mp

for version 0.2, January 3, 2022

C. Dardis <christopherdardis@gmail.com>

Table of Contents

1 Preamble

`lexical binding` is `t`.
Copyright: 2018-2022 - Chris Dardis
Author: C. Dardis <christopherdardis@gmail.com>
URL: `http://github.com/dardisco/mp`
Version: 0.2
Keywords: R, Sweave, knitr, latex, noweb, org
Package-Requires: ((ess) (org) (org-element) (tex))
Commentary: Makes a pdf from the materials in the 'default-directory' (or will search up the directory tree for the appropriate file type).

These may include .R, .tex, .Rnw, .org, .Rmd and .bib files. Support is provided for indexes, glossaries and table of contents.

Installation:

This package can be installed using

(package-install-from-file "/path/to/mp.el")

Or place the folliwng in your init.el file

(add-to-list 'load-path "~/path/to/directory") (require 'mp)

Usage:

To generate a .pdf file use the 'mp-mp' command. 'M-x mp-mp RET' This command can be called within the directory used to produce the .pdf. It can also be called from one of the files used to produce the .pdf. In the latter case, the file will first be saved if there are any changes.

There are a number of addition commands which you may wish to call independently: - 'mp-customize-common' - Set common customizable variables - 'mp-example' - Generate an example .pdf - 'mp-custom-example' - Call both commands above, in sequence - 'mp-customize-all' - Set all customizable variables - 'mp-customize-skeleton-string' - Set all customizable skeleton strings for the package (strings used by skeletons) - 'mp-clean' - Delete files in which have 'mp-clean-extensions' - 'mp-tex-new' - Make a new .tex file - 'mp-chunk-new' - Insert a new chunk

You may wish to define a shortcut for calling 'mp-chunk-new'. 'f8' is suggested as there are 8 characters in these prefixes. That is, "## —- " is 8 characters in length. The prefixes follow the conventions established by knitr. If so, also place something like this in your init.el file: "' (global-set-key (kbd "<f8>") (lambda () (interactive) (mp-chunk-new))) "'

or, for example, if using ESS and 'bind-key':

"' (bind-key (kbd "<f8>") (lambda () (interactive) (mp-chunk-new)) ess-mode-map) "'

Details:

The purpose of 'mp-mp' is to move from an R file or an intermediary file (.Rnw, .Rmd, .org, .bib, .tex) to a .pdf with a single command. This is done by selecting an entwiner (named after the conventions of 'weaving' with Sweave or 'knitting' with knitr; the default is knitr).

The simplest way to see this in action is to call 'mp-example'. This will generate a new directory in the current 'default-directory' for the files used to generate the final .pdf. This may also be a good way to start a new project.

Assuming we start with an .R file, ‘mp-mp’ generates an intermediary file, as appropriate, then uses the entwiner to generate a .tex file from the intermediary, then uses latexmk to generate a .pdf from the .tex file, which is then displayed.

Alternatively, we may start with an intermediary e.g. a .Rnw file. A .R file is generated from this before making the .tex file, as above. We may also start directly with a .tex file. In this case, no intermediary or .R file is created; we proceed directly to .pdf. Similarly in the case of a .bib file. Note that a .pdf will *not* be generated from a .bib file without a corresponding .tex file in the same directory.

Output from the various steps in compiling the .pdf appears in the ’*make-pdf*’ buffer, which (by default) will by used to display the .pdf once complete. Alternative .pdf viewers are supported. If another viewer is used, this output will remain visible in the ’*make-pdf*’ buffer.

Note that all code in the .R file is run via Rscript, to ensure there are no errors with this and in case this modifies another file in the workign directory e.g. a .bib file as is the case when running ‘mp-example’.

Updating R code chunks:

An advantage of ‘mp-mp’ is that code can be updated in either the .R file or the intermediary. Whichever has been more recently modified will be used as the source and the other as the destination. This allows for editing of code in either the .R file or the intermediary.

That is, when ‘mp-mp’ is called from an .R file or an intermediary, a corresponding .R or intermediary will be created or updated, as appropriate.

This starts by reading the chunks from the working file. This is the buffer currently open (if it has unsaved changes). This will be an .R file, or an entwiner file (.Rnw, .Rmd, .org).

For example, when working with an .R file, if there is *no* entwiner file (.Rnw, .org) with the same file name prefix in the same directory, this file will be *created* before generating a .tex file from this entwiner.

If an entwiner file *already exists*, this is *deleted*. A new entwiner file is created after reading the code chunks in the .R file.

As above, note that changes to a .tex file do *not* update the corresponding intermediary. Thus free text (outside .R code chunks) should generally be updated in the intermediary file, unless no further changes to the .R code is planned.

Customization:

To avoid errors, it is best to set the following customizable variables once at the start of a project: - ‘mp-entwiner’ - ‘mp-latex’ - ‘mp-bib’

If these are changed mid-way through, it may be necessary to first delete the intermediary, .tex and .pdf files before starting afresh from the .R file in the directory.

This package contains a variety of defaults which may be modified. For example, the preamble for a LaTeX document, which varies depending on whether LaTeX or XeTeX/LuaLaTeX is used to generate the .pdf.

Multiple plots per chunk:

All of the entwiners will save graphical output in one or more separate files, then include some these in the final .pdf.

Only knitr supports multiple plots per chunk by default.

By default, both Sweave and Org will include only the first plot/graph/figure produced. They both use the TeX command '\includegraphics' for this. The function 'mp-tex-to-pdf' will replace instances of this command with '\includepdf[pages=-, width=0.9\linewidth]', so that all plots/pages are included.

An alternative is to save each plot as a separate file and then output the TeX command to include all of them.

An example of a code block using this approach follows below. ‘‘‘ for(i in 1:4) file=paste("f", i, ".eps", sep="") postscript(file=file, paper="special", width=8, height=8) plot(rnorm(100)+i) dev.off() cat(" \\includegraphics", file, "\n", sep="") ‘‘‘ In Sweave, we would wrap the code above in: ‘‘‘ <<plots, results=tex, strip.white=false>>= **code** @ ‘‘‘ In Org, we would wrap the code above in: ‘‘‘ #+NAME: plots2 #+begin_src R :session *org-R-session* :exports both :results output :results verbatim :results latex **code** #+end_src ‘‘‘

History/goals:

This package was initially developed in 2012 to help to speed up the process of scientific writing. At that time, there was no easy was to tranistion quickly from a .R file to a .pdf and knitr was relatively new and was not being used routinely e.g. to generate package documentation.

