747     \socket_assign_plug:nn {tagsupport/tikz/picture/text/end} {noop}
748 } {
749 <debug>         \_mmzx_debug:n {Not memoizing. Enabling tagging for current tikz
750 <debug>         picture.}
751     \socket_assign_plug:nn {tagsupport/tikz/picture/init} {default}
752     \socket_use:nn {tagsupport/tikz/picture/init} {#1}
753 }
754 }

```

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

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

```

755 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{latex-lab-testphase-tikz}{>}{.}
756 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tagpdf}{>}{.}
757 \hook_gset_rule:nnnn {cmd/tikz@picture/before}{tikz}{>}{.}
758 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{latex-lab-testphase-tikz}{>}{.}
759 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tikz}{>}{.}
760 \hook_gset_rule:nnnn {cmd/endpgfpicture/after}{tagpdf}{>}{.}
761 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {latex-lab-testphase-tikz}
762 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tagpdf}
763 \hook_gset_rule:nnnn {package/tikz/after} {.} {>} {tikz}
764 <debug> \hook_log:n {package/tikz/after}
765 \hook_gput_code:nnn {begindocument/end} {.}
766 {
767 <debug> \hook_log:n {cmd/tikz@picture/before}
768 <debug> \hook_log:n {cmd/endpgfpicture/after}

```

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

```

769 \cs_if_free:NT \mmzx_property_ref_orig:nn
770 {
771   \cs_new_eq:NN \mmzx_property_ref_orig:nn \property_ref:nn
772   \cs_generate_variant:Nn \mmzx_property_ref_orig:nn {ee}
773 }

```

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

```

774 \cs_if_free:NT \__mmzx_property_record_orig:nn
775 {
776   \cs_new_eq:NN \__mmzx_property_record_orig:nn \property_record:nn
777   \cs_generate_variant:Nn \__mmzx_property_record_orig:nn {ee}
778 }

779 }

```

</sty>

memoize-ext-talk

Clea F. Rees

11663 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>

```
780 \GetIdInfo $Id: memoize-ext-talk.dtx 11663 2026-02-21 01:18:19Z cfrees $ {Extension
781 \!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
782 \!debug> {v0.1 \ExplFileVersion}{\ExplFileDescription}
783 \debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
784 \debug> {v0.1 \ExplFileVersion}{\ExplFileDescription}
785 %
786 \!debug> \disable@package@load {memoize-ext-talk-debug}
787 \debug> \disable@package@load {memoize-ext-talk}
788 { Only one of memoize-ext-talk and memoize-ext-talk-debug
789 should be loaded.
790 Since
791 \!debug> memoize-ext-talk
792 \debug> memoize-ext-talk-debug
793 has been loaded,I will ignore your request for
794 \debug> memoize-ext-talk
795 \!debug> memoize-ext-talk-debug
796 .}
797 \!debug> \RequirePackage{memoize-ext}
798 \debug> \RequirePackage{memoize-ext-debug}
```

We don’t want inconsistent names in hooks.

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

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

```

800 \hook_gput_code:nnn {class/ltx-talk/after}{.}
801 {
802   <debug>    \__mmzx_debug:n {Enabling ltx-talk support.}
803   \mmzset{
804     per overlay/.code={},
805     talk mode to prefix/.style={
806       prefix=\mmz@prefix@dir\mmz@prefix@name\l__mmzx_talk_mode_str -,
807     },
808   }

```

mmzx_talk_opacity_seq (var.)

alk_saved_opacity_seq (var.)

_mmzx_talk_opacity_tl (var.)

```

809 \seq_new:N \g__mmzx_talk_opacity_seq
810 \seq_new:N \g__mmzx_talk_saved_opacity_seq
811 \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

```

812
813 \cs_new_eq:NN \g__mmzx_talk_frame_int \c@frame
814 \cs_new_eq:NN \g__mmzx_talk_slide_int \c@slide
815 \cs_new_eq:NN \g__mmzx_talk_pauses_int \c@pauses
816 \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)

```

817 \cs_generate_variant:Nn \opacity_select:n {V}

```

Initialise sequence.

