

memoize-ext: Extensions for memoize

Clea F. Rees*

v0.4.3 11978 2026/06/25

Abstract

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

Contents

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

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

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

Part I

Usage

1 Basics

Usage is simple.

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

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

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

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

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

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

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

`expl3 (opt.) = true|false`

Loads code supporting `expl3` syntax.

Default is `true`. Initially `false`.

`l3draw (opt.) = true|false`

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

Default is `true`. Initially `true`.

`tag (opt.) = true|false`

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

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

`talk (opt.) = true|false`

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

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

Default is `true`. Initially `true`.

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

2 expl3

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

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

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

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

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

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

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

3 l3draw

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

4 ltx-talk

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

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

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

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

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

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

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

5 Tagged pdf

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

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

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

5.1 TikZ pictures

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

5.2 Other content

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

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

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

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

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

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

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

`tagssupport/memoize/include/extern/before`

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

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

`tagssupport/memoize/include/extern/after`

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

5.2.1 expl3 functions

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

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

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

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

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

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

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

during utilisation.

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

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

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

during utilisation.

`\mmzx_tag_socket_plug_record:`

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

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

during utilisation.

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

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

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

Part II

Implementation

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

memoize-ext

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

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

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

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

almost verbatim from `siunitx.sty`

should check date requirement (copied from `chronos`)

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

⁶This follows `memoize`’s own practice.


```

7  \PackageError{memoize-ext}{Support package expl3 too old}
8  {%
9    You need to update your installation of the bundles 'l3kernel' and
10   'l3packages'.\MessageBreak
11   Loading memoize-ext will abort!%
12 }%
13  \endinput
14 }%
15  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16  \GetIdInfo $Id: memoize-ext.dtx 11978 2026-06-25 01:01:43Z cfrees $ {Extensions for
17  <debug>    \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
18  <debug>    {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
19  <debug>    \ProvidesExplPackage{\ExplFileName-debug}
20  <debug>    {\ExplFileDate}{v0.4.3 \ExplFileVersion}{\ExplFileDescription}
21  %
22  \str_new:N \g__mmzx_name_str
23  \str_gset:NV \g__mmzx_name_str \ExplFileName
24  %
25  <debug>    \disable@package@load {memoize-ext-debug}
26  <debug>    \disable@package@load {memoize-ext}
27  {
28    Only one of memoize-ext and memoize-ext-debug should be loaded.
29    Since
30  <debug>    memoize-ext
31  <debug>    memoize-ext-debug
32    had been loaded,I will ignore your request for
33  <debug>    memoize-ext
34  <debug>    memoize-ext-debug
35  .}
36  \SetDefaultHookLabel{memoize-ext}

```

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

```

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

```

`\l__mmzx_opt_draw_bool (var.)` Other bools.

`\l__mmzx_opt_expl_bool (var.)`

```

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

```

```

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

```

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

The fix is applied unconditionally if the format is at least the June 2026 release. Otherwise, the fix doesn't get applied to the `dev` format, since it claims to be the November release even in June (sigh)⁷. If I can find a way to distinguish the release, I'll make it conditional again but, for now, see if this works.

```

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

```

Pass unknown options to `memoize`, loaded below.

```

75 \DeclareUnknownKeyHandler[memoize-ext]{
76   \PassOptionsToPackage{\CurrentOption}{memoize}
77 }

```

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

```

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

79 \IfFormatAtLeastTF { 2022-06-01 }
80 {
81   \ProcessKeyOptions [ memoize-ext ]
82 }{
83   \RequirePackage { l3keys2e }
84   \ProcessKeysOptions { memoize-ext }
85 }

86 \IfFormatAtLeastTF { 2020-10-01 }{
87 }{
88   \RequirePackage { xparse }
89   \providecommand \ExpandArgs [1]
90   { \cs_if_exist_use:c { exp_args:N #1 } }
91 }

```

⁷I originally assumed the `dev` releases would carry 'real' dates, but apparently not.

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

^^ ????

```

92 \RequirePackage{memoize}
93 <debug>      \mmzset{
94 <debug>      trace,
95 <debug>      include context in ccmemo,
96 <debug>      }

```

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

```

97 \sys_if_engine luatex:F
98 {
99   \clist_map_inline:nn {\toksapp,\etoksapp,\gtoksapp,\xtoksapp}
100   {
101     \tl_if_head_eq_meaning:VNF #1 \protected
102     {
103       \expandafter\protected\expandafter\def\expandafter#1\expandafter{#1}
104     }
105   }
106 }

```

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

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

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

```

107 \cs_new_protected_nopar:Npn \@mmzx@kernel@after@shipout@background
108 {
109   \use:c {@kernel@after@shipout@background}
110 }
111 \FirstAidNeededT{memoize}{sty}{2024/12/02 v1.4.1 Fast and flexible externalization}
112 \sys_if_engine_pdftex:T {
113   \hook_gput_code:nnn {begindocument/end} {.} {
114     \hook_gput_code:nnn {cmd/mmz@shipout@extern/before} {.} {
115       \noexpand\@mmzx@kernel@after@shipout@background
116     }
117   }
118 }
119 }

```

temporary variables, quarks

```

120 \bool_new:N \l__mmzx_tmpa_bool
121 \fp_new:N \l__mmzx_tmpa_fp
122 \int_new:N \l__mmzx_tmpa_int
123 \quark_new:N \q__mmzx_stop
124 \tl_new:N \l__mmzx_tmpa_tl
125 \tl_new:N \l__mmzx_tmpb_tl
126 \tl_new:N \l__mmzx_tmpe_tl
127 \seq_new:N \l__mmzx_tmpa_seq
128 \str_new:N \l__mmzx_tmpa_str
129 \str_new:N \l__mmzx_tmpe_str
130 <*debug>
131 \cs_new_protected:Npn \__mmzx_debug:n #1
132 {
133   \iow_log:n {[mmzx debug]:: #1}
134 }
135 \cs_generate_variant:Nn \__mmzx_debug:n {e}
136 \cs_new_protected:Npn \__mmzx_debug:N #1
137 {
138   \__mmzx_debug:e {\cs_to_str:N #1: \exp_args:NV \exp_not:n #1}
139 }
140 </debug>

```

__mmzx_noop: Do nothing successfully.

```

\__mmzx_noop:n
141 \cs_new:Npn \__mmzx_noop: {}
142 \cs_new_eq:NN \__mmzx_noop:n \use_none:n

```

tag, expl, l3draw, talk loaded conditionally

```

143 \bool_if:NT \l__mmzx_opt_tag_bool
144 {
145   <!debug> \RequirePackage{\g__mmzx_name_str -tag}
146   <debug> \RequirePackage{\g__mmzx_name_str -tag-debug}
147   \hook_gput_code:nnn {package/forest/after}{.}
148   {
149     \hook_gput_code:nnn {begindocument/before}{.}
150     {
151       \IfPackageLoadedF {forest-lib-ext.tagging}
152       {
153         \IfPackageLoadedF {forest-lib-ext.tagging-debug}
154         {
155           \msg_warning:nnnnnn {memoize-ext}{unsupported}{forest}
156           {forest-lib-ext.tagging.sty}{forest-ext}
157           {forest trees will not be correctly tagged and may cause fatal
158            compilation errors.}
159         }
160       }
161     }
162   }
163 }

```

memoize-ext-expl3[-debug]

```

164 \bool_if:NT \l__mmzx_opt_expl3_bool
165 {
166   <!debug> \RequirePackage{\g__mmzx_name_str -expl3}
167   <debug> \RequirePackage{\g__mmzx_name_str -expl3-debug}