The range of options has expanded dramatically since. In particulary, .Rmd appears to have become the most popular choice for performing this type of work.

Dependencies:

The following should be installed and on your path: - R, including the packages knitr and rmarkdown - LaTeX - pandoc - bash commands: locate, grep

The following emacs packages are also recommended: - org - ess - tex, bibtex - doc-view - markdown-mode - polymode, poly-R

This is designed for Emacs on debian linux with bash. It has not been evaluated on other systems.

For developers:

Function-local/temporary variables are named using as name1 e.g. 'v1', 'buffer1'.

2 defgroup

Documentation for `defgroup`:
Declare SYMBOL as a customization group containing MEMBERS. SYMBOL does not need to be quoted.

Third argument DOC is the group documentation. This should be a short description of the group, beginning with a capital and ending with a period. Words other than the first should not be capitalized, if they are not usually written so.

MEMBERS should be an alist of the form ((NAME WIDGET)...) where NAME is a symbol and WIDGET is a widget for editing that symbol. Useful widgets are 'custom-variable' for editing variables, 'custom-face' for editing faces, and 'custom-group' for editing groups.

The remaining arguments should have the form

[KEYWORD VALUE]...

For a list of valid keywords, see the common keywords listed in 'defcustom'.

See Info node '(elisp) Customization' in the Emacs Lisp manual for more information.

A table of all `defgroup`s follows:

`mp`

`mp-common`

Details of each `defgroup` follow below:

mp

`mp` [defgroup]

group-documentation
: Make Pdf. This is a series of variables and functions to simplity the process of pdf creation using R, LaTeX and the intermediaries (entwiners) knitr, Sweave and Org ('org-mode').

custom-prefix
: mp-

custom-version
: 0.2

custom-group
: ((mp custom-group) (mp-common custom-group) (mp-example-file-name custom-variable) (mp-timeout custom-variable) (mp-file-name-extensions custom-variable) (mp-function-sequence custom-variable) (mp-entwiner custom-variable) (mp-R-chunks-new-R custom-variable) (mp-chunk-name-prefix-R custom-variable) (mp-chunk-format-prefix-R custom-variable) (mp-chunk-content-suffix-R custom-variable) (mp-chunk-heading-level custom-variable) (mp-R-chunk-defaults-R-pkg

custom-variable) (mp-chunk-format-default-R-pkg custom-variable) (mp-chunk-format-default-sweave custom-variable) (mp-chunk-format-default-org custom-variable) (mp-bib custom-variable) (mp-preamble-bib custom-variable) (mp-preamble-biblatex custom-variable) (mp-doc-class-tex custom-variable) (mp-preamble-knitr custom-variable) (mp-preamble-sweave custom-variable) (mp-preamble-org custom-variable) (mp-latex custom-variable) (mp-latex-font custom-variable) (mp-xetex-font custom-variable) (mp-preamble-tex custom-variable) (mp-add-preamble-tex-extra custom-variable) (mp-preamble-tex-extra custom-variable) (mp-tex-beginning custom-variable) (mp-bibtex-style custom-variable) (mp-tex-end custom-variable) (mp-sentinel-sleep custom-variable) (mp-latex-options custom-variable) (mp-latexmkrc custom-variable) (mp-bib-default custom-variable) (mp-pdf-viewer custom-variable) (mp-clean custom-variable) (mp-clean-extensions custom-variable) (mp-clean-others custom-variable) (mp-skeleton-string custom-group))

mp-common

`mp-common` [defgroup]

group-documentation

Options that the user is most likely to wish to change. This group is part of ‘mp’.

custom-group

((mp-timeout custom-variable) (mp-entwiner custom-variable) (mp-chunk-heading-level custom-variable) (mp-bib custom-variable) (mp-clean custom-variable))

3 defun

Documentation for `defun`:
Define NAME as a function. The definition is (lambda ARGLIST [DOCSTRING] BODY...). See also the function 'interactive'. DECL is a declaration, optional, of the form (declare DECLS...) where DECLS is a list of elements of the form (PROP . VALUES). These are interpreted according to 'defun-declarations-alist'. The return value is undefined.

(fn NAME ARGLIST &optional DOCSTRING DECL &rest BODY)

A table of all `defun`s follows:

`mp-custom-example`	Run 'mp-customize-common' then 'mp-example'.
`mp-example`	Generate and run an example.
`mp-initialize`	Set the following 'defvar's to their staring value:
`mp-set-mp-dd`	Set the value of 'mp-dd'.
`mp-valid-file-name-p`	Return t is STRING is a valid filename.
`mp-run`	Run the appropriate list of functions from 'mp-function-sequence'.
`mp-R-entwiner`	Convert R to an entwiner or vice versa.
`mp-bib`	Check if there is a file named 'mp-fnb'.bib in 'mp-dd'.
`mp-R-new`	Make an .R file.
`mp-chunk-update-format`	Update the format of a CHUNK, per the value of 'mp-entwiner'.
`mp-R-chunk-R-citations`	Return a string which is R code that writes a bibliography file.
`mp-file-display`	Displays FILE-NAME.
`mp-file-open`	Open FILE-NAME in a buffer; set buffer to appropriate mode.
`mp-chunk-set-adfixes`	Set the prefixes and suffixes for chunks.
`mp-set-mode`	Set the appropriate mode for a file.

Details of each `defun` follow below:

mp-custom-example

`mp-custom-example` [Function]

Command ? *yes*
Arguments: *none*
Documentation: Run ‘mp-customize-common’ then ‘mp-example’.

```
((interactive)
 (mp-customize-common)
 (mp-example))
```

mp-example

`mp-example` [Function]

Command ? *yes*
Arguments: *none*
Documentation: Generate and run an example. Calls ‘mp-mp’, in the subdirectory ‘mp-example-file-name’.

```
((interactive)
 (mp-set-mp-dd)
 (let
     ((d1
       (concat mp-dd mp-example-file-name "/")))
   (delete-directory d1 t)
   (mkdir d1 t)
   (cd d1)
   (setq mp-dd d1 mp-fnb mp-example-file-name mp-fne "R")
   (mp-initialize)
   (message
    (concat "mp-example ... running in "
              (print mp-dd))))
 (mp-run mp-fne))
```

mp-initialize

`mp-initialize` [Function]

