

Config

Leo Aparisi de Lannoy

February 18, 2023

Contents

1	Compilation	3
2	Basic	3
2.1	ID	3
2.2	Good defaults	3
2.2.1	Browser	4
2.2.2	Which-key	4
2.3	Visual	4
2.3.1	Font	4
2.3.2	Theme	5
2.3.3	Default visual	5
2.3.4	Starting image	5
2.3.5	Theme Magic	5
2.4	Marginalia	6
2.5	File Templates	7
2.6	Editor config	7
3	Org-Mode	7
3.1	Defaults	7
3.2	Babel	8
3.3	Visuals	8
3.3.1	Org-modern	8
3.3.2	General	10
3.3.3	Org-appear	11
3.3.4	Ligatures	12
3.3.5	Latex improvement	13
3.4	Bullets	13
3.5	Agenda	13
3.5.1	Visual	13

3.6	Super-Agenda	14
3.6.1	Config	14
3.6.2	Customize	14
3.7	Roam	14
3.7.1	Defaults	14
3.7.2	Daily	14
3.7.3	Visuals	15
3.8	Ob-async	16
3.8.1	Julia support	16
3.8.2	Jupyter Integration	16
3.9	Org-Diff	16
3.10	Pandoc import	17
3.11	Export	17
3.11.1	Preview	17
3.12	Zotero Integration	18
3.13	Org-Chef	18
3.14	Bibtex-Integration	19
3.14.1	Citar	19
3.14.2	Org-Roam integration	20
3.15	Latex templates	21
3.15.1	Preview	21
3.15.2	Article	22
3.15.3	Beamer	24
3.15.4	Export	27
3.16	Capture	28
3.16.1	Doct	28
4	Company	35
5	LSP	36
5.1	Digestif	36
5.2	LTeX	36
6	VLFI	37
7	PDF-Tools	37
7.1	Fix annotation bug	37
7.2	Dark mode	37
8	Option key Fix	37
9	Centaur	38

10 Email	38
10.1 mu4e	38
10.2 Notification	40
11 RSS	40
11.1 Visual	40

1 Compilation

```

1 brew tap railwaycat/emacs-macport
2 brew install emacs-mac --with-mac-metal --with-natural-title-bar
  ↳ --with-native-compilation --with-xwidget

```

2 Basic

2.1 ID

```

1 (setq user-full-name "Leo Aparisi de Lannoy"
2   user-mail-address "leoaparisi@gmail.com")

```

2.2 Good defaults

```

1 (setq-default
2   delete-by-moving-to-trash t           ; Delete files to trash
3   window-combination-resize t          ; take new window space from
  ↳ all other windows (not just current)
4   x-stretch-cursor t)                  ; Stretch cursor to the glyph
  ↳ width
5
6 (setq undo-limit 80000000               ; Raise undo-limit to 80Mb
7   evil-want-fine-undo t                 ; By default while in insert
  ↳ all changes are one big blob. Be more granular
8   auto-save-default t                   ; Nobody likes to loose work,
  ↳ I certainly don't
9   truncate-string-ellipsis "...")
10  scroll-margin 2)                       ; It's nice to maintain a
  ↳ little margin

```

```

11
12 (display-time-mode 1) ; Enable time in the mode-line
13
14 (unless (string-match-p "^Power N/A" (battery)) ; On laptops...
15   (display-battery-mode 1)) ; it's nice to know how much
    ↪ power you have
16
17 (global-subword-mode 1) ; Iterate through CamelCase
    ↪ words

```

2.2.1 Browser

```

1 (setq browse-url-chrome-program "brave")

```

2.2.2 Which-key

```

1 (setq which-key-idle-delay 0.5 ;; Default is 1.0
2   which-key-idle-secondary-delay 0.05) ;; Default is nil
3 (setq which-key-allow-multiple-replacements t)
4
5 (after! which-key
6   (pushnew! which-key-replacement-alist
7     '((" . "\\`+?evil[-:]?\\(?:a-\\)?\\(.*\\)") . (nil . " .\\1"))
8     '(("\\`g s" . "\\`evilem--?motion-\\(.*\\)") . (nil .
    ↪ " .\\1")))))