```

```
168 }
```

```
memoize-ext-l3draw[-debug]
```

```
169 \hook_gput_code:nnn {package/l3draw/after}{.}
170 {
171   \bool_if:NT \l__mmzx_opt_draw_bool
172   {
173     <!debug>      \RequirePackage {\g__mmzx_name_str -l3draw}
174     <debug>       \__mmzx_debug:n {Loading memoize-ext-l3draw-debug.}
175     <debug>       \RequirePackage {\g__mmzx_name_str -l3draw-debug}
176   }
177 }
```

```
memoize-ext-talk[-debug]
```

```
178 \hook_gput_code:nnn {class/ltx-talk/after} {.}
179 {
180   \bool_if:NT \l__mmzx_opt_talk_bool
181   {
182     <!debug>      \RequirePackage{memoize-ext-talk}
183     <debug>       \__mmzx_debug:n {Loading memoize-ext-talk-debug.}
184     <debug>       \RequirePackage{memoize-ext-talk-debug}
185   }
186 }
```

```
NaCl | halen | salt
```

```
187 \toksapp\mmzSalt{
188   Tagging status: \tag_if_active_p:
189 }
```

```
messages
```

```
190 \msg_new:nnnn {memoize-ext}{unsupported}
191 {
192   \msg_warning_text:n {memoize-ext}:
193   Non-existent or inappropriate version of #2 from #3 \msg_line_context:.
194   #4
195 } {
196   memoize-ext#1 requires an appropriate version of #2 from #3.
197 }
```

```
</sty>
```

memoize-ext-expl3

Clea F. Rees

11978 2026-06-18

Abstract

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

Contents

<@@=mmzx> <*common>

```
198 \GetIdInfo $Id: memoize-ext-expl3.dtx 11978 2026-06-25 01:01:43Z cfrees $ {Extension
199 \!debug} \ProvidesExplPackage{\ExplFileName-common}{\ExplFileDate}{v0.0 %
200 \!debug} \ExplFileVersion}{\ExplFileDescription}
201 \!debug} \ProvidesExplPackage{\ExplFileName-common-debug}{\ExplFileDate}{v0.0 %
202 \!debug} \ExplFileVersion}{\ExplFileDescription}
203 %
204 \!debug} \disable@package@load {memoize-ext-expl3-common-debug}
205 \!debug} \disable@package@load {memoize-ext-expl3-common}
206 { Only one of memoize-ext-expl3-common and memoize-ext-expl3-common-debug
207 should be loaded.
208 Since
209 \!debug} memoize-ext-expl3-common
210 \!debug} memoize-ext-expl3-common-debug
211 has been loaded,I will ignore your request for
212 \!debug} memoize-ext-expl3-common
213 \!debug} memoize-ext-expl3-common-debug
214 .}

215 \!debug} \RequirePackage{memoize-ext}
216 \!debug} \RequirePackage{memoize-ext-debug}
217 %
```

We don't want inconsistent names in hooks.

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

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

```
219 \bool_new:N \l__mmzx_replicating_bool
220 \bool_set_false:N \l__mmzx_replicating_bool
```

```

mmzx_if_replicating:TF (fn.)
mmzx_if_replicating_p: (fn.)
221 \prg_new_conditional:Npnn \_mmzx_if_replicating: {p,T,TF,F}
222 {
223   \if_bool:N \l__mmzx_replicating_bool
224   <debug> \_mmzx_debug:n {Replicating true.}
225   \prg_return_true:
226   \else:
227   <debug> \_mmzx_debug:n {Replicating false.}
228   \prg_return_false:
229   \fi:
230 }

```

\AdviceRunIfNotReplicating Run condition.

```

231 \cs_new:Npn \AdviceRunIfNotReplicating
232 {
233   \_mmzx_if_replicating:F
234   {
235   <debug> \_mmzx_debug:n {Not replicating,so proceeding.}
236   \ifmemoizing \AdviceRuntrue \fi
237   }
238 }

```

if not replicating (*pgfkey*) Style.

```

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

```

Constants

```

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

```

expl3, memoizing c  d ynddo

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

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

```

256 \bool_new:N \l__mmzx_expl_bool

```

```

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

```

Auto-switching for expl3 syntax.

```

260 \hook_gput_code:nnn {begindocument/end}{.}
261 {
262   \bool_set_false:N \l__mmzx_expl_bool
263 }

264 \hook_gput_code:nnn {begindocument/end}{.}
265 {
266   <debug> \__mmzx_debug:n {Tracking expl3 syntax changes.}
267   \bool_set_false:N \l__mmzx_expl_bool
268   \hook_gput_code:nnn {cmd/ExplSyntaxOn/before} { . }
269   {
270     \bool_if:NF \l__mmzx_expl_bool
271     {
272       \bool_set_true:N \l__mmzx_expl_bool
273       \ifmmz@direct@ccmemo@input
274         \relax
275       \else
276         \cs_set_protected_nopar:Npn \__mmzx_restore_ccmemo_input:
277         {
278           \mmz@direct@ccmemo@inputfalse
279         }
280       \fi
281       \mmz@direct@ccmemo@inputtrue
282       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtBegin: \mmzxExplAtBegin
283       \cs_set_eq:NN \__mmzx_saved_mmzxExplAtEnd: \mmzxExplAtEnd
284       \__mmzx_expl_at_start:
285     }
286   }

```

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

```

287 \hook_gput_code:nnn {cmd/ExplSyntaxOn/after} { . }
288 {
289   \hook_gput_code:nnn {cmd/ExplSyntaxOff/before} { . }
290   {
291     \bool_set_false:N \l__mmzx_expl_bool
292     \cs_set_eq:NN \mmzxExplAtBegin \__mmzx_saved_mmzxExplAtBegin:
293     \cs_set_eq:NN \mmzxExplAtEnd \__mmzx_saved_mmzxExplAtEnd:
294     \__mmzx_restore_ccmemo_input:
295   }
296 }
297 }

```

fns mewnol

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

```

298 \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.) 299 \cs_new_protected_nopar:Npn \__mmzx_expl_at_begin:
\__mmzx_nexpl_at_start: (fn.) 300 {
\__mmzx_cctab_stop: (fn.) 301 \cctab_begin:N \c__mmzx_expl_at_cctab
302 }
303 \cs_new_protected_nopar:Npn \__mmzx_nexpl_at_begin:
304 {
305 \cctab_begin:N \c__mmzx_nexpl_at_cctab
306 }
307 \cs_new_nopar:Npn \__mmzx_expl_at_start:
308 {
309 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_expl_at_begin:}
310 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
311 }
312 \cs_new_nopar:Npn \__mmzx_nexpl_at_start:
313 {
314 \cs_set_nopar:Npn \mmzxExplAtBegin {\__mmzx_nexpl_at_begin:}
315 \cs_set_nopar:Npn \mmzxExplAtEnd {\__mmzx_cctab_end:}
316 }
317 \cs_new_protected_nopar:Npn \__mmzx_cctab_stop:
318 {
319 \cs_set_nopar:Npn \mmzxExplAtBegin {relax}
320 \cs_set_nopar:Npn \mmzxExplAtEnd {relax}
321 }

```

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.

```

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

```

init

```

341 \hook_gput_code:nnn { begindocument/end } {mmzx}
342 {
343 % % % % % % \__mmzx_cctab_stop:
344 % }

```

</common>

<*sty>

```

345 \GetIdInfo $Id: memoize-ext-expl3.dtx 11978 2026-06-25 01:01:43Z cfrees $ {Extension
346 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
347 \!debug} {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
348 \debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
349 \debug} {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
350 %
351 \!debug} \disable@package@load {memoize-ext-expl3-debug}
352 \debug} \disable@package@load {memoize-ext-expl3}
353 { Only one of memoize-ext-expl3 and memoize-ext-expl3-debug
354 should be loaded.
355 Since
356 \!debug} memoize-ext-expl3
357 \debug} memoize-ext-expl3-debug
358 has been loaded,I will ignore your request for
359 \debug} memoize-ext-expl3
360 \!debug} memoize-ext-expl3-debug
361 .}

362 \!debug} \RequirePackage{memoize-ext}
363 \debug} \RequirePackage{memoize-ext-debug}
364 \!debug} \RequirePackage{memoize-ext-expl3-common}
365 \debug} \RequirePackage{memoize-ext-expl3-common-debug}
366 %

```

We don't want inconsistent names in hooks.

```
367 \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.)

```

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

```

375 \cs_new:Npn \__mmzx_expl_replicate_fn_aux:nnN #1#2#3
376 {
377 \cs_if_exist:cF { __mmzx_rep_#1:#2 }
378 {
379 \int_zero:N \l__mmzx_tmpa_int
380 \tl_clear:N \l__mmzx_tmpa_tl
381 \tl_clear:N \l__mmzx_tmpc_tl
382 \tl_map_function:nN {#2} \__mmzx_expl_replicate__aux:n

```

```

383 \cs_if_exist:cF { __mmzx_rep_#1:\l__mmzx_tmpc_tl }
384 {
385   \cs_gset_protected:ce {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
386   {
387     \xtoksapp\mmzCCMemo{
388       \exp_after:wN \exp_not:N \cs:w #1:#2 \cs_end: \l__mmzx_tmpa_tl
389     }
390     \exp_not:N \expandonce \exp_not:N \AdviceOriginal \l__mmzx_tmpa_tl
391     \exp_not:N \group_end:
392     \__mmzx_tmp_cctab_stop:
393   }
394 }
395 \str_if_eq:eeF {#2}{\l__mmzx_tmpc_tl}
396 { % ych!
397   \cs_new_eq:cc {__mmzx_rep_#1:#2} {__mmzx_rep_#1:\l__mmzx_tmpc_tl}
398 }
399 }
400 \use:c { __mmzx_rep_#1:#2 }
401 }
402 \cs_new:Npn \__mmzx_expl_replicate__aux:n #1
403 {
404   \int_incr:N \l__mmzx_tmpa_int
405   \str_case:nnF { #1 }
406   {
407     {c} { \__mmzx_expl_replicate__b: }
408     {e} { \__mmzx_expl_replicate__b: }
409     {o} { \__mmzx_expl_replicate__b: }
410     {p} { }
411     {n} { \__mmzx_expl_replicate__b: }
412     {v} { \__mmzx_expl_replicate__b: }
413     {D} { }
414     {F} { \__mmzx_expl_replicate__b: }
415     {N} { \__mmzx_expl_replicate__t: }
416     {T} { \__mmzx_expl_replicate__b: }
417     {V} { \__mmzx_expl_replicate__t: }
418     {w} { \__mmzx_expl_replicate__e: }
419   }{
420     \__mmzx_expl_replicate__e:
421   }
422 }