Command ? *no*
Arguments: *none*
Documentation: Set the following ‘defvar’s to their staring value: - ‘mp-df’ - ‘mp-fn’ - ‘mp-chunks’ Called by ‘mp-example’ and ‘mp-mp’.

```
((while
     (get-buffer-window "*make-pdf*")
   (delete-window
    (get-buffer-window "*make-pdf*")))
 (if
     (get-buffer "*make-pdf*")
     (progn
       (kill-buffer "*make-pdf*")))
 (balance-windows)
 (setq mp-df
       (sort
          (directory-files mp-dd nil
                           (mapconcat
                            #'(lambda
                                  (x)
                                  (concat "\\." x "$"))
                            mp-file-name-extensions "\\|")
                           t)
          #'file-newer-than-file-p)
       mp-fn
       (car mp-df)
       mp-chunks 'nil mp-exiting nil))
```

mp-set-mp-dd

`mp-set-mp-dd` [Function]

Command ? *no*

Arguments: *none*

Documentation: Set the value of 'mp-dd'. If 'mp-fnb' is an empty string, also set this to the same value as 'mp-dd'.

```
((setq mp-dd
       (cond
          ((buffer-file-name)
           (file-name-directory buffer-file-name))
          ((equal major-mode 'dired-mode)
           dired-directory)
          (t
           (shell-command-to-string "printf '%s' $HOME/")))
       default-directory mp-dd)
 (message
  (concat "mp-set-mp-dd ... mp-dd set to "
           (print mp-dd)))
 (if
     (equalp "" mp-fnb)
     (progn
       (setq mp-fnb
             (file-name-nondirectory
              (directory-file-name mp-dd)))
       (message
          (concat "mp-set-mp-dd ... mp-fnb set to "
                  (print mp-fnb))))))
```

mp-valid-file-name-p

`mp-valid-file-name-p` [Function]

Command ? *yes*

Arguments: *STRING*

Documentation: Return t is STRING is a valid filename. Uses 'file-name-invalid-regexp', which depends on the 'system-type'. An example of an invalid filename is "\0". See also the following for additional restrictions on Windows: URL 'https://docs.microsoft.com/en-us/windows/win32/fileio/naming-a-file'.

```
((interactive "FFilename: ")
       (not
        (string-match file-name-invalid-regexp STRING)))
```

mp-run

`mp-run` [Function]

Command ? *no*

Arguments: *none*

Documentation: Run the appropriate list of functions from ‘mp-function-sequence’. This is given by ‘mp-fne’.

```
((let
     ((seq1
       (catch '--cl-block-nil--
           (let
               ((--dolist-tail-- mp-function-sequence))
             (while --dolist-tail--
               (let
                     ((x
                       (car --dolist-tail--)))
                 (if
                       (string-match
                        (car x)
                        mp-fne)
                       (progn
                         (throw '--cl-block-nil--
                                  (eval
                                   (cdr x)))))
                 (setq --dolist-tail--
                         (cdr --dolist-tail--))))))))
   (message
    (concat "mp-run, with "
              (car mp-entwiner)
              " and "
              (cdr mp-entwiner)
              ", starting with ." mp-fne " file ..."))
   (let
       ((--dolist-tail-- seq1))
     (while --dolist-tail--
       (let
             ((x
               (car --dolist-tail--)))
           (funcall x)
           (setq --dolist-tail--
                 (cdr --dolist-tail--))))))
  (mp-exit "mp-run... done"))
```

mp-R-entwiner

`mp-R-entwiner` [Function]

Command ? *no*

Arguments: *none*

Documentation: Convert R to an entwiner or vice versa. Calls ‘mp-chunks-read’ to get code chunks from the current file. If this is an .R file and an 'entwiner' file already exists, this is deleted and replaced. update the *least recently used* file i.e. the .R or the entwiner, using, as appropriate 'mp-update-entwiner-from-R' or 'mp-update-R-from-entwiner'. If no such file exists, make one with 'mp-entwiner-new'.

```
((let
     ((rf1
       (car
          (cl-member
           (concat mp-fnb ".R")
           mp-df :test #'equalp)))
      (ef1
       (car
          (cl-member
           (concat mp-fnb "."
                     (car mp-entwiner))
           mp-df :test #'equalp))))
   (message "mp-R-entwiner ...")
   (if
       (not rf1)
       (progn
           (mp-R-new)
           (if
               (mp-empty-string-to-nil ef1)
               nil
             (mp-entwiner-new)))
     (if ef1
           (if
               (file-newer-than-file-p rf1 ef1)
               (mp-update-entwiner-from-R)
             (mp-update-R-from-entwiner))
       (progn
           (mp-file-display
            (concat mp-fnb ".R"))
           (mp-chunks-read)
           (mp-entwiner-new)))))
 (if mp-bib
     (progn
       (mp-bib)))
 (mp-shell-sentinel "mp-Rscript-run")
 (message "mp-R-entwiner ... done"))
```

mp-bib

mp-bib [Function]

Command ? *no*

Arguments: *none*

Documentation: Check if there is a file named 'mp-fnb'.bib in 'mp-dd'. If not, make one by calling 'mp-bib-skeleton'. See also the variable 'mp-bib'.

```
((if
     (directory-files mp-dd nil
                      (concat mp-fnb ".bib$"))
     nil
   (mp-file-display
    (concat mp-fnb ".bib"))
```

```
         (mp-bib-skeleton)
         (save-buffer)
         (kill-buffer)
         (delete-window)))
```

mp-R-new

mp-R-new [Function]

Command ? *no*

Arguments: *none*

Documentation: Make an .R file. If ‘mp-chunks’ is not an empty list, use these, otherwise use ‘mp-R-chunks-new-R’. Called by ‘mp-R-entwiner’ and ‘mp-update-R-from-entwiner’.

```
((if
     (equalp 'nil mp-chunks)
     (progn
       (if mp-bib
           (progn
             (setq mp-chunks
                     (cons
                      (list "R citations" ""
                              (mp-R-chunk-R-citations))
                      mp-chunks))))
       (let
           ((--dolist-tail--
             (reverse
              (copy-tree mp-R-chunks-new-R))))
         (while --dolist-tail--
           (let
               ((x
                   (car --dolist-tail--)))
             (setq mp-chunks
                     (cons x mp-chunks))
             (setq --dolist-tail--
                     (cdr --dolist-tail--)))))
       (if
           (string-match "^R-pkg-"
                           (cdr mp-entwiner))
           (progn
             (setq mp-chunks
                     (cons
                      (copy-tree mp-R-chunk-defaults-R-pkg)
                      mp-chunks))))))
 (let
     ((--dolist-tail-- mp-chunks))
   (while --dolist-tail--
     (let
           ((x
             (car --dolist-tail--)))
       (mp-chunk-update-format x)
       (setq --dolist-tail--
               (cdr --dolist-tail--)))))
 (setq mp-fne "R")
 (let
     ((fn1
       (concat mp-fnb ".R")))
   (if
       (get-buffer fn1)
       nil
     (mp-file-display fn1))
   (save-current-buffer
     (set-buffer fn1)
     (goto-char
      (point-min))
     (mapc #'mp-chunk-insert mp-chunks)
     (goto-char
      (point-max)))
   (save-buffer)
   (kill-buffer)
   (delete-window)))
```

mp-chunk-update-format

`mp-chunk-update-format` [Function]