```

818 \seq_gpush:Nn \g__mmzx_talk_opacity_seq {1}

```

mmzx_talk_opacity_save: (fn.)

```

819 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_save:
820 {
821   \seq_gset_eq:NN \g__mmzx_talk_opacity_saved_seq \g__mmzx_talk_opacity_seq
822   \seq_get:NN \g__mmzx_talk_opacity_seq \l__mmzx_talk_opacity_tl
823   \exp_args:NV \tl_if_eq:NNTF \l__mmzx_talk_opacity_tl \q_no_value
824   {
825     \seq_gpush:Nn \g__mmzx_talk_opacity_saved_seq {1}
826     \opacity_select:n {1}
827   } {
828     \seq_gpush:NV \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
829     \opacity_select:V \l__mmzx_talk_opacity_tl
830   }
831 }

```

_talk_opacity_restore: (fn.)

```

832 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_restore:
833 {

```

```

834 \seq_gpop:NN \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
835 \opacity_select:V \l__mmzx_talk_opacity_tl
836 \seq_gset_eq:NN \g__mmzx_talk_opacity_seq \g__mmzx_talk_opacity_saved_seq
837 }

```

`\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 ...

```

838 \long\def\mmzSingleExternDriver#1{
839 \xtoksapp\mmzCCMemo{\mmz@maybe@quitvmode}
840 \setbox\mmz@box\mmz@capture{
841 \__mmzx_talk_opacity_save:
842 #1
843 \__mmzx_talk_opacity_restore:
844 }
845 \mmzExternalizeBox\mmz@box\mmz@temptoks
846 \xtoksapp\mmzCCMemo{\the\mmz@temptoks}
847 \mmz@maybe@quitvmode\box\mmz@box
848 }

```

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

```

849 \hook_gset_rule:nnnn {begindocument} {ltx-talk} {<} {..}
850 \hook_gput_code_with_args:nnn {cmd/opacity_select:n/before} {..}
851 {
852 \seq_gpush:Nn \g__mmzx_talk_opacity_seq {#1}
853 % \typeout{Opacity select: #1}
854 }
855 \hook_gput_code:nnn {cmd/opacity_end:/after}{..}
856 {
857 (debug) \__mmzx_debug:n{Opacity after}
858 \seq_gpop:NN \g__mmzx_talk_opacity_seq \l_tmpa_tl
859 (debug) \seq_log:N \g__mmzx_talk_opacity_seq
860 }
861 \mmzset{
862 per overlay/.style={
863 /mmz/context={
864 overlay=\csname theslide\endcsname,
865 pauses=\ifmemoizing
866 \mmzxTalkPauses
867 \else
868 \expandafter\the\csname c@pauses\endcsname
869 \fi
870 },
871 /mmz/at begin memoization={
872 \xdef\mmzxTalkPauses{
873 \int_to_arabic:n {\g__mmzx_talk_pauses_int}
874 }
875 \xtoksapp\mmzCCMemo{
876 \noexpand\mmzxSetTalkOverlays{\mmzxTalkPauses}{
877 \int_to_arabic:n {\g__mmzx_talk_slide_int}
878 }
879 }
880 \gtoksapp\mmzCCMemo{
881 \only<all:\mmzxTalkOverlays>{}
882 }

```



```

883     \seq_get:NNTF \g__mmzx_talk_opacity_seq \l__mmzx_opacity_tl
884     {
885         \fp_gset:NV \l__mmzx_tmpa_fp \l__mmzx_opacity_tl
886     } {
887         \fp_gset:Nn \l__mmzx_tmpa_fp {1}
888     }
889     \xtoksapp\mmzContextExtra{
890         opacity=\fp_to_decimal:N \l__mmzx_tmpa_fp
891     }
892 },
893 /mmz/at end memoization={
894     \xtoksapp\mmzCCMemo{
895         \exp_not:N \setcounter{pauses}
896         {\int_to_arabic:n {\g__mmzx_talk_pauses_int}}
897     }
898 },
899 /mmz/per overlay/.code={},
900 },
901 }

```

`\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.