```

`__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.

```

423 \cs_new_nopar:Npn \__mmzx_expl_replicate__b:
424 {
425   \tl_put_right:Nn \l__mmzx_tmpc_tl {n}
426   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__bb_tl
427   \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
428   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__ba_tl
429 }
430 \cs_new_nopar:Npn \__mmzx_expl_replicate__t:
431 {
432   \tl_put_right:Nn \l__mmzx_tmpc_tl {N}
433   \tl_put_right:NV \l__mmzx_tmpa_tl \g__mmzx_expl_replicate__tb_tl

```

```

434 \tl_put_right:Ne \l__mmzx_tmpa_tl { \int_to_arabic:n { \l__mmzx_tmpa_int } }
435 }
436 \cs_new_nopar:Npn \__mmzx_expl_replicate__e:
437 {
438   \PackageError{mmzx}{No do, sorry. Use replicate with args instead.}{}
439 }

```

`__mmzx_expl_replicate_fn: (fn.)` For replicating `expl3` functions.

```

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

```

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

```

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

```

</sty>

memoize-ext-l3draw

Clea F. Rees

11978 2026-06-18

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>

```
473 \GetIdInfo $Id: memoize-ext-l3draw.dtx 11978 2026-06-25 01:01:43Z cfrees $ {Extensi
474 \!debug> \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
475 \!debug> {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
476 \!debug> \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
477 \!debug> {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
478 %
479 \!debug> \disable@package@load {memoize-ext-l3draw-debug}
480 \!debug> \disable@package@load {memoize-ext-l3draw}
481 { Only one of memoize-ext-l3draw and memoize-ext-l3draw-debug
482 should be loaded.
483 Since
484 \!debug> memoize-ext-l3draw
485 \!debug> memoize-ext-l3draw-debug
486 has been loaded,I will ignore your request for
487 \!debug> memoize-ext-l3draw
488 \!debug> memoize-ext-l3draw-debug
489 .}
490 %
491 \!debug> \RequirePackage{memoize-ext}
492 \!debug> \RequirePackage{memoize-ext-debug}
493 \!debug> \RequirePackage{memoize-ext-expl3}
494 \!debug> \RequirePackage{memoize-ext-expl3-debug}
```

l3draw

mmzx_draw_id_begin_int (*var.*) Temporary int.

```
495 \int_new:N \g__mmzx_draw_id_begin_int
```

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

```

496 \cs_new:Npn \__mmzx_advice_collect_draw_args:w #1 \draw_end:
497 {
498   \toks0={ #1 \draw_end: }
499   \exp_args:No \AdviceInnerHandler {\the\toks0}
500 }

```

`AdviceCollectDrawArguments` Public wrapper.

```

501 \cs_new:Npn \AdviceCollectDrawArguments{
502   \toks0 = {}
503   \__mmzx_advice_collect_draw_args:w
504 }

```

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

```

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

```

</sty>

memoize-ext-sockets

Clea F. Rees

11978 2026-06-18

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.

```
517 \cs_new_nopar:Npn \_mmzx_socket_assigned_plug:n #1
518 { % rhybudd: fn mewno1
519   \str_use:c { l__socket_#1_plug_str }
520 }
```

</sty>

memoize-ext-tag

Clea F. Rees

11978 2026-06-18

Abstract

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

Contents

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

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

<*sty> <@@=mmzx>

521 \GetIdInfo \$Id: memoize-ext-tag.dtx 11978 2026-06-25 01:01:43Z cfrees \$ {Extensions


```

522 <!debug>    \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
523 <!debug>    {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
524 <debug>    \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
525 <debug>    {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
526 %
527 <!debug>    \disable@package@load {memoize-ext-tag-debug}
528 <debug>    \disable@package@load {memoize-ext-tag}
529 { Only one of memoize-ext-tag and memoize-ext-tag-debug
530  should be loaded.
531  Since
532 <!debug>    memoize-ext-tag
533 <debug>    memoize-ext-tag-debug
534  has been loaded,I will ignore your request for
535 <debug>    memoize-ext-tag
536 <!debug>    memoize-ext-tag-debug
537 .}

538 <!debug>    \RequirePackage{memoize-ext}
539 <debug>    \RequirePackage{memoize-ext-debug}
540 %

```

We don't want inconsistent names in hooks.

```

541 \SetDefaultHookLabel{memoize-ext}
542 %
543 \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.) 544 \bool_new:N \l_mmzx_ok_bool
\g_mmzx_plug_orig_str (var.) 545 \int_new:N \g_mmzx_tagpic_int
\mmzxtagtoks (toks) 546 \str_new:N \g_mmzx_plug_orig_str
547 \tl_new:N \l_mmzx_toks_tl
548 \newtoks\mmzxtagtoks

```

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

```

\include/extern/after (socket) 549 \socket_new:nn {tagsupport/memoize/include/extern/before} {3}
550 \socket_new:nn {tagsupport/memoize/include/extern/after} {0}

```

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

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

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

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

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

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

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

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

```

551 \hook_gput_code:nnn {package/tikz/after} {..}
552 {
553   \pgfqkeys{/tikz}{
554     alt/.forward to=/mmz/alt,
555     actualtext/.forward to=/mmz/actualtext,
556     artifact/.forward to=/mmz/artifact,
557     tagging-setup/.forward to=/mmz/tagging-setup,
558   }
559   \mmzset{
560     alt/.code={
561       \pgfqkeys{/tikz/tagging-setup}{alt={#1}}
562       \bool_set_true:N \l__mmzx_ok_bool
563       \str_gset:Nn \g__mmzx_plug_orig_str {alt}
564       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
565       \socket_assign_plug:nn
566         {tagsupport/memoize/include/extern/before} {mmzx/alt}
567       \socket_assign_plug:nn
568         {tagsupport/memoize/include/extern/after} {mmzx/alt}
569 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
570     },
571     actualtext/.code={
572       \pgfqkeys{/tikz/tagging-setup}{actualtext={#1}}
573       \bool_set_true:N \l__mmzx_ok_bool
574       \str_gset:Nn \g__mmzx_plug_orig_str {actualtext}
575       \exp_args:Ne \mmzxtagtoks { \text_purify:n {#1} }
576       \socket_assign_plug:nn
577         {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
578       \socket_assign_plug:nn
579         {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
580 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
581     },
582     artifact/.code={
583       \pgfqkeys{/tikz/tagging-setup}{artifact}
584       \bool_set_true:N \l__mmzx_ok_bool
585       \str_gset:Nn \g__mmzx_plug_orig_str {artifact}
586       \mmzxtagtoks {}
587       \socket_assign_plug:nn
588         {tagsupport/memoize/include/extern/before} {mmzx/artifact}
589       \socket_assign_plug:nn
590         {tagsupport/memoize/include/extern/after} {mmzx/artifact}
591 <debug> \tl_log:e {\exp_args:No \exp_not:n {\the\mmzCCMemo}}
592     },

```

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

```

593   tagging-setup/.is choice,
594   tagging-setup/alt/.forward to=/mmz/alt,
595   tagging-setup/actualtext/.forward to=/mmz/actualtext,

```

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.

```

596 tagging-setup/artifact/.forward to=/mmz/artifact,
597 tagging-setup/.unknown/.code =
598 {

```

I guess this still does something useful?

```

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

```

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

The basic idea is this:

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

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

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

```

611 \appto\mmzAtBeginMemoization{
612 \gtoksapp\mmzCCMemo{
613 \let\mmzxIncludeExternOrig\mmzIncludeExtern
614 \let\mmzIncludeExtern\mmzxIncludeExtern
615 }
616 }

```

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

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

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

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

But does that lessen my guilt? :-)

```

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