Command ? *no*

Arguments: *CHUNK*

Documentation: Update the format of a CHUNK, per the value of 'mp-entwiner'. The format is the second element of CHUNK, a string.

```
((if
     (equalp ""
             (car
              (cdr CHUNK)))
     (progn
       (setcar
        (cdr CHUNK)
        (mp-chunk-format)))))
```

mp-R-chunk-R-citations

`mp-R-chunk-R-citations` [Function]

Command ? *no*

Arguments: *none*

Documentation: Return a string which is R code that writes a bibliography file. This file is named '‘mp-fnb’.bib.'. A citation appears therein for each loaded R package. Adds a handle to each citation, if necessary. This is added to 'mp-chunks' when 'mp-bib' is non-nil. Called by 'mp-R-new'.

```
((concat "print(.packages())\n## vector for citations\nvc1 <- unlist(sapply(X=rev(.packages()),\n
```

mp-file-display

`mp-file-display` [Function]

Command ? *no*

Arguments: *FILE-NAME*

Documentation: Displays FILE-NAME. If the file is already open if a buffer, calls 'pop-to-buffer'. If not, calls 'mp-file-open'. Finally, calls 'mp-set-mode' and 'mp-chunk-set-adfixes'.

```
((setq mp-fne
       (file-name-extension FILE-NAME))
 (let
     ((default-directory mp-dd)
      (bn1
       (car
          (member FILE-NAME
                   (mapcar 'buffer-name
                            (buffer-list))))))
   (if bn1
       (pop-to-buffer bn1)
     (mp-file-open FILE-NAME)))
 (mp-set-mode)
 (mp-chunk-set-adfixes))
```

mp-file-open

`mp-file-open` [Function]

Command ? *no*

Arguments: *FILE-NAME*

Documentation: Open FILE-NAME in a buffer; set buffer to appropriate mode. Return buffer. This function is derived from a combination of the functions 'find-file-noselect' and 'find-file-noselect-1'. Called by 'mp-file-display'.

```
((let*
     ((default-directory mp-dd)
      (truename1
       (abbreviate-file-name
          (file-truename FILE-NAME)))
      (buf1 nil))
   (if
       (featurep 'uniquify)
       (let
             ((uniquify-managed
               (progn
                 (make-local-variable 'uniquify-managed)
                 nil)))
           (setq buf1
                 (create-file-buffer FILE-NAME)))
     (setq buf1
           (create-file-buffer FILE-NAME)))
   (pop-to-buffer buf1 nil t)
   (set-buffer buf1)
   (set-buffer-multibyte t)
   (if
       (and
          (=
           (buffer-size buf1)
           0)
          (directory-files mp-dd nil
                           (concat "^" FILE-NAME "$")))
       (progn
           (insert-file-contents-literally truename1 t)))
   (setq buffer-file-truename truename1 buffer-file-name
           (expand-file-name buffer-file-truename)
           buffer-file-number
           (nthcdr 10
                   (file-attributes truename1)))))
```

mp-chunk-set-adfixes

`mp-chunk-set-adfixes` [Function]

Command ? *no*

Arguments: *none*

Documentation: Set the prefixes and suffixes for chunks. These depend on the 'file-name-extension' of the curent buffer, if applicable, otherwise on 'mp-fne'. The following file-local variables are set: - 'mp-chunk-heading-level-prefix' - 'mp-chunk-heading-level-suffix' - 'mp-chunk-name-prefix' - 'mp-chunk-name-suffix' - 'mp-chunk-format-prefix' - 'mp-chunk-content-suffix' Called by 'mp-file-display'.

```
((let
     ((fe1
       (car
        (cl-member
         (file-name-extension
          (buffer-file-name))
         '("R" "Rnw" "org" "Rmd")
         :test #'equalp))))
   (if fe1 nil
     (setq fe1 mp-fne))
   (cond
    ((equalp "R" fe1)
     (setq mp-chunk-name-prefix mp-chunk-name-prefix-R mp-chunk-format-prefix mp-chunk-format-prefix-R mp
    ((equalp "Rnw" fe1)
     (setq mp-chunk-heading-level-prefix
           (elt mp-chunk-heading-level 0)
           mp-chunk-heading-level-suffix "}" mp-chunk-name-prefix mp-chunk-name-prefix-Rnw mp-chunk-nam
    ((equalp "org" fe1)
     (setq mp-chunk-heading-level-prefix
           (elt mp-chunk-heading-level 1)
           mp-chunk-heading-level-suffix "" mp-chunk-name-prefix mp-chunk-name-prefix-org mp-chunk-form
    ((equalp "Rmd" fe1)
     (setq mp-chunk-heading-level-prefix
           (elt mp-chunk-heading-level 2)
           mp-chunk-heading-level-suffix "" mp-chunk-name-prefix mp-chunk-name-prefix-Rmd mp-chunk-name
   (message
    (concat "mp-chunk-set-adfixes... set per for ." fe1 " file extension"))))
```

mp-set-mode

`mp-set-mode` [Function]

Command ? *no*

Arguments: *none*

Documentation: Set the appropriate mode for a file.

```
((cond
((equalp "R" mp-fne)
 (R-mode))
((equalp "Rnw" mp-fne)
 (Rnw-mode))
((equalp "org" mp-fne)
 (org-mode))
((equalp "Rmd" mp-fne)
 (markdown-mode)
 (if poly-markdown+r-mode nil
   (poly-markdown+r-mode)))
((equalp "tex" mp-fne)
 (tex-mode))
((equalp "bib" mp-fne)
 (bibtex-mode))
(t
 (fundamental-mode))))
```

4 defcustom

Documentation for `defcustom`:
Declare SYMBOL as a customizable variable. SYMBOL is the variable name; it should not be quoted. STANDARD is an expression specifying the variable's standard value. It should not be quoted. It is evaluated once by 'defcustom', and the value is assigned to SYMBOL if the variable is unbound. The expression itself is also stored, so that Customize can re-evaluate it later to get the standard value. DOC is the variable documentation.