```

902 \cs_new_nopar:Npn \mmzxSetTalkOverlays#1#2{
903 (debug)    \__mmzx_debug:e {Executing \cs_to_str:N \mmzxSetTalkOverlays}
904     \int_compare:nNnTF {\g__mmzx_talk_pauses_int} = {#1}
905     {
906         \gdef\mmzxTalkOverlays{#2}
907         \int_compare:nNnTF {\g__mmzx_talk_slide_int} < {#2}
908         {
909             \bool_set_true:N \l__mmzx_tmpa_bool
910         }{
911             \bool_set_false:N \l__mmzx_tmpa_bool
912         }
913     }{
914         \bool_set_true:N \l__mmzx_tmpa_bool
915     }
916     \bool_if:NT \l__mmzx_tmpa_bool
917     {
918         \bool_gset_true:N \g__talk_slide_continue_bool
919         \appto\mmzAtBeginMemoization{
920             \gtoksapp\mmzCCMemo{\mmzxSetTalkOverlays{#1}{#2}}
921         }
922     }
923 }
924 }

```

</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](#).

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>\@ifundefined</code>	2
<code>__mmzx_advice_collect_draw_args:w</code> (fn.)	455, 462
<code>__mmzx_cctab_end:</code> (fn.)	259
<code>__mmzx_cctab_stop:</code> (fn.)	260, 323
<code>__mmzx_debug:N</code>	100
<code>__mmzx_debug:e</code>	102, 597, 707, 903
<code>__mmzx_debug:n</code>	95, 99, 136, 145, 185, 188, 196, 227, 290, 292, 299, 303, 305, 310, 314, 550, 553, 555, 562, 607, 615, 622, 628, 635, 666, 672, 677, 684, 694, 700, 704, 742, 749, 802, 857
<code>__mmzx_expl_at_begin:</code> (fn.)	260
<code>__mmzx_expl_at_start:</code> (fn.)	245, 260
<code>__mmzx_expl_replicate_aux:n</code> (fn.)	355
<code>__mmzx_expl_replicate_b:</code> (fn.)	386, 387, 388, 390, 391, 393, 395, 402
<code>__mmzx_expl_replicate_e:</code> (fn.)	397, 399, 402
<code>__mmzx_expl_replicate_t:</code> (fn.)	394, 396, 402
<code>__mmzx_expl_replicate_fn:</code> (fn.)	419
<code>__mmzx_expl_replicate_fn_aux:nnN</code> (fn.)	355, 423
<code>__mmzx_if_replicating:</code>	182
<code>__mmzx_if_replicating:F</code>	194
<code>__mmzx_if_replicating:TF</code> (fn.)	182
<code>__mmzx_if_replicating_p:</code> (fn.)	182
<code>__mmzx_include_extern:nNnnnnnnn</code> (fn.)	545
<code>__mmzx_nexpl_at_begin:</code> (fn.)	260
<code>__mmzx_nexpl_at_start:</code> (fn.)	260
<code>__mmzx_noop:</code>	149
<code>__mmzx_noop:n</code>	149
<code>__mmzx_noop:nNnnnnnnn</code> (fn.)	545
<code>__mmzx_pgftikz_tag_bbox:ennn</code> (fn.)	582, 639
<code>__mmzx_pgftikz_tag_bbox:nnnn</code> (fn.)	639
<code>__mmzx_pgftikz_tag_bbox_aux:eennn</code> (fn.)	641, 651
<code>__mmzx_pgftikz_tag_bbox_aux:nnnnn</code> (fn.)	651
<code>__mmzx_property_record_orig:ee</code>	600
<code>__mmzx_property_record_orig:nn</code> (fn.)	774
<code>__mmzx_restore_ccmemo_input:</code> (fn.)	218, 237, 255
<code>__mmzx_saved_mmzxExplAtBegin:</code> (fn.)	218, 243, 253