```

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

```

632 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/alt}
633 {
634   <debug> \__mmzx_debug:n {Executing plug mmzx/alt in socket
635   <debug> tagsupport/memoize/include/extern/before.}
636   \mode_if_vertical:T
637   {
638     \if@inlabel
639       \mode_leave_vertical:
640     \else
641       \tag_socket_use:n {para/begin}
642     \fi
643   }
644   \tag_mc_end_push:
645   \tl_set:No \l__mmzx_toks_tl {\the\mmzxtagtoks}
646   \tag_struct_begin:n
647   {
648     tag=Figure,
649     alt=\l__mmzx_toks_tl,
650   }
651   \tag_mc_begin:n {tag=Figure}
652   \cs_new:cpe {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
653   {
654     \__mmzx_pgftikz_tag_bbox:ennn {mmzx-id\int_to_arabic:n {\g__mmzx_tagpic_int}}
655     {#1}{#2}{#3}
656   }
657   <debug> \cs_log:c {mmzx@tag@tikz@mark@pos@\int_to_arabic:n
658   <debug> {\g__mmzx_tagpic_int}}
659   \tag_struct_gput:ene
660   {\tag_get:n {struct_num}}
661   {attribute}
662   {
663     /O /Layout /BBox
664     [
665       \use:c
666       {mmzx@tag@tikz@mark@pos@\int_to_arabic:n {\g__mmzx_tagpic_int}}
667     ]

```

```

668 }
669 <debug>    \_mmzx_debug:e {Recording xpos,
670 <debug>      ypos of mmxc-id\int_to_arabic:n {\g_mmzx_tagpic_int}.}
671 \tex_savepos:D
672 \_mmzx_property_record_orig:ee
673 {mmzx-id\int_to_arabic:n {\g_mmzx_tagpic_int}}
674 {xpos,ypos}
675 \tex_savepos:D
676 }

```

»>

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 «<

```

677 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/alt}
678 {
679 <debug>    \_mmzx_debug:n {Executing plug mmzx/alt in socket
680 <debug>      tagsupport/memoize/include/extern/after.}
681 \tag_mc_end:
682 \tag_struct_end:
683 \tag_mc_begin_pop:n {}
684 }

```

»>

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

```

after mmzx/actualtext (plug)
685 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/actualtext}
686 {
687 <debug>    \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
688 <debug>      tagsupport/memoize/include/extern/before.}
689 \pgfqkeys{/tikz/tagging-setup}{actualtext/.expand once = {\the\mmzxtagtoks},}
690 \socket_use:n {tagsupport/tikz/picture/begin}
691 }
692 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/actualtext}
693 {
694 <debug>    \_mmzx_debug:n {Executing plug mmzx/actualtext in socket
695 <debug>      tagsupport/memoize/include/extern/after.}
696 \socket_use:n {tagsupport/tikz/picture/end}
697 }

```

»>

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

```

n/after mmzx/artifact (plug)
698 \socket_new_plug:nnn {tagsupport/memoize/include/extern/before} {mmzx/artifact}
699 {
700 <debug>    \_mmzx_debug:n {Executing plug mmzx/artifact in socket
701 <debug>      tagsupport/memoize/include/extern/before.}
702 \pgfqkeys{/tikz/tagging-setup}{artifact,}
703 \socket_use:n {tagsupport/tikz/picture/begin}
704 }
705 \socket_new_plug:nnn {tagsupport/memoize/include/extern/after} {mmzx/artifact}
706 {
707 <debug>    \_mmzx_debug:n {Executing plug mmzx/artifact in socket

```

```

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

```

»>

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

_pgftikz_tag_bbox:ennn (fn.)

```

711 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox:nnnn #1#2#3#4
712 {
713   \__mmzx_pgftikz_tag_bbox_aux:eennn
714   {
715     \mmzx_property_ref_orig:ee {#1}{xpos}
716   }
717   {
718     \mmzx_property_ref_orig:ee {#1}{ypos}
719   }
720   {#2}{#3}{#4}
721 }
722 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox:nnnn {ennn}

```

ikz_tag_bbox_aux:nnnnn (fn.) Auxiliary.

ikz_tag_bbox_aux:eennn (fn.)

```

723 \cs_new_nopar:Npn \__mmzx_pgftikz_tag_bbox_aux:nnnnn #1#2#3#4#5
724 {
725   \dim_to_decimal_in_bp:n {#1sp}
726   \c_space_tl
727   \dim_to_decimal_in_bp:n {#2sp-#5}
728   \c_space_tl
729   \dim_to_decimal_in_bp:n {#1sp+#3}
730   \c_space_tl
731   \dim_to_decimal_in_bp:n {#2sp+#4+#5}
732 }
733 \cs_generate_variant:Nn \__mmzx_pgftikz_tag_bbox_aux:nnnnn {eennn}

734 \hook_gput_code_with_args:nnn {cmd/tikz@picture/before} {mmzx}
735 {
736   \tag_if_active:T
737   {
738     <debug>      \__mmzx_debug:n {Executing mmzx code in hook cmd/tikz@picture/before.}
739     \socket_assign_plug:nn {tagsupport/tikz/picture/init} {mmzx}
740   }
741 }

```

»>

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](#)).

```

742 \socket_new_plug:nnn {tagsupport/tikz/picture/init} {mmzx}
743 {
744   <debug>      \__mmzx_debug:n {Executing plug mmzx in socket
745   <debug>      tagsupport/tikz/picture/init.}
746   \bool_set_false:N \l__mmzx_ok_bool
747   \legacy_if:nT {memoizing}
748   {
749     <debug>      \__mmzx_debug:n {Arg to tagsupport/tikz/picture/init is #1.}
750     \pgfkeyssavekeyfilterstateto\mmzx@temp@tagging@saved@filterstate
751     \pgfkeysinstallkeyfilter{/pgf/key filters/active families or no family}
752     {/pgf/key filters/false}/pgf/key filters/false}}
753     \pgfqkeysactivatesinglefamilyandfilteroptions{/tikz/tagging-setup}{/tikz}{#1}
754
755   <debug>      \__mmzx_debug:e { Restoring filter state: \exp_not:V
756   <debug>      \mmzx@temp@tagging@saved@filterstate }
757     \mmzx@temp@tagging@saved@filterstate
758     \bool_if:NF \l__mmzx_ok_bool
759     {
760   <debug>      \__mmzx_debug:n {
761   <debug>      No plug set for utilisation. Trying to match latex-lab plug.
762   <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.

```

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

```

```

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

811 \hook_gput_code:nnn {cmd/mmzAbort/after} {.}
812 {
813     \legacy_if:nT {memoizing} {
814         \PackageWarning{memoize-ext}{
815             Memoization has been aborted. If something like a reference needs
816             resolving, just compile again. If the content is unmemoizable --
817             e.g. contains remember picture, a breakable tcolorbox or suchlike,
818             you must mark the content unmemoizable or the tagging will be
819             incorrect. You may do this automatically or manually. Aborted
820         }
821     }
822 }

»>

```

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

_socket_plug_record:nn (fn.)

_socket_plug_record:ee (fn.)

ag_socket_plug_record: (fn.)

```

823 \cs_new_protected:Npn \__mmzx_tag_socket_plug_record:nn #1#2
824 {
825     \xtoksapp\mmzCCMemo{
826         \exp_not:N \mmzxtagtoks
827         =
828         \c_left_brace_str
829         \the\mmzxtagtoks
830         \c_right_brace_str
831         \exp_not:N \AssignSocketPlug
832         \c_left_brace_str
833         tagsupport/memoize/include/extern/before
834         \c_right_brace_str

```



```

835 \c_left_brace_str
836 #1
837 \c_right_brace_str
838 \exp_not:N \AssignSocketPlug
839 \c_left_brace_str
840 tagsupport/memoize/include/extern/after
841 \c_right_brace_str
842 \c_left_brace_str
843 #2
844 \c_right_brace_str
845 }
846 }
847 \cs_generate_variant:Nn \__mmzx_tag_socket_plug_record:nn {ee}
848 \cs_new_protected:Npn \__mmzx_tag_socket_plug_record:
849 {
850 \__mmzx_tag_socket_plug_record:ee
851 {\__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/before}}
852 {\__mmzx_socket_assigned_plug:n {tagsupport/memoize/include/extern/after}}
853 }
854 \cs_new_protected:Npn \__mmzx_tag_socket_plug_record:nnn #1#2#3
855 {
856 \mmzxtagtoks { \text_purify:n {#3} }
857 \__mmzx_tag_socket_plug_record:nn {#1} {#2}
858 }

»>

```

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

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

```

859 \cs_new_protected_nopar:Npn \mmzx_tag_get_recorded:N #1
860 {
861 \tl_set:No #1 {\the\mmzxtagtoks}
862 }

»>

```

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

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

```

863 \cs_new_eq:NN \mmzx_tag_socket_plug_record:nnn \__mmzx_tag_socket_plug_record:nnn
864 \cs_new_eq:NN \mmzx_tag_socket_plug_record:nn \__mmzx_tag_socket_plug_record:nn
865 \cs_new_eq:NN \mmzx_tag_socket_plug_record: \__mmzx_tag_socket_plug_record:

```

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

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