This macro uses 'defvar' as a subroutine, which also marks the variable as "special", so that it is always dynamically bound even when 'lexical-binding' is t.

The remaining arguments to 'defcustom' should have the form

[KEYWORD VALUE]...

The following keywords are meaningful:

:type VALUE should be a widget type for editing the symbol's value. Every 'defcustom' should specify a value for this keyword. :options VALUE should be a list of valid members of the widget type. :initialize VALUE should be a function used to initialize the variable. It takes two arguments, the symbol and value given in the 'defcustom' call. The default is 'custom-initialize-reset'. :set VALUE should be a function to set the value of the symbol when using the Customize user interface. It takes two arguments, the symbol to set and the value to give it. The function should not modify its value argument destructively. The default choice of function is 'set-default'. :get VALUE should be a function to extract the value of symbol. The function takes one argument, a symbol, and should return the current value for that symbol. The default choice of function is 'default-value'. :require VALUE should be a feature symbol. If you save a value for this option, then when your init file loads the value, it does (require VALUE) first. :set-after VARIABLES Specifies that SYMBOL should be set after the list of variables VARIABLES when both have been customized. :risky Set SYMBOL's 'risky-local-variable' property to VALUE. :safe Set SYMBOL's 'safe-local-variable' property to VALUE. See Info node '(elisp) File Local Variables'.

The following common keywords are also meaningful.

:group VALUE should be a customization group. Add SYMBOL (or FACE with 'defface') to that group. :link LINK-DATA Include an external link after the documentation string for this item. This is a sentence containing an active field which references some other documentation.

There are several alternatives you can use for LINK-DATA:

(custom-manual INFO-NODE) Link to an Info node; INFO-NODE is a string which specifies the node name, as in "(emacs)Top".

(info-link INFO-NODE) Like 'custom-manual' except that the link appears in the customization buffer with the Info node name.

(url-link URL) Link to a web page; URL is a string which specifies the URL.

(emacs-commentary-link LIBRARY) Link to the commentary section of LIBRARY.

(emacs-library-link LIBRARY) Link to an Emacs Lisp LIBRARY file.

(file-link FILE) Link to FILE.

(function-link FUNCTION) Link to the documentation of FUNCTION.

(variable-link VARIABLE) Link to the documentation of VARIABLE.

(custom-group-link GROUP) Link to another customization GROUP.

You can specify the text to use in the customization buffer by adding ':tag NAME' after the first element of the LINK-DATA; for example, (info-link :tag "foo" "(emacs)Top") makes a link to the Emacs manual which appears in the buffer as 'foo'.

An item can have more than one external link; however, most items have none at all. :version VALUE should be a string specifying that the variable was first introduced, or its default value was changed, in Emacs version VERSION. :package-version VALUE should be a list with the form (PACKAGE . VERSION) specifying that the variable was first introduced, or its default value was changed, in PACKAGE version VERSION. This keyword takes priority over :version. For packages which are bundled with Emacs releases, the PACKAGE and VERSION must appear in the alist 'customize-package-emacs-version-alist'. Since PACKAGE must be unique and the user might see it in an error message, a good choice is the official name of the package, such as MH-E or Gnus. :tag LABEL Use LABEL, a string, instead of the item's name, to label the item in customization menus and buffers. :load FILE Load file FILE (a string) before displaying this customization item. Loading is done with 'load', and only if the file is not already loaded.

If SYMBOL has a local binding, then this form affects the local binding. This is normally not what you want. Thus, if you need to load a file defining variables with this form, or with 'defvar' or 'defconst', you should always load that file _outside_ any bindings for these variables. ('defvar' and 'defconst' behave similarly in this respect.)

See Info node '(elisp) Customization' in the Emacs Lisp manual for more information.

A table of all `defcustom`s follows:

`mp-example-file-name`
: Name of directory and of 'file-name-base' to use for an example.

`mp-timeout`
: Seconds to wait for user input when calling interactive functions.

`mp-file-name-extensions`
: The 'file-name-extension's (as strings) with which 'mp' works.

`mp-function-sequence`
: An alist where:

`mp-entwiner`
: How to 'entwine' the files in 'mp-dd'.

`mp-R-chunks-new-R`
: A 'list' of chunks for a new .R file.

`mp-chunk-name-prefix-R`
: Prefix for the chunk name in an .R file.

`mp-chunk-format-prefix-R`
: Prefix for the chunk format in an .R file.

`mp-chunk-content-suffix-R`
: Suffix for the chunk content in an .R file.

`mp-chunk-heading-level`
: Prefixes indicating the heading level for a chunk.

`mp-R-chunk-defaults-R-pkg`
: Default setup options for 'knitr'.

Details of each `defcustom` follow below:

mp-example-file-name

`mp-example-file-name` [User Option]

standard-value
: ((funcall (function (closure (t) nil mp-example))))

custom-type
: (string)

safe-local-variable
: mp-valid-file-name-p

custom-requests
: nil

variable-documentation
: Name of directory and of ‘file-name-base’ to use for an example. Used by ‘mp-example’. Avoid using special characters in this string. See also ‘mp-valid-file-name-p’.

mp-timeout

`mp-timeout` [User Option]

standard-value
: ((funcall (function (closure (t) nil 0.2))))

custom-type
: (number)

custom-requests
: nil

variable-documentation
: Seconds to wait for user input when calling interactive functions. A low value will not give the user time to respond to prompts. In this case, the default values are used. See the functions ‘with-timeout’ and ‘y-or-n-p-with-timeout’. Used by: - ‘mp-find-file-name-interactive’

mp-file-name-extensions

`mp-file-name-extensions` [User Option]

standard-value
: ((funcall (function (closure (t) nil (quote (R Rnw org Rmd md tex bib pdf))))))

custom-type
: (repeat (string `:tag` extension))

custom-requests
: nil

variable-documentation
: The ‘file-name-extension’s (as strings) with which ‘mp’ works. These age given as a list of strings. They are not case sensitive (see ‘case-fold-search’).