<code>__mmzx_saved_mmzxExplAtEnd:</code> (fn.)	218, 244, 254
<code>__mmzx_socket_assigned_plug:n</code> (fn.)	468, 688, 708, 732, 739
<code>__mmzx_talk_opacity_restore:</code> (fn.)	832, 843
<code>__mmzx_talk_opacity_save:</code> (fn.)	819, 841
A	
<code>\AdviceCollectDrawArguments</code>	460, 465
<code>\AdviceInnerHandler</code>	458
<code>\AdviceOriginal</code>	370
<code>\AdviceReplaced</code>	424
<code>\AdviceRunIfNotReplicating</code>	192, 202
<code>\AdviceRuntrue</code>	197
<code>\appto</code>	289, 302, 539, 919
<code>\AssignSocketPlug</code>	727, 734

auto/run if not replicating (pgfkey) [200](#)

B

\bool_gset_true:N [918](#)
 \bool_if:NF [231](#), [682](#)
 \bool_if:NT [105](#), [126](#), [133](#), [142](#), [916](#)
 \bool_if:NTF [719](#)
 \bool_lazy_or:nnTF [283](#)
 \bool_new:N [37](#), [84](#), [180](#), [217](#), [493](#)
 \bool_set_false:N [40](#), [181](#), [223](#), [228](#), [252](#), [674](#), [911](#)
 \bool_set_true:N
 [39](#), [233](#), [422](#), [509](#), [519](#), [529](#), [909](#), [914](#)
 \box [847](#)

C

\c@frame [813](#)
 \c@pauses [815](#)
 \c@slide [814](#)
 \c_mmzx_expl_at_cctab [205](#), [262](#)
 \c_mmzx_nexpl_at_cctab [209](#), [266](#)
 \c_code_cctab [206](#), [210](#)
 \c_left_brace_str . [351](#), [724](#), [728](#), [731](#), [735](#), [738](#)
 \c_right_brace_str [353](#), [726](#), [730](#), [733](#), [737](#), [740](#)
 \c_space_tl [654](#), [656](#), [658](#)
 \cctab_begin:N [262](#), [266](#)
 \cctab_const:Nn [205](#), [209](#)
 \cctab_end: [259](#)
 \cctab_select:N [206](#), [210](#)
 \char_set_catcode_active:n [215](#)
 \char_set_catcode_space:n [213](#), [214](#)
 \cs:w [368](#)
 \cs_end: [368](#)
 \cs_generate_variant:Nn
 [99](#), [650](#), [661](#), [772](#), [777](#), [817](#)
 \cs_gset_protected:ce [365](#)
 \cs_if_exist:cF [357](#), [363](#)
 \cs_if_exist_use:c [77](#)
 \cs_if_free:NT [769](#), [774](#)
 \cs_log:c [585](#)
 \cs_new:cpe [580](#)
 \cs_new:Npn [149](#), [150](#), [192](#), [355](#), [381](#), [419](#), [455](#), [460](#)
 \cs_new_eq:cc [376](#)
 \cs_new_eq:NN [259](#),
[426](#), [558](#), [559](#), [771](#), [776](#), [813](#), [814](#), [815](#), [816](#)
 \cs_new_nopar:Npn [268](#),
[273](#), [278](#), [402](#), [409](#), [415](#), [468](#), [639](#), [651](#), [902](#)
 \cs_new_protected:Npn [95](#), [100](#)
 \cs_new_protected_nopar:Npn
 . [218](#), [219](#), [220](#), [260](#), [264](#), [545](#), [547](#), [819](#), [832](#)
 \cs_set_eq:NN [243](#), [244](#), [253](#), [254](#)
 \cs_set_nopar:Npn . [270](#), [271](#), [275](#), [276](#), [280](#), [281](#)
 \cs_set_protected_nopar:Npn [237](#)
 \cs_split_function:N [424](#)