```

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

```

```

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

```

»>

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

```

880 \cs_if_free:NT \mmzx_property_ref_orig:nn
881 {
882   \cs_new_eq:NN \mmzx_property_ref_orig:nn \property_ref:nn
883   \cs_generate_variant:Nn \mmzx_property_ref_orig:nn {ee}
884 }

```

»>

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

```

885 \cs_if_free:NT \_mmzx_property_record_orig:nn
886 {
887   \cs_new_eq:NN \_mmzx_property_record_orig:nn \property_record:nn
888   \cs_generate_variant:Nn \_mmzx_property_record_orig:nn {ee}
889 }

```

»>

890 }

</sty>

memoize-ext-talk

Clea F. Rees

11978 2026-06-18

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>

```
891 \GetIdInfo $Id: memoize-ext-talk.dtx 11978 2026-06-25 01:01:43Z cfrees $ {Extension
892 \!debug} \ProvidesExplPackage{\ExplFileName}{\ExplFileDate}
893 \!debug} {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
894 \debug} \ProvidesExplPackage{\ExplFileName-debug}{\ExplFileDate}
895 \debug} {v0.4.3 \ExplFileVersion}{\ExplFileDescription}
896 %
897 \!debug} \disable@package@load {memoize-ext-talk-debug}
898 \debug} \disable@package@load {memoize-ext-talk}
899 { Only one of memoize-ext-talk and memoize-ext-talk-debug
900 should be loaded.
901 Since
902 \!debug} memoize-ext-talk
903 \debug} memoize-ext-talk-debug
904 has been loaded,I will ignore your request for
905 \debug} memoize-ext-talk
906 \!debug} memoize-ext-talk-debug
907 .}
```

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?

```
908 \!debug} \RequirePackage{memoize-ext-tag}
909 \debug} \RequirePackage{memoize-ext-tag-debug}
```

We don't want inconsistent names in hooks.

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

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

```
911 \hook_gput_code:nnn {class/ltx-talk/after}{.}
912 {
913   <debug>      \__mmzx_debug:n {Enabling ltx-talk support.}
914   \mmzset{
915     per overlay/.code={},
916     talk mode to prefix/.style={
917       prefix=\mmz@prefix@dir\mmz@prefix@name\l__mmzx_talk_mode_str -,
918     },
919   }
```

```
mmzx_talk_opacity_seq (var.)
```

```
alk_saved_opacity_seq (var.)
```

```
_mmzx_talk_opacity_tl (var.)
```

```
920 \seq_new:N \g__mmzx_talk_opacity_seq
921 \seq_new:N \g__mmzx_talk_saved_opacity_seq
922 \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
```

```
923
924 \cs_new_eq:NN \g__mmzx_talk_frame_int \c@frame
925 \cs_new_eq:NN \g__mmzx_talk_slide_int \c@slide
926 \cs_new_eq:NN \g__mmzx_talk_pauses_int \c@pauses
927 \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)

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

Initialise sequence.

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

```
mmzx_talk_opacity_save: (fn.)
```

```
930 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_save:
931 {
932   \seq_gset_eq:NN \g__mmzx_talk_opacity_saved_seq \g__mmzx_talk_opacity_seq
933   \seq_get:NN \g__mmzx_talk_opacity_seq \l__mmzx_talk_opacity_tl
934   \exp_args:NV \tl_if_eq:NNTF \l__mmzx_talk_opacity_tl \q_no_value
935   {
936     \seq_gpush:Nn \g__mmzx_talk_opacity_saved_seq {1}
937     \opacity_select:n {1}
938   } {
939     \seq_gpush:NV \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
940     \opacity_select:V \l__mmzx_talk_opacity_tl
941   }
942 }
```

_talk_opacity_restore: (fn.)

```

943 \cs_new_protected_nopar:Npn \__mmzx_talk_opacity_restore:
944 {
945   \seq_gpop:NN \g__mmzx_talk_opacity_saved_seq \l__mmzx_talk_opacity_tl
946   \opacity_select:V \l__mmzx_talk_opacity_tl
947   \seq_gset_eq:NN \g__mmzx_talk_opacity_seq \g__mmzx_talk_opacity_saved_seq
948 }

```

`\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 ...

```

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

```

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

```

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

```

```

990      {
991        \socket_assign_plug:nn {mmzx/talk/memoization/begin}{peroverlay}
992        \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.

```

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

```

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

```

998      /mmz/per overlay/.code={},
999    },
1000  }

```

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

alk/memoization/end (*socket*)

```

1001  \socket_new:nn {mmzx/talk/memoization/begin}{0}
1002  \socket_new:nn {mmzx/talk/memoization/end}{0}

```

tion/begin peroverlay (*plug*) Plugs.

zation/end peroverlay (*plug*)

```

1003  \socket_new_plug:nnn {mmzx/talk/memoization/begin}{peroverlay}
1004  {
1005  <debug>    \__mmzx_debug:n {Executing plug peroverlay in socket
1006  <debug>      mmzx/talk/memoization/begin.}
1007    \xdef\mmzxTalkPauses{
1008      \int_to_arabic:n {\g__mmzx_talk_pauses_int}
1009    }
1010    \xtoksapp\mmzCMemo{
1011      \noexpand\mmzxSetTalkOverlays{\mmzxTalkPauses}{
1012        \int_to_arabic:n {\g__mmzx_talk_slide_int}
1013      }
1014    }
1015    \gtoksapp\mmzCCMemo{
1016      \only<all:\mmzxTalkOverlays>{}
1017    }
1018    \seq_get:NNTF \g__mmzx_talk_opacity_seq \l__mmzx_opacity_tl
1019    {
1020      \fp_gset:NV \l__mmzx_tmpa_fp \l__mmzx_opacity_tl
1021    } {
1022      \fp_gset:Nn \l__mmzx_tmpa_fp {1}
1023    }
1024    \xtoksapp\mmzContextExtra{
1025      opacity=\fp_to_decimal:N \l__mmzx_tmpa_fp
1026    }
1027  }
1028  \socket_new_plug:nnn {mmzx/talk/memoization/end}{peroverlay}
1029  {
1030  <debug>    \__mmzx_debug:n {Executing plug peroverlay in socket
1031  <debug>      mmzx/talk/memoization/end.}
1032    \xtoksapp\mmzCCMemo{
1033      \exp_not:N \setcounter{pauses}

```

```

1034     {\int_to_arabic:n {\g__mmzx_talk_pauses_int}}
1035   }
1036 }

```

`\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.

```

1037 \cs_new_nopar:Npn \mmzxSetTalkOverlays#1#2{
1038 <debug> \__mmzx_debug:e {Executing \cs_to_str:N \mmzxSetTalkOverlays}
1039 \int_compare:nNnTF {\g__mmzx_talk_pauses_int} = {#1}
1040 {
1041   \gdef\mmzxTalkOverlays{#2}
1042   \int_compare:nNnTF {\g__mmzx_talk_slide_int} < {#2}
1043   {
1044     \bool_set_true:N \l__mmzx_tmpa_bool
1045   }{
1046     \bool_set_false:N \l__mmzx_tmpa_bool
1047   }
1048 }{
1049   \bool_set_true:N \l__mmzx_tmpa_bool
1050 }
1051 \bool_if:NT \l__mmzx_tmpa_bool
1052 {
1053   \bool_gset_true:N \g__talk_slide_continue_bool
1054   \appto\mmzAtBeginMemoization{
1055     \gtoksapp\mmzCMemo{\mmzxSetTalkOverlays{#1}{#2}}
1056   }
1057 }
1058 }