mp-function-sequence

`mp-function-sequence` [User Option]

standard-value
: ((funcall (function (closure (t) nil (quote ((R cond ((equalp Rnw (car mp-entwiner)) (quote (mp-R-entwiner mp-Rnw-to-tex mp-tex-to-pdf mp-view-pdf))) ((equalp org (car mp-entwiner)) (quote (mp-R-entwiner mp-org-to-tex mp-tex-to-pdf mp-view-pdf))) ((equalp Rmd (car mp-entwiner)) (quote (mp-R-entwiner mp-Rmd-to-pdf mp-view-pdf)))) (Rnw quote (mp-R-entwiner mp-Rnw-to-tex mp-tex-to-pdf mp-view-pdf)) (org quote (mp-R-entwiner mp-org-to-tex mp-tex-to-pdf mp-view-pdf)) (Rmd quote (mp-R-entwiner mp-Rmd-to-pdf mp-view-pdf)) (tex\|bib quote (mp-tex-to-pdf mp-view-pdf)) (pdf quote (mp-view-pdf))))))))

custom-type
: (alist `:key-type` (string `:format` %t %v `:tag` file extension, a string) `:value-type` (sexp `:format` %t %v `:tag` sexp: a list, or function returning a list))

custom-requests
: nil

variable-documentation
: An alist where: The key is the file name extension, a regular expression. The value is list of functions to run. Used by ‘mp-run’ as the overall control flow.

mp-entwiner

`mp-entwiner` [User Option]

standard-value
: ((funcall (function (closure (t) nil (quote (Rnw . R-pkg-knitr))))))

custom-type
: (radio (const `:doc` Rnw with knitr (the default) `:value` (Rnw . R-pkg-knitr)) (const `:doc` Rnw with Sweave `:value` (Rnw . Sweave)) (const `:doc` Org (uses .org file) `:value` (org . Org)) (const `:doc` md with knitr `:value` (Rmd . R-pkg-rmarkdown)))

custom-requests

nil

variable-documentation

How to 'entwine' the files in 'mp-dd'. Given in the form of a cons, where: - key-type = 'file-name-extension' (a string) - value-type = method (a string)

The 'entwiner' is an intermediary/ bridging file, which acts as a 'go-between' for .R and .tex files.

"R-pkg-knitr" and "Sweave" use an .Rnw file (see 'Rnw-mode'). "Org" uses an .org file (see 'org-mode') See also info node '(org)LaTeX export'. "R-pkg-rmarkdown" uses an .Rmd file (see 'poly-markdown+r-mode') and an .md file (see 'markdown-mode').

mp-R-chunks-new-R

`mp-R-chunks-new-R` [User Option]

standard-value

((funcall (function (closure (t) nil (quote ((Hello world print('Hello, World!')) (Plots (cond ((string-match R-pkg- (cdr mp-entwiner)) results='asis') ((equalp Sweave (cdr mp-entwiner)) results=verbatim, fig=TRUE, pdf=TRUE) ((equalp Org (cdr mp-entwiner)) `:session` *org-R-session* `:exports` both `:results` output `:results` graphics `:file` "p1.pdf")) ## The examples below are taken from ?graphics::`plot` require('stats') # for lowess, rpois, rnorm plot(cars) lines(lowess(cars)) plot(sin, -pi, 2*pi) # see ?plot.function ## Discrete Distribution Plot: plot(table(rpois(100, 5)), type='h', col='red', lwd=10, main='rpois(100, lambda=5)') ## Simple quantiles/ECDF, see ecdf() library(stats) for a better one: plot(x <- sort(rnorm(47)), type='s', main='plot(x, type=\'s\')') points(x, cex=0.5, col='dark red')) (R session information (cond ((string-match R-pkg- (cdr mp-entwiner)) results='asis') ((equalp Sweave (cdr mp-entwiner)) results=tex) ((equalp Org (cdr mp-entwiner)) `:session` *org-R-session* `:exports` both `:results` output `:results` verbatim `:results` latex)) (if (equalp Rmd (car mp-entwiner)) utils::`sessionInfo`() utils::`toLatex`(utils::`sessionInfo`())))))))))

custom-type

(alist `:key-type` (string `:format` %t %v `:tag` Chunk name) `:value-type` (group (sexp `:format` %t %v `:tag` Chunk format) (sexp `:format` %t %v `:tag` Chunk content)))

custom-requests

nil

variable-documentation

A 'list' of chunks for a new .R file. Each element is a list with 3 elements. - first element = chunk name, a string - second element = chunk format, a string or a symbolic expression which evaluates to a string - third element = chunk content, a string or a symbolic expression which evaluates to a string The second and third elements may e.g. be conditional, e.g. depending on the value of ‘mp-entwiner’. Be sure to escape: - double quotes i.e. `""` - single quotes within single quotes e.g. ’abc ’def’ ghi’ When editing this with ‘customize’, use ‘electric-newline-and-maybe-indent’ for carraige return.

mp-chunk-name-prefix-R

`mp-chunk-name-prefix-R` [User Option]

standard-value
: ((funcall (function (closure (t) nil ## —-))))

custom-type
: (string)

custom-requests
: nil

variable-documentation
: Prefix for the chunk name in an .R file. The default value follows the convention used by ’knitr’.

mp-chunk-format-prefix-R

`mp-chunk-format-prefix-R` [User Option]

standard-value
: ((funcall (function (closure (t) nil ,))))

custom-type
: (string)

custom-requests
: nil

variable-documentation
: Prefix for the chunk format in an .R file. The default value follows the convention used by ’knitr’.

mp-chunk-content-suffix-R

`mp-chunk-content-suffix-R` [User Option]

standard-value
: ((funcall (function (closure (t) nil))))

custom-type
: (sexp)

custom-requests
: nil

variable-documentation

Suffix for the chunk content in an .R file.

mp-chunk-heading-level

`mp-chunk-heading-level` [User Option]

standard-value

((funcall (function (closure (t) nil (quote (\section *** ###))))))

custom-type

(choice `:tag` (list `:tag` level 1 = part (string `:tag` LaTeX `:value` \part) (string `:tag` org `:value` *) (string `:tag` Rnw `:value` #)) (list `:tag` level 2 = chapter [LaTeX - only for \documentclassbook or report] (string `:tag` LaTeX `:value` \chapter) (string `:tag` org `:value` **) (string `:tag` Rnw `:value` ##)) (list `:tag` level 3 = section (string `:tag` LaTeX `:value` \section) (string `:tag` org `:value` ***) (string `:tag` Rnw `:value` #)) (list `:tag` level 4 = subsection (string `:tag` LaTeX \subsection) (string `:tag` org `:value` ****) (string `:tag` Rnw `:value` ##)) (list `:tag` level 5 = subsubsection (string `:tag` LaTeX `:value` \subsubsection) (string `:tag` org `:value` *****) (string `:tag` Rnw `:value` ###)) (list `:tag` level 6 = paragraph (string `:tag` LaTeX `:value` \paragraph) (string `:tag` org `:value` ******) (string `:tag` Rnw `:value` ######)) (list `:tag` level 7 = sub-paragraph [markdown - not applicable] (string `:tag` LaTeX `:value` \subparagraph) (string `:tag` org `:value` *******) (string `:tag` Rnw `:value`)) (list `:tag` none (const `:tag` LaTeX `:value`) (const `:tag` org `:value`) (const `:tag` Rnw `:value`)) (list `:tag` custom (string `:tag` LaTeX `:value` prefix-for-LaTeX) (string `:tag` org `:value` prefix-for-org) (string `:tag` Rnw `:value` prefix-for-Rnw)))

custom-requests

nil

variable-documentation

Prefixes indicating the heading level for a chunk. Specify these in the form of a list of two in the form: '(heading-level-for-LaTeX heading-for-Org)

The value is used to prior to a chunk-name. As a TeX command, a closing `""` is added e.g. '\subsection'chunk-name''.