 \cs_to_str:N [102](#), [903](#)
 \csname [297](#), [308](#), [318](#), [864](#), [868](#)
 \CurrentOption [63](#)

D

\DeclareUnknownKeyHandler [62](#)
 \def [838](#)
 \dim_to_decimal_in_bp:n ... [653](#), [655](#), [657](#), [659](#)
 \disable@package@load [25](#), [26](#), [165](#),
[166](#), [331](#), [332](#), [439](#), [440](#), [478](#), [479](#), [786](#), [787](#)
 \draw_end: [455](#), [457](#)

E

\else [236](#), [568](#), [867](#)
 \else: [187](#)
 \endcsname [297](#), [308](#), [318](#), [864](#), [868](#)
 \endinput [13](#)
 \exp_after:wN [368](#)
 \exp_args:Ne [510](#), [520](#), [691](#), [697](#)
 \exp_args:No [299](#), [310](#), [458](#), [515](#), [525](#), [535](#)
 \exp_args:NV [102](#), [424](#), [823](#)
 \exp_last_unbraced:Ne [423](#)
 \exp_not:N [296](#), [297](#),
[308](#), [318](#), [368](#), [370](#), [371](#), [722](#), [727](#), [734](#), [895](#)
 \exp_not:n [102](#), [515](#), [525](#), [535](#)
 \exp_not:V [692](#), [698](#)
 \expandafter [868](#)
 \ExpandArgs [76](#)
 \expandonce [370](#)
 expl3 (opt.) [3](#)
 expl3 functions:
 _mmzx_advice_collect_draw_args:w ...
 [455](#), [462](#)
 _mmzx_cctab_end: [259](#)
 _mmzx_cctab_stop: [260](#), [323](#)
 _mmzx_expl_at_begin: [260](#)
 _mmzx_expl_at_start: [245](#), [260](#)
 _mmzx_expl_replicate_aux:n [355](#)
 _mmzx_expl_replicate_b:
 [386](#), [387](#), [388](#), [390](#), [391](#), [393](#), [395](#), [402](#)
 _mmzx_expl_replicate_e: .. [397](#), [399](#), [402](#)
 _mmzx_expl_replicate_t: .. [394](#), [396](#), [402](#)
 _mmzx_expl_replicate_fn: [419](#)
 _mmzx_expl_replicate_fn_aux:nnN [355](#), [423](#)
 _mmzx_if_replicating:TF [182](#)
 _mmzx_if_replicating_p: [182](#)
 _mmzx_include_extern:nNnnnnnnn ... [545](#)
 _mmzx_nexpl_at_begin: [260](#)
 _mmzx_nexpl_at_start: [260](#)
 _mmzx_noop:nNnnnnnnnn [545](#)
 _mmzx_pgftikz_tag_bbox:ennn .. [582](#), [639](#)
 _mmzx_pgftikz_tag_bbox:nnnn [639](#)
 _mmzx_pgftikz_tag_bbox_aux:ennnn [641](#), [651](#)
 _mmzx_pgftikz_tag_bbox_aux:nnnnn .. [651](#)
 _mmzx_property_record_orig:nn [774](#)
 _mmzx_restore_ccmemo_input: [218](#), [237](#), [255](#)
 _mmzx_saved_mmzxExplAtBegin:
 [218](#), [243](#), [253](#)
 _mmzx_saved_mmzxExplAtEnd: [218](#), [244](#), [254](#)

_mmzx_socket_assigned_plug:n	468, 688, 708, 732, 739
_mmzx_talk_opacity_restore:	..	832, 843
_mmzx_talk_opacity_save:	819, 841
\mmzx_property_ref_orig:nn	769
\opacity_select:V	817, 829, 835
expl3 variables:		
\g_mmzx_expl_replicate__ba_tl	..	348, 407
\g_mmzx_expl_replicate__bb_tl	..	348, 405
\g_mmzx_expl_replicate__tb_tl	..	348, 412
\g_mmzx_tagpic_int	493, 549, 580, 582, 586, 594, 598, 601
\g_mmzx_talk_opacity_seq	809, 818, 821, 822, 836, 852, 858, 859, 883