1059 }

</sty>

```

References

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

Change History

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

<code>\l__mmzx_opt_expl_bool</code> : Fix for GitHub latex3/latex2e/issues/2110. <small>L^AT_EX 2_ε issue</small> #2110.	10	vo.4.3	<code>\l__mmzx_opt_expl_bool</code> : Don't time-limit redefinition of <code>\DeclareUnknownKeyHandler</code>	10
Fix non-recognition of memoize pkg options in memoize-ext-debug.	9			

Index

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

Symbols	
<code>\@currname</code>	63
<code>\@ifl@t@r</code>	5, 78
<code>\@ifundefined</code>	2
<code>\@mmzx@kernel@after@shipout@background</code>	107
<code>__keys_options_expand_module:Nn</code>	66
<code>__mmzx_advice_collect_draw_args:w</code> (fn.)	496, 503
<code>__mmzx_cctab_end</code> : (fn.)	298
<code>__mmzx_cctab_stop</code> : (fn.)	299, 343, 456
<code>__mmzx_debug:N</code>	136
<code>__mmzx_debug:e</code>	138, 669, 755, 787, 1038
<code>__mmzx_debug:n</code>	131, 135, 174, 183, 224, 227, 235, 266, 323, 329, 333, 622, 625, 627, 634, 679, 687, 694, 700, 707, 738, 744, 749, 760, 773, 779, 783, 803, 913, 967, 1005, 1030
<code>__mmzx_expl_at_begin</code> : (fn.)	299
<code>__mmzx_expl_at_start</code> : (fn.)	284, 299
<code>__mmzx_expl_replicate_aux:n</code> (fn.)	375
<code>__mmzx_expl_replicate_b</code> : (fn.)	407, 408, 409, 411, 412, 414, 416, 423
<code>__mmzx_expl_replicate_e</code> : (fn.)	418, 420, 423
<code>__mmzx_expl_replicate_t</code> : (fn.)	415, 417, 423
<code>__mmzx_expl_replicate_fn</code> : (fn.)	440
<code>__mmzx_expl_replicate_fn_aux:nnN</code> (fn.)	375, 463
<code>__mmzx_if_replicating</code> :	221
<code>__mmzx_if_replicating:F</code>	233
<code>__mmzx_if_replicating:TF</code> (fn.)	221
<code>__mmzx_if_replicating_p</code> : (fn.)	221
<code>__mmzx_include_extern:nNnnnnnnn</code> (fn.)	617
<code>__mmzx_nexpl_at_begin</code> : (fn.)	299
<code>__mmzx_nexpl_at_start</code> : (fn.)	299, 447
<code>__mmzx_noop</code> :	141, 440, 459
<code>__mmzx_noop:n</code>	141
<code>__mmzx_noop:nNnnnnnnn</code> (fn.)	617
<code>__mmzx_pgftikz_tag_bbox:ennn</code> (fn.)	654, 711
<code>__mmzx_pgftikz_tag_bbox:nnnn</code> (fn.)	711
<code>__mmzx_pgftikz_tag_bbox_aux:ennn</code> (fn.)	713, 723
<code>__mmzx_pgftikz_tag_bbox_aux:nnnn</code> (fn.)	723
<code>__mmzx_saved_mmzxExplAtBegin</code> : (fn.)	257, 282, 292
<code>__mmzx_saved_mmzxExplAtEnd</code> : (fn.)	257, 283, 293
<code>__mmzx_socket_assigned_plug:n</code> (fn.)	517, 764, 767, 851, 852, 989
<code>__mmzx_tag_socket_plug_record</code> : (fn.)	802, 823, 865
<code>__mmzx_tag_socket_plug_record:ee</code> (fn.)	823
<code>__mmzx_tag_socket_plug_record:nn</code> (fn.)	823, 864
<code>__mmzx_tag_socket_plug_record:nnn</code> (fn.)	823, 863
<code>__mmzx_talk_opacity_restore</code> : (fn.)	943, 954
<code>__mmzx_talk_opacity_save</code> : (fn.)	930, 952
<code>__mmzx_tmp_cctab_stop</code> : (fn.)	392, 440
A	
<code>\AdviceCollectDrawArguments</code>	501, 514
<code>\AdviceInnerHandler</code>	499
<code>\AdviceOriginal</code>	390
<code>\AdviceReplaced</code>	464
<code>\AdviceRunIfNotReplicating</code>	231, 241
<code>\AdviceRuntrue</code>	236
<code>\appto</code>	322, 611, 1054
<code>\AssignSocketPlug</code>	831, 838
<code>auto/run if not replicating</code> (pgfkey)	239
B	
<code>\bool_gset_true:N</code>	1053
<code>\bool_if:NF</code>	270, 758
<code>\bool_if:NT</code>	143, 164, 171, 180, 800, 1051
<code>\bool_new:N</code>	37, 120, 219, 256, 544
<code>\bool_set_false:N</code>	40, 220, 262, 267, 291, 443, 746, 786, 1046
<code>\bool_set_true:N</code>	39, 272, 446, 462, 562, 573, 584, 1044, 1049

\box	958	\draw_end:	496, 498
C		E	
\c@frame	924	\else	275, 640, 983
\c@pauses	926	\else:	226
\c@slide	925	\endcsname	327, 337, 449, 454, 980, 984
\c_mmzx_expl_at_cctab	244, 301	\endinput	13
\c_mmzx_nexpl_at_cctab	248, 305	\etoksapp	99
\c_code_cctab	245, 249	\exp_after:wN	388
\c_left_brace_str ..	371, 828, 832, 835, 839, 842	\exp_args:Ne	564, 575, 770, 776
\c_right_brace_str ..	373, 830, 834, 837, 841, 844	\exp_args:Nno	599
\c_space_tl	726, 728, 730	\exp_args:No	329, 499, 569, 580, 591
\cctab_begin:N	301, 305	\exp_args:NV	69, 138, 464, 934
\cctab_const:Nn	244, 248	\exp_last_unbraced:Ne	463
\cctab_end:	298	\exp_not:c	70
\cctab_select:N	245, 249	\exp_not:N	69, 71, 327, 337, 388, 390, 391, 449, 454, 826, 831, 838, 1033
\char_set_catcode_active:n	254	\exp_not:n	138, 569, 580, 591
\char_set_catcode_space:n	252, 253	\exp_not:V	755, 771, 777
\clist_map_inline:nn	99	\expandafter	103, 984
\cs:w	388	\ExpandArgs	89
\cs_end:	388	\expandonce	390
\cs_generate_variant:Nn	135, 543, 722, 733, 847, 883, 888, 928	expl3 (opt.)	3
\cs_gset_protected:ce	385	expl3 functions:	
\cs_if_exist:cF	377, 383	_mmzx_advice_collect_draw_args:w ...	496, 503
\cs_if_exist_use:c	90	_mmzx_cctab_end:	298
\cs_if_free:NT	880, 885	_mmzx_cctab_stop:	299, 343, 456
\cs_log:c	657	_mmzx_expl_at_begin:	299
\cs_log:N	331, 340	_mmzx_expl_at_start:	284, 299
\cs_new:cpe	652	_mmzx_expl_replicate_aux:n	375
\cs_new:Npn ...	141, 231, 375, 402, 441, 496, 501	_mmzx_expl_replicate_b:	407, 408, 409, 411, 412, 414, 416, 423
\cs_new_eq:cc	397	_mmzx_expl_replicate_e: ..	418, 420, 423
\cs_new_eq:NN ..	142, 298, 440, 466, 630, 631, 863, 864, 865, 882, 887, 924, 925, 926, 927	_mmzx_expl_replicate_t: ..	415, 417, 423
\cs_new_nopar:Npn ..	307, 312, 423, 430, 436, 517, 711, 723, 1037	_mmzx_expl_replicate_fn:	440
\cs_new_protected:Npn ..	131, 136, 823, 848, 854	_mmzx_expl_replicate_fn_aux:nnN	375, 463
\cs_new_protected_nopar:Npn ..	107, 257, 258, 259, 299, 303, 317, 617, 619, 859, 930, 943	_mmzx_if_replicating:TF	221
\cs_set:Npn	451	_mmzx_if_replicating_p:	221
\cs_set_eq:NN	282, 283, 292, 293, 459	_mmzx_include_extern:nNnnnnnnn ...	617
\cs_set_nopar:Npn ..	309, 310, 314, 315, 319, 320	_mmzx_nexpl_at_begin:	299
\cs_set_protected:cpn	65	_mmzx_nexpl_at_start:	299, 447
\cs_set_protected_nopar:Npn	276	_mmzx_noop:nNnnnnnnnn	617
\cs_split_function:N	464	_mmzx_pgftikz_tag_bbox:ennn ...	654, 711
\cs_to_str:N	138, 1038	_mmzx_pgftikz_tag_bbox:nnnn	711
\csname	327, 337, 449, 454, 980, 984	_mmzx_pgftikz_tag_bbox_aux:eennn	713, 723
\CurrentOption	76	_mmzx_pgftikz_tag_bbox_aux:nnnnn ..	723
D		_mmzx_property_record_orig:nn ...	885
\DeclareDocumentCommand	63	_mmzx_restore_ccmemo_input: ..	257, 276, 294
\DeclareUnknownKeyHandler	63, 75	_mmzx_saved_mmzxExplAtBegin:	257, 282, 292