Regarding levels of depth: - LaTeX uses 7 heading levels by default - 'org-mode' supports up to 8 heading levels (only 7, * to *******, are provided as options here) - markdown uses 6 heading levels, # to ###### Thus, we here restrict outselves to 7, bearing in mind that other formats can be more restrictive e.g. markdown supports 6. This is not meant to indicate a direct correspondance between heading levels in org-mode and LaTeX.

These values are used by 'mp-chunk-insert'.

For org-mode, see also 'org-level-faces', 'org-heading-components' and 'org-element-headline-parser'. For markdown-mode, see also 'markdown-regex-header'.

mp-R-chunk-defaults-R-pkg

`mp-R-chunk-defaults-R-pkg` [User Option]

standard-value

((funcall (function (closure (t) nil (quote (Default options for knitr and rmarkdown ### Defaults for chunks typeset with knitr library('knitr') ### defaults for all chunks opts_chunk$set(eval=TRUE, ## text results echo=TRUE, results=c('markup', 'asis', 'hold', 'hide')[1], collapse=FALSE, warning=TRUE, message=TRUE, error=TRUE, split=FALSE, include=TRUE, strip.white=TRUE, ## code decoration tidy=FALSE, prompt=FALSE, comment='##', highlight=TRUE, size='normalsize', background=c('#F7F7F7', colors()[479], c(0.1, 0.2, 0.3))[1], ## cache cache=FALSE, ## plots fig.path=c('figure', 'figure/minimal-')[1], fig.keep=c('high', 'none', 'all', 'first', 'last')[1], fig.align=c('center', 'left', 'right', 'default')[1], fig.show=c('hold', 'asis', 'animate', 'hide')[1], dev=c('pdf', 'png', 'tikz')[1], fig.width=7, fig.height=7, #inches fig.env=c('figure', 'marginfigure')[1], fig.pos=c('', 'h', 't', 'b', 'p', 'H')[1]) opts_knit$set(out.format='latex') knit_theme$set('biogoo') ### Set R options options(formatR.arrow=TRUE, width=60) knit_hooks$set(inline = function(x) ## if (is.numeric(x)) return(knitr::`:format`_sci(x, 'latex')) highr::`:hi`_latex(x)) ### uncomment below to change theme ## knit_theme$get() ## opts_knit$set(out.format='latex') ## thm1 <- knit_theme$get('acid') ## knit_theme$set(thm1)))))))

custom-links

((url-link `:tag` Code chunks and package options http://yihui.name/knitr/options) (url-link `:tag` Hooks - knitr documentation http://yihui.name/knitr/hooks))

custom-type

(string)

custom-requests

nil

variable-documentation

Default setup options for 'knitr'. Used by 'mp-R-new'. This list is not exhaustive. Common options are given as vectors, with a choice indicated by the index, in square brackets.

5 defvar

Documentation for `defvar`:
Define SYMBOL as a variable, and return SYMBOL. You are not required to define a variable in order to use it, but defining it lets you supply an initial value and documentation, which can be referred to by the Emacs help facilities and other programming tools. The ‘defvar’ form also declares the variable as "special", so that it is always dynamically bound even if ‘lexical-binding’ is t.

If SYMBOL’s value is void and the optional argument INITVALUE is provided, INITVALUE is evaluated and the result used to set SYMBOL’s value. If SYMBOL is buffer-local, its default value is what is set; buffer-local values are not affected. If INITVALUE is missing, SYMBOL’s value is not set.

If SYMBOL has a local binding, then this form affects the local binding. This is usually not what you want. Thus, if you need to load a file defining variables, with this form or with ‘defconst’ or ‘defcustom’, you should always load that file _outside_ any bindings for these variables. (‘defconst’ and ‘defcustom’ behave similarly in this respect.)

The optional argument DOCSTRING is a documentation string for the variable.

To define a user option, use ‘defcustom’ instead of ‘defvar’.

(fn SYMBOL &optional INITVALUE DOCSTRING)

A table of all `defvar`s follows:

`mp-dd`	The ‘default-directory’; set when ‘mp-set-mp-dd’ is run.
`mp-fnb`	The ‘file-name-base’ (a string) for the working file in ‘mp-dd’.
`mp-fne`	The ‘file-name-extension’ (a string) for the working file in ‘mp-dd’.
`mp-chunks`	Stores value for code chunks as a list of strings.
`mp-df`	A list of the ‘directory-files’ in ‘mp-dd’.
`mp-fn`	The file-name i.e. ‘mp-fnb’.‘mp-fne’.

Details of each `defvar` follow below:

mp-dd

`mp-dd` [defvar]

variable-documentation

The ‘default-directory’; set when ‘mp-set-mp-dd’ is run. Defined here as a dynamicaly-bound variable so that various functions can refer to this value.

mp-fnb

`mp-fnb` [defvar]

variable-documentation

The ‘file-name-base’ (a string) for the working file in ‘mp-dd’. Set when ‘mp-mp’ is run. Defined here as a dynamic variable so that various functions can use this value.

mp-fne

`mp-fne` [defvar]

variable-documentation

The ‘file-name-extension’ (a string) for the working file in ‘mp-dd’. This is set when ‘mp-mp’ is run. Defined here as a dynamic variable so that various functions can use this value.

mp-chunks

`mp-chunks` [defvar]

variable-documentation

Stores value for code chunks as a list of strings. The value should be a ‘list’ of 3 elements where the first is the name of the chunk, the second provides formattinhg information the third is the content of the chunk.