\g_mmzx_talk_saved_opacity_seq	809
\l_mmzx_expl_bool	217, 223, 228, 231, 233, 252
\l_mmzx_ok_bool	493, 509, 519, 529, 674, 682, 719
\l_mmzx_opt_draw_bool	41, 133
\l_mmzx_opt_expl_bool	41, 126
\l_mmzx_opt_tag_bool	37, 56, 105
\l_mmzx_replicating_bool	...	180, 184, 422
\l_mmzx_talk_opacity_tl	809, 822, 823, 828, 829, 834, 835
\l_mmzx_toks_tl	493, 573, 577
\ExplFileDate	17, 20, 160, 162, 326, 328, 434, 436, 473, 475, 781, 783
\ExplFileDescription	18, 20, 161, 163, 327, 329, 435, 437, 474, 476, 782, 784
\ExplFileName	17, 19, 23, 160, 162, 326, 328, 434, 436, 473, 475, 781, 783
\ExplFileVersion	18, 20, 161, 163, 327, 329, 435, 437, 474, 476, 782, 784
\ExplLoaderFileDate	5
F		
\fi	197, 241, 570, 869
\fi:	190
\fmtversion	65
\fp_gset:Nn	887
\fp_gset:Nv	885
\fp_new:N	85
\fp_to_decimal:N	890
G		
\g_mmzx_expl_replicate__ba_tl (var)	..	348, 407
\g_mmzx_expl_replicate__bb_tl (var)	..	348, 405
\g_mmzx_expl_replicate__tb_tl (var)	..	348, 412
\g_mmzx_name_str	22, 23, 107, 108, 128, 129, 135, 137
\g_mmzx_tagpic_int (var)	493, 549, 580, 582, 586, 594, 598, 601
\g_mmzx_talk_frame_int	812
\g_mmzx_talk_opacity_saved_seq	821, 825, 828, 834, 836
\g_mmzx_talk_opacity_seq (var)	809, 818, 821, 822, 836, 852, 858, 859, 883
\g_mmzx_talk_pauses_int	..	812, 873, 896, 904
\g_mmzx_talk_saved_opacity_seq (var)	...	809
\g_mmzx_talk_slide_int	812, 877, 907
\g_talk_slide_continue_bool	918
\gdef	906
\GetIdInfo	16, 159, 325, 433, 472, 780
\group_begin:	421
\group_end:	371
\gtoksapp	540, 880, 920
H		
\hook_gput_code:nnn	..	109, 111, 131, 140, 221, 225, 229, 248, 250, 321, 499, 765, 800, 855
\hook_gput_code_with_args:nnn	662, 850
\hook_gset_rule:nnnn	755, 756, 757, 758, 759, 760, 761, 762, 763, 849
\hook_log:n	764, 767, 768
I		
\if@inlabel	566
\if_bool:N	184
\IfFormatAtLeastTF	65, 66, 73
\ifmemoizing	197, 865
\ifmmz@direct@ccmemo@input	234
\IfPackageLoadedF	113, 115
\int_compare:nNnTF	904, 907
\int_gincr:N	549
\int_incr:N	383
\int_new:N	86, 494
\int_set:Nn	212
\int_to_arabic:n	406, 413, 580, 582, 585, 594, 598, 601, 873, 877, 896
\int_zero:N	359
\iow_log:n	97
K		
\keys_define:nn	41
\keys_set:nn	508, 518, 528, 617, 630
L		
l3draw (opt.)	3
\l_mmzx_expl_bool (var)	217, 223, 228, 231, 233, 252
\l_mmzx_ok_bool (var)	493, 509, 519, 529, 674, 682, 719
\l_mmzx_opacity_tl	883, 885
\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)	..	180, 184, 422
\l_mmzx_talk_mode_str	806, 812

\l_mmzx_talk_opacity_tl (var)		\mmzxExplAtBegin	243, 253, 270, 275, 280, 297, 308
.....	809, 822, 823, 828, 829, 834, 835	\mmzxExplAtEnd	244, 254, 271, 276, 281, 318
\l_mmzx_tmpa_bool	84, 909, 911, 914, 916	\mmzxIncludeExtern	542, 545