\def	103, 949	_mmzx_saved_mmzxExplAtEnd: ..	257, 283, 293
\dim_to_decimal_in_bp:n	725, 727, 729, 731	_mmzx_socket_assigned_plug:n	517, 764, 767, 851, 852, 989
\disable@package@load	25, 26, 204, 205, 351, 352, 479, 480, 527, 528, 897, 898		

_mmzx_tag_socket_plug_record:	\fp_gset:NV 1020
. 802, 823, 865	\fp_new:N 121
_mmzx_tag_socket_plug_record:ee . . . 823	\fp_to_decimal:N 1025
_mmzx_tag_socket_plug_record:nn 823, 864	
_mmzx_tag_socket_plug_record:nnn . . .	G
. 823, 863	\g_mmzx_draw_id_begin_int (var) 495
_mmzx_talk_opacity_restore: . . . 943, 954	\g_mmzx_expl_replicate__ba_tl (var) 368, 428
_mmzx_talk_opacity_save: 930, 952	\g_mmzx_expl_replicate__bb_tl (var) 368, 426
_mmzx_tmp_cctab_stop: 392, 440	\g_mmzx_expl_replicate__tb_tl (var) 368, 433
\mmzx_property_ref_orig:nn 880	\g_mmzx_name_str
\mmzx_tag_get_recorded:N 8, 859 22, 23, 145, 146, 166, 167, 173, 175
\mmzx_tag_socket_plug_record: 8, 863	\g_mmzx_plug_orig_str (var) 544, 563, 574, 585
\mmzx_tag_socket_plug_record:nn . . . 7, 863	\g_mmzx_tagpic_int (var)
\mmzx_tag_socket_plug_record:nnn . . . 7, 863 544, 621, 652, 654, 658, 666, 670, 673
\opacity_select:V 928, 940, 946	\g_mmzx_talk_frame_int 923
expl3 variables:	\g_mmzx_talk_opacity_saved_seq
\g_mmzx_draw_id_begin_int 495 932, 936, 939, 945, 947
\g_mmzx_expl_replicate__ba_tl . . . 368, 428	\g_mmzx_talk_opacity_seq (var) 920,
\g_mmzx_expl_replicate__bb_tl . . . 368, 426 929, 932, 933, 947, 963, 968, 969, 1018
\g_mmzx_expl_replicate__tb_tl . . . 368, 433	\g_mmzx_talk_pauses_int 923, 1008, 1034, 1039
\g_mmzx_plug_orig_str . . . 544, 563, 574, 585	\g_mmzx_talk_saved_opacity_seq (var) . . . 920
\g_mmzx_tagpic_int	\g_mmzx_talk_slide_int 923, 1012, 1042
. 544, 621, 652, 654, 658, 666, 670, 673	\g_talk_slide_continue_bool 1053
\g_mmzx_talk_opacity_seq 920,	\gdef 1041
. 929, 932, 933, 947, 963, 968, 969, 1018	\GetIdInfo 16, 198, 345, 473, 521, 891
\g_mmzx_talk_saved_opacity_seq 920	\group_begin: 461
\l_mmzx_expl_bool	\group_end: 391
. 256, 262, 267, 270, 272, 291	\gtoksapp 99, 508, 612, 1015, 1055
\l_mmzx_ok_bool	
. 544, 562, 573, 584, 746, 758, 786, 800	H
\l_mmzx_opt_draw_bool 41, 171	\hook_gput_code:nnn
\l_mmzx_opt_expl_bool 41, 164 113, 114, 147, 149, 169, 178, 260,
\l_mmzx_opt_tag_bool 37, 56, 143 264, 268, 287, 289, 341, 551, 811, 876, 911, 965
\l_mmzx_replicating_bool 219, 223, 462	\hook_gput_code_with_args:nnn 734, 961
\l_mmzx_talk_opacity_tl	\hook_gset_rule:nnnn 866,
. 920, 933, 934, 939, 940, 945, 946 867, 868, 869, 870, 871, 872, 873, 874, 960
\l_mmzx_toks_tl 544, 645, 649	\hook_log:n 875, 878, 879
\ExplFileDate 17, 20, 199,	
. 201, 346, 348, 474, 476, 522, 524, 892, 894	I
\ExplFileDescription 18, 20, 200,	\if@inlabel 638
. 202, 347, 349, 475, 477, 523, 525, 893, 895	\if_bool:N 223
\ExplFileName 17, 19, 23, 199,	\IfFormatAtLeastT 62
. 201, 346, 348, 474, 476, 522, 524, 892, 894	\IfFormatAtLeastTF 78, 79, 86
\ExplFileVersion 18, 20, 200,	\ifmemoizing 236, 981
. 202, 347, 349, 475, 477, 523, 525, 893, 895	\ifmmz@direct@ccmemo@input 273
\ExplLoaderFileDate 5	\IfPackageLoadedF 151, 153
	\int_compare:nNnTF 1039, 1042
F	\int_gincr:N 621
\fi 236, 280, 642, 985	\int_incr:N 404
\fi: 229	\int_new:N 122, 495, 545
\FirstAidNeededT 111	\int_set:Nn 251
\fmtversion 78	\int_to_arabic:n 427, 434,
\fp_gset:Nn 1022 652, 654, 657, 666, 670, 673, 1008, 1012, 1034
	\int_zero:N 379
	\iow_log:n 133

K	
<code>\keys_define:ne</code>	66
<code>\keys_define:nn</code>	41
L	
<code>l3draw (opt.)</code>	3
<code>\l_mmzx_expl_bool (var)</code>	256, 262, 267, 270, 272, 291
<code>\l_mmzx_ok_bool (var)</code>	544, 562, 573, 584, 746, 758, 786, 800
<code>\l_mmzx_opacity_tl</code>	1018, 1020
<code>\l_mmzx_opt_draw_bool (var)</code>	41, 171
<code>\l_mmzx_opt_expl_bool (var)</code>	41, 164
<code>\l_mmzx_opt_tag_bool (var)</code>	37, 56, 143
<code>\l_mmzx_opt_talk_bool</code>	58, 180
<code>\l_mmzx_replicating_bool (var)</code> ..	219, 223, 462
<code>\l_mmzx_talk_mode_str</code>	917, 923
<code>\l_mmzx_talk_opacity_tl (var)</code>	920, 933, 934, 939, 940, 945, 946
<code>\l_mmzx_tmpa_bool</code>	120, 443, 446, 1044, 1046, 1049, 1051
<code>\l_mmzx_tmpa_fp</code>	121, 1020, 1022, 1025
<code>\l_mmzx_tmpa_int</code>	122, 379, 404, 427, 434
<code>\l_mmzx_tmpa_seq</code>	127
<code>\l_mmzx_tmpa_str</code>	128
<code>\l_mmzx_tmpa_tl</code>	124, 380, 388, 390, 426, 427, 428, 433, 434
<code>\l_mmzx_tmpb_str</code>	129
<code>\l_mmzx_tmpb_tl</code>	125
<code>\l_mmzx_tmpc_tl</code>	126, 381, 383, 385, 395, 397, 425, 432
<code>\l_mmzx_toks_tl (var)</code>	544, 645, 649
<code>\l_talk_mode_str</code>	927
<code>\l_tikz_tagging_actualtext_tl</code>	777
<code>\l_tikz_tagging_alt_tl</code>	771
<code>\l_keys_key_str</code>	71
<code>\l_tmpa_tl</code>	968
<code>\legacy_if:nT</code>	747, 813, 993
<code>\let</code>	613, 614
<code>\long</code>	949
M	
<code>\makeatletter</code>	246, 250
<code>\MessageBreak</code>	10
<code>mmz/auto/replicate expl fn (pgfkey)</code>	467
<code>mmz/auto/replicate expl var (pgfkey)</code>	467
<code>\mmz@box</code>	951, 956, 958
<code>\mmz@capture</code>	951
<code>\mmz@direct@ccmemo@inputfalse</code>	278
<code>\mmz@direct@ccmemo@inputtrue</code>	281
<code>\mmz@maybe@quitvmode</code>	950, 958
<code>\mmz@prefix@dir</code>	917
<code>\mmz@prefix@name</code>	917
<code>\mmz@temptoks</code>	956, 957
<code>\mmzAtBeginMemoization</code>	322, 324, 331, 611, 1054
<code>\mmzAtEndMemoization</code>	332, 334, 340
<code>\mmzCCMemo</code>	325, 329, 335, 387, 448, 453, 508, 569, 580, 591, 612, 825, 950, 957, 1015, 1032
<code>\mmzCMemo</code>	1010, 1055
<code>\mmzContextExtra</code>	763, 1024
<code>\mmzExternalizeBox</code>	956
<code>\mmzIncludeExtern</code>	613, 614
<code>\mmzSalt</code>	187
<code>\mmzset</code>	93, 239, 467, 505, 559, 770, 776, 782, 914, 971
<code>\mmzSingleExternDriver</code>	949
<code>\mmzUnmemoizable</code>	795
<code>mmzx (plug)</code>	6
<code>mmzx/actualtext (plug)</code>	6
<code>mmzx/alt (plug)</code>	6
<code>mmzx/artifact (plug)</code>	6
<code>mmzx/talk/memoization/begin (socket)</code>	1001
<code>mmzx/talk/memoization/begin peroverlay (plug)</code>	1003
<code>mmzx/talk/memoization/end (socket)</code>	1001
<code>mmzx/talk/memoization/end peroverlay (plug)</code>	1003
<code>\mmzx@expl@replicate@fn</code>	440, 470
<code>\mmzx@temp@tagging@saved@filterstate</code>	750, 756, 757
<code>\mmzx_property_ref_orig:ee</code>	715, 718