The second element, chunk formatting should reflect the entwiner in use; see also ‘mp-sweave-opts’ and ‘mp-org-opts’.

mp-df

`mp-df` [defvar]

variable-documentation

A list of the ‘directory-files’ in ‘mp-dd’. Defined here as a dynamicaly-bound variable so that various functions can refer to this value.

mp-fn

`mp-fn` [defvar]

variable-documentation

The file-name i.e. ‘mp-fnb’.‘mp-fne’. Defined here as a dynamicaly-bound variable so that various functions can refer to this value.

6 defconst

Documentation for `defconst`:
Define SYMBOL as a constant variable. This declares that neither programs nor users should ever change the value. This constancy is not actually enforced by Emacs Lisp, but SYMBOL is marked as a special variable so that it is never lexically bound.

The 'defconst' form always sets the value of SYMBOL to the result of evalling INIT-VALUE. If SYMBOL is buffer-local, its default value is what is set; buffer-local values are not affected. If SYMBOL has a local binding, then this form sets the local binding's value. However, you should normally not make local bindings for variables defined with this form.

The optional DOCSTRING specifies the variable's documentation string.

(fn SYMBOL INITVALUE [DOCSTRING])

A table of all `defconst`s follows:

`mp-chunk-name-prefix-Rnw`	Prefix for the chunk name in an .Rnw file.
`mp-chunk-name-prefix-org`	Prefix for the chunk name in an .org file.
`mp-chunk-name-prefix-Rmd`	Prefix for the chunk name in an .Rmd file.
`mp-chunk-format-prefix-Rnw`	Prefix for the chunk format in an .Rnw file.
`mp-chunk-format-prefix-org`	Prefix for the chunk format in an .org file.
`mp-chunk-format-prefix-Rmd`	Prefix for the chunk format in an .Rmd file.
`mp-chunk-name-suffix-Rnw`	Suffix for the chunk name in an .Rnw file.
`mp-chunk-name-suffix-Rmd`	Suffix for the chunk name in an .Rmd file.
`mp-chunk-content-suffix-Rnw`	Suffix for the chunk content in an .Rnw file.
`mp-chunk-content-suffix-Rmd`	Suffix for the chunk content in an .Rmd file.
`mp-chunk-content-suffix-org`	Suffix for the chunk content in an .org file.

Details of each `defconst` follow below:

mp-chunk-name-prefix-Rnw

`mp-chunk-name-prefix-Rnw` [defconst]

variable-documentation
: Prefix for the chunk name in an .Rnw file.

risky-local-variable
: t

mp-chunk-name-prefix-org

`mp-chunk-name-prefix-org` [defconst]

variable-documentation
: Prefix for the chunk name in an .org file. See also 'mp-org-opts'.

risky-local-variable
: t

mp-chunk-name-prefix-Rmd

`mp-chunk-name-prefix-Rmd` [defconst]

variable-documentation
: Prefix for the chunk name in an .Rmd file.

risky-local-variable
: t

mp-chunk-format-prefix-Rnw

`mp-chunk-format-prefix-Rnw` [defconst]

variable-documentation
: Prefix for the chunk format in an .Rnw file.

risky-local-variable
: t

mp-chunk-format-prefix-org

`mp-chunk-format-prefix-org` [defconst]

variable-documentation
: Prefix for the chunk format in an .org file.

risky-local-variable
: t

mp-chunk-format-prefix-Rmd

`mp-chunk-format-prefix-Rmd` [defconst]

variable-documentation

Prefix for the chunk format in an .Rmd file. Note that this is not essential i.e. a code chunk could begin "'r chunk-name, echo=TRUE rather than "'r, chunk-name, echo=TRUE The convention of an extra comma follows that used by knitr.

risky-local-variable

t

mp-chunk-name-suffix-Rnw

`mp-chunk-name-suffix-Rnw` [defconst]

variable-documentation

Suffix for the chunk name in an .Rnw file.

risky-local-variable

t

mp-chunk-name-suffix-Rmd

`mp-chunk-name-suffix-Rmd` [defconst]

variable-documentation

Suffix for the chunk name in an .Rmd file.

risky-local-variable

t

mp-chunk-content-suffix-Rnw

`mp-chunk-content-suffix-Rnw` [defconst]

variable-documentation

Suffix for the chunk content in an .Rnw file.

risky-local-variable

t

mp-chunk-content-suffix-Rmd

`mp-chunk-content-suffix-Rmd` [defconst]

variable-documentation

Suffix for the chunk content in an .Rmd file.

risky-local-variable

t

mp-chunk-content-suffix-org

`mp-chunk-content-suffix-org` [defconst]

variable-documentation

Suffix for the chunk content in an .org file.

risky-local-variable

t

7 defvar-local

Documentation for `defvar-local`:
Define VAR as a buffer-local variable with default value VAL. Like ‘defvar’ but additionally marks the variable as being automatically buffer-local wherever it is set.

(fn VAR VAL &optional DOCSTRING)

A table of all `defvar-local`s follows:

`mp-chunk-heading-level-prefix`	Value of chunk heading level prefix.
`mp-chunk-heading-level-suffix`	Value of chunk heading level suffix.
`mp-chunk-name-prefix`	Value of chunk name prefix.
`mp-chunk-name-suffix`	Value of chunk name suffix.
`mp-chunk-format-prefix`	Value of chunk format prefix.
`mp-chunk-content-suffix`	Value of chunk content suffix.

Details of each `defvar-local` follow below:

mp-chunk-heading-level-prefix

`mp-chunk-heading-level-prefix` [defvar-local]

variable-documentation

Value of chunk heading level prefix. Set with ‘mp-chunk-set-adfixes’.

mp-chunk-heading-level-suffix

`mp-chunk-heading-level-suffix` [defvar-local]

variable-documentation

Value of chunk heading level suffix. Set with ‘mp-chunk-set-adfixes’.

mp-chunk-name-prefix

`mp-chunk-name-prefix` [defvar-local]

variable-documentation

Value of chunk name prefix. Set with ‘mp-chunk-set-adfixes’.

mp-chunk-name-suffix

`mp-chunk-name-suffix` [defvar-local]

variable-documentation

Value of chunk name suffix. Set with ‘mp-chunk-set-adfixes’.

mp-chunk-format-prefix

`mp-chunk-format-prefix` [defvar-local]

variable-documentation

Value of chunk format prefix. Set with ‘mp-chunk-set-adfixes’.

mp-chunk-content-suffix

`mp-chunk-content-suffix` [defvar-local]

variable-documentation

Value of chunk content suffix. Set with ‘mp-chunk-set-adfixes’.

8 Other symbolic expressions

```
(mapc #'require
      '(org ess tex bibtex doc-view ibuffer autorevert cl calendar timer))

(setq org-src-fontify-natively nil)
```

Index