\l_mmzx_tmpa_fp	85, 885, 887, 890	\mmzxIncludeExternOrig	541, 545
\l_mmzx_tmpa_int	86, 359, 383, 406, 413	\mmzxSetTalkOverlays	876, 902
\l_mmzx_tmpa_seq	91	\mmzxxtagtoks	493, 510, 520, 530, 573, 617, 722, 725
\l_mmzx_tmpa_str	92	\mmzxTalkOverlays	881, 906
\l_mmzx_tmpa_tl		\mmzxTalkPauses	866, 872, 876
..	88, 360, 368, 370, 405, 406, 407, 412, 413	\mode_if_vertical:T	564
\l_mmzx_tmpr_str	93	\mode_leave_vertical:	567
\l_mmzx_tmpr_tl	89	\msg_line_context:	154
\l_mmzx_tmprc_tl		\msg_new:nnnn	151
.....	90, 361, 363, 365, 374, 376, 404, 411	\msg_warning:nnnnnn	117
\l_mmzx_toks_tl (var)	493, 573, 577	\msg_warning_text:n	153
\l_talk_mode_str	816		N
\l_tikz_tagging_actualtext_tl	698	\NeedsTeXFormat	1
\l_tikz_tagging_alt_tl	692	\newtoks	496
\l_tmpa_tl	858	\noexpand	876
\legacy_if:nT	675		O
\let	541, 542	\only	881
\long	838	\opacity_select:n	817, 826
	M	\opacity_select:V (fn.)	817, 829, 835
\makeatletter	207, 211	options:	
\MessageBreak	10	expl3	3
mmz/auto/replicate expl fn (pgfkey)	427	l3draw	3
mmz/auto/replicate expl var (pgfkey)	427	tag	3
\mmz@box	840, 845, 847	talk	3
\mmz@capture	840		P
\mmz@direct@ccmemo@inputfalse	239	\PackageError	7, 417
\mmz@direct@ccmemo@inputtrue	242	\PackageWarning	45, 710
\mmz@maybe@quitvmode	839, 847	\PassOptionsToPackage	63
\mmz@prefix@dir	806	per overlay (pgfkey)	4
\mmz@prefix@name	806	pgfkeys:	
\mmz@temptoks	845, 846	auto/run if not replicating	200
\mmzAbort	715	mmz/auto/replicate expl fn	427
\mmzAtBeginMemoization		mmz/auto/replicate expl var	427
.....	289, 291, 302, 304, 539, 919	per overlay	4
\mmzAtEndMemoization	313, 315	replicate expl fn	4
\mmzCCMemo	294, 299, 306, 310, 316, 367,	talk mode to prefix	4
.....	515, 525, 535, 540, 721, 839, 846, 880, 894	\pgfkeysfiltered	681
\mmzCMemo	875, 920	\pgfqkeys	678
\mmzContextExtra	889	plugins:	
\mmzExternalizeBox	845	mmzx	5
\mmzIncludeExtern	541, 542	mmzx/actualtext	6
\mmzset	80,	mmzx/alt	6
.....	200, 427, 464, 506, 691, 697, 703, 803, 861	mmzx/artifact	6
\mmzSingleExternDriver	838	tagsupport/memoize/include/extern/after	
mmzx (plug)	5	mmzx/actualtext	613
mmzx/actualtext (plug)	6	tagsupport/memoize/include/extern/after	
mmzx/alt (plug)	6	mmzx/alt	605
mmzx/artifact (plug)	6	tagsupport/memoize/include/extern/after	
\mmzx@expl@replicate@fn	419, 430	mmzx/artifact	626
\mmzx_property_ref_orig:ee	643, 646		
\mmzx_property_ref_orig:nn (fn.)	769		

tagssupport/memoize/include/extern/before mmzx/actualtext	613
tagssupport/memoize/include/extern/before mmzx/alt	560
tagssupport/memoize/include/extern/before mmzx/artifact	626
tagssupport/tikz/picture/init mmzx	670
\pretoskip	313
\prg_new_conditional:Npn	182
\prg_return_false:	189
\prg_return_true:	186
\ProcessKeyOptions	68