<code>\mmzx_property_ref_orig:nn (fn.)</code>	880
<code>\mmzx_tag_get_recorded:N (fn.)</code>	8, 859
<code>\mmzx_tag_socket_plug_record: (fn.)</code> ...	8, 863
<code>\mmzx_tag_socket_plug_record:nn (fn.)</code> ..	7, 863
<code>\mmzx_tag_socket_plug_record:nnn (fn.)</code> ..	7, 863
<code>\mmzxExplAtBegin</code>	282, 292, 309, 314, 319, 327, 449
<code>\mmzxExplAtEnd</code>	283, 293, 310, 315, 320, 337, 444, 454
<code>\mmzxIncludeExtern</code>	614, 617
<code>\mmzxIncludeExternOrig</code>	613, 617
<code>\mmzxSetTalkOverlays</code>	1011, 1037
<code>\mmzxtagtoks (toks)</code>	544, 564, 575, 586, 645, 689, 826, 829, 856, 861
<code>\mmzxTalkOverlays</code>	1016, 1041
<code>\mmzxTalkPauses</code>	982, 1007, 1011
<code>\mode_if_vertical:T</code>	636
<code>\mode_leave_vertical:</code>	639
<code>\msg_line_context:</code>	193
<code>\msg_new:nnnn</code>	190
<code>\msg_warning:nnnnnn</code>	155
<code>\msg_warning_text:n</code>	192
N	
<code>\NeedsTeXFormat</code>	1
<code>\newtoks</code>	548
<code>\noexpand</code>	115, 1011
O	
<code>\only</code>	1016

<code>\opacity_select:n</code>	928, 937	<code>\protected</code>	101, 103
<code>\opacity_select:V</code> (fn.)	928, 940, 946	<code>\providecommand</code>	78, 89
options:		<code>\ProvidesExplPackage</code>	17, 19, 199, 201, 346, 348, 474, 476, 522, 524, 892, 894
<code>expl3</code>	3	Q	
<code>l3draw</code>	3	<code>\q_mmzx_stop</code>	123
<code>tag</code>	3	<code>\q_no_value</code>	934
<code>talk</code>	3	<code>\quark_new:N</code>	123
P			
<code>\PackageError</code>	7, 438	R	
<code>\PackageWarning</code>	45, 790, 814	<code>\relax</code>	274
<code>\PassOptionsToPackage</code>	76	<code>replicate expl fn (pgfkey)</code>	4
<code>per overlay (pgfkey)</code>	4	<code>\RequirePackage</code> 3, 83, 88, 92, 145, 146, 166, 167, 173, 175, 182, 184, 215, 216, 362, 363, 364, 365, 491, 492, 493, 494, 538, 539, 908, 909	
pgfkeys:		S	
<code>auto/run if not replicating</code>	239	<code>\seq_get:NN</code>	933
<code>mmz/auto/replicate expl fn</code>	467	<code>\seq_get:NNTF</code>	1018
<code>mmz/auto/replicate expl var</code>	467	<code>\seq_gpop:NN</code>	945, 968
<code>per overlay</code>	4	<code>\seq_gpush:Nn</code>	929, 936, 963
<code>replicate expl fn</code>	4	<code>\seq_gpush:NV</code>	939
<code>talk mode to prefix</code>	4	<code>\seq_gset_eq:NN</code>	932, 947
<code>\pgfkeyscurrentname</code>	600	<code>\seq_log:N</code>	969
<code>\pgfkeysinstallkeyfilter</code>	751	<code>\seq_new:N</code>	127, 920, 921
<code>\pgfkeyssavekeyfilterstateto</code>	750	<code>\setbox</code>	951
<code>\pgfqkeys</code> 553, 561, 572, 583, 600, 689, 702		<code>\setcounter</code>	1033
<code>\pgfqkeysactivatesinglefamilyandfilteroptions</code>	753	<code>\SetDefaultHookLabel</code> 36, 218, 367, 541, 910	
plugins:		<code>\socket_assign_plug:nn</code>	543, 565, 567, 576, 578, 587, 589, 739, 805, 806, 807, 808, 991, 992
<code>mmzx</code>	6	<code>\socket_new:nn</code>	549, 550, 1001, 1002
<code>mmzx/actualtext</code>	6	<code>\socket_new_plug:nnn</code>	632, 677, 685, 692, 698, 705, 742, 1003, 1028
<code>mmzx/alt</code>	6	<code>\socket_use:n</code>	628, 690, 696, 703, 709, 973, 976, 994
<code>mmzx/artifact</code>	6	<code>\socket_use:nnnn</code>	623
<code>mmzx/talk/memoization/begin peroverlay</code>	1003	sockets:	
<code>mmzx/talk/memoization/end peroverlay</code>	1003	<code>mmzx/talk/memoization/begin</code>	1001
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/actualtext</code>	685	<code>mmzx/talk/memoization/end</code>	1001
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/alt</code>	677	<code>tagssupport/memoize/include/extern/after</code>	7, 549
<code>tagssupport/memoize/include/extern/after</code> <code>mmzx/artifact</code>	698	<code>tagssupport/memoize/include/extern/before</code>	7, 549
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/actualtext</code>	685	<code>\str_case:nnF</code>	405
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/alt</code>	632	<code>\str_case_e:nn</code>	766
<code>tagssupport/memoize/include/extern/before</code> <code>mmzx/artifact</code>	698	<code>\str_gset:Nn</code>	563, 574, 585
<code>tagssupport/tikz/picture/init mmzx</code> ...	742	<code>\str_gset:NV</code>	23
<code>\preto</code>	332	<code>\str_if_eq:eeF</code>	395, 988
<code>\prg_new_conditional:Npnn</code>	221	<code>\str_if_eq:eeT</code>	444
<code>\prg_return_false:</code>	228	<code>\str_new:N</code>	22, 128, 129, 546
<code>\prg_return_true:</code>	225	<code>\str_use:c</code>	519
<code>\ProcessKeyOptions</code>	81	<code>\string</code>	324, 334
<code>\ProcessKeysOptions</code>	84	<code>\sys_if_engine luatex:F</code>	97
<code>\property_record:nn</code>	887	<code>\sys_if_engine pdftex:T</code>	112
<code>\property_ref:nn</code>	882		

T		talk mode to prefix (pgfkey) 4	
tag (opt.)	3	\tex_endlinechar:D	251
\tag_get:n	660	\tex_savepos:D	671, 675
\tag_if_active:T	736	\text_purify:n	564, 575, 856
\tag_if_active:TF	38	\the	329, 499,
\tag_if_active_p:	188		569, 580, 591, 645, 689, 829, 861, 957, 984
\tag_mc_begin:n	651	\tl_clear:N	380, 381
\tag_mc_begin_pop:n	683	\tl_gput_right:Nn	372, 374
\tag_mc_end:	681	\tl_gput_right:NV	371, 373
\tag_mc_end_push:	644	\tl_if_eq:NNTF	934
\tag_socket_use:n	641	\tl_if_head_eq_meaning:VNF	101
\tag_struct_begin:n	646	\tl_log:e	569, 580, 591
\tag_struct_end:	682	\tl_map_function:nN	382
\tag_struct_gput:ene	659	\tl_new:N	124, 125, 126, 368, 369, 370, 547, 922
tagsupport/memoize/include/extern/after		\tl_put_right:Ne	427, 434
(socket)	7, 549	\tl_put_right:Nn	425, 432
tagsupport/memoize/include/extern/after		\tl_put_right:NV	426, 428, 433
mmzx/actualtext (plug)	685	\tl_set:No	645, 861
tagsupport/memoize/include/extern/after		token registers:	
mmzx/alt (plug)	677	\mmzxtagtoks	544,
tagsupport/memoize/include/extern/after			564, 575, 586, 645, 689, 826, 829, 856, 861
mmzx/artifact (plug)	698	\toks	498, 499, 502
tagsupport/memoize/include/extern/before		\toksapp	99, 187
(socket)	7, 549		
tagsupport/memoize/include/extern/before			
mmzx/actualtext (plug)	685		
tagsupport/memoize/include/extern/before			
mmzx/alt (plug)	632		
tagsupport/memoize/include/extern/before			
mmzx/artifact (plug)	698		
tagsupport/tikz/picture/init mmzx (plug)	742		
talk (opt.)	3		

U

\use:c	109, 400, 665
\use_none:n	142
\UseName	510

X

\xdef	1007
\xtoksapp	99, 325, 335, 387,
	448, 453, 763, 825, 950, 957, 1010, 1024, 1032