\ProcessKeysOptions	71
\property_record:n	776
\property_ref:n	771
\providecommand	65, 76
\ProvidesExplPackage	17, 19, 160, 162, 326, 328, 434, 436, 473, 475, 781, 783
Q	
\q_mmzx_stop	87
\q_no_value	823
\quark_new:N	87
R	
\relax	235
replicate expl fn (pgfkey)	4
\RequirePackage	3, 70, 75, 79, 107, 108, 128, 129, 135, 137, 144, 146, 176, 177, 342, 343, 344, 345, 451, 452, 453, 454, 489, 490, 797, 798
S	
\seq_get:NN	822
\seq_get:NNTF	883
\seq_gpop:NN	834, 858
\seq_gpunch:Nn	818, 825, 852
\seq_gpunch:Nv	828
\seq_gset_eq:NN	821, 836
\seq_log:N	859
\seq_new:N	91, 809, 810
\setbox	840
\setcounter	895
\SetDefaultHookLabel	36, 179, 347, 492, 799
\socket_assign_plug:n	511, 513, 521, 523, 531, 533, 667, 744, 745, 746, 747, 751
\socket_new:n	497, 498
\socket_new_plug:nnn	560, 605, 613, 620, 626, 633, 670
\socket_use:n	556, 618, 624, 631, 637
\socket_use:nn	752
\socket_use:nnnn	551
sockets: tagssupport/memoize/include/extern/after	6, 497
tagssupport/memoize/include/extern/before	6, 497
\str_case:nnF	384
\str_gset:Nv	23
\str_if_eq:eeF	374
\str_new:N	22, 92, 93
\str_use:c	470
\string	291, 304, 315
\sys_if_engine_pdftex_p:	285
\sys_if_engine_xetex_p:	287
T	
tag (opt.)	3
\tag_get:n	588
\tag_if_active:T	664
\tag_if_active:TF	38
\tag_mc_begin:n	579
\tag_mc_begin_pop:n	611
\tag_mc_end:	609
\tag_mc_end_push:	572
\tag_socket_use:n	569
\tag_struct_begin:n	574
\tag_struct_end:	610
\tag_struct_gput:ene	587
tagssupport/memoize/include/extern/after (socket)	6, 497
tagssupport/memoize/include/extern/after mmzx/actualtext (plug)	613
tagssupport/memoize/include/extern/after mmzx/alt (plug)	605
tagssupport/memoize/include/extern/after mmzx/artifact (plug)	626
tagssupport/memoize/include/extern/before (socket)	6, 497
tagssupport/memoize/include/extern/before mmzx/actualtext (plug)	613
tagssupport/memoize/include/extern/before mmzx/alt (plug)	560
tagssupport/memoize/include/extern/before mmzx/artifact (plug)	626
tagssupport/tikz/picture/init mmzx (plug)	670
talk (opt.)	3
talk mode to prefix (pgfkey)	4
\tex_endlinechar:D	212
\tex_savepos:D	599, 603
\text_purify:n	510, 520
\the	299, 310, 458, 515, 525, 535, 573, 617, 725, 846, 868
\tikzset	501
\tl_clear:N	360, 361
\tl_gput_right:Nn	352, 354
\tl_gput_right:Nv	351, 353
\tl_if_eq:NNTF	823
\tl_log:e	515, 525, 535
\tl_map_function:nN	362
\tl_new:N ...	88, 89, 90, 348, 349, 350, 495, 811
\tl_put_right:Ne	406, 413

<code>\tl_put_right:Nn</code>	404, 411	U	
<code>\tl_put_right:NV</code>	405, 407, 412	<code>\use:c</code>	379, 593
<code>\tl_set:No</code>	573	X	
<code>\toks</code>	457, 458, 461	<code>\xdef</code>	872
<code>\typeout</code>	853	<code>\xtoksapp</code>	294, 306, 316, 367, 721, 839, 846, 875, 889, 894