```

2.3 Visual

2.3.1 Font

```

1 (setq doom-font (font-spec :family "Iosevka" :size 14)
2   doom-variable-pitch-font (font-spec :family "Lato")
3   doom-unicode-font (font-spec :family "JuliaMono")
4   doom-big-font (font-spec :family "Iosevka" :size 24)
5   doom-serif-font (font-spec :family "Iosevka Aile" :weight 'light))

```

2.3.2 Theme

1. Catppuccin theme

```
1 ;; (load-theme 'catppuccin t t)
2 (setq doom-theme 'doom-nord-aurora)
3 ;; (setq catppuccin-flavor 'frappe) ;; or 'latte, 'macchiato, or 'mocha
4 ;; (catppuccin-reload)
```

2. Treemacs styling

```
1 (setq doom-themes-treemacs-theme "doom-colors") ; use "doom-colors" for
  ↳ less minimal icon theme
2 (with-eval-after-load 'doom-themes
3   (doom-themes-treemacs-config))
```

2.3.3 Default visual

Transparency and fontification

```
1 ;; Corrects (and improves) org-mode's native fontification.
2 (doom-themes-org-config)
```

```
1 ;; set transparency
2 (set-frame-parameter (selected-frame) 'alpha '(97 97))
3 (add-to-list 'default-frame-alist '(alpha 97 97))
4 (add-to-list 'default-frame-alist '(fullscreen . maximized))
```

2.3.4 Starting image

```
1 (setq fancy-splash-image (expand-file-name "themes/doom-emacs-bw-light.svg"
  ↳ doom-user-dir))
```

2.3.5 Theme Magic

```
1 (package! theme-magic)
```

```
1 (after! theme-magic
2   :commands theme-magic-from-emacs
3   :config
4   (defadvice! theme-magic--auto-extract-16-doom-colors ()
5     :override #'theme-magic--auto-extract-16-colors
6     (list
7       (face-attribute 'default :background)
8       (doom-color 'error)
9       (doom-color 'success)
10      (doom-color 'type)
11      (doom-color 'keywords)
12      (doom-color 'constants)
13      (doom-color 'functions)
14      (face-attribute 'default :foreground)
15      (face-attribute 'shadow :foreground)
16      (doom-blend 'base8 'error 0.1)
17      (doom-blend 'base8 'success 0.1)
18      (doom-blend 'base8 'type 0.1)
19      (doom-blend 'base8 'keywords 0.1)
20      (doom-blend 'base8 'constants 0.1)
21      (doom-blend 'base8 'functions 0.1)
22      (face-attribute 'default :foreground))))
23
```

2.4 Marginalia

```
1 (package! info-colors)
```

```
1 (after! info-colors
2   :commands (info-colors-fontify-node))
3 (add-hook! 'Info-selection-hook 'info-colors-fontify-node)
```

2.5 File Templates

```
1 (set-file-template! "\\tex$" :trigger "__" :mode 'latex-mode)
2 (set-file-template! "\\org$" :trigger "__" :mode 'org-mode)
```

2.6 Editor config

```
1 (setq display-line-numbers-type `relative)
2 (setq-default tab-width 4)
3 (setq byte-compile-warnings '(cl-functions))
```

3 Org-Mode

3.1 Defaults

```
1 (setq org-directory "~/org/"
2     org-agenda-files (list org-directory)           ; Seems like the
    ↪ obvious place.
3     org-use-property-inheritance t                 ; It's convenient
    ↪ to have properties inherited.
4     org-log-done 'time                             ; Having the time a
    ↪ item is done sounds convenient.
5     org-list-allow-alphabetical t                   ; Have a. A. a) A)
    ↪ list bullets.
6     org-catch-invisible-edits 'smart                ; Try not to
    ↪ accidentally do weird stuff in invisible regions.
7     org-export-with-sub-superscripts '{}            ; Don't treat lone
    ↪ _ / ^ as sub/superscripts, require _{} / ^{}.
8     org-export-allow-bind-keywords t                ; Bind keywords can
    ↪ be handy
9     org-image-actual-width '(0.9))                 ; Make the
    ↪ in-buffer display closer to the exported result..#+end_src
```

3.2 Babel

```
1 (setq org-babel-default-header-args
2   '((:session . "none")
3     (:results . "replace")
4     (:exports . "code")
5     (:cache . "no")
6     (:noweb . "no")
7     (:hlines . "no")
8     (:tangle . "no")
9     (:comments . "link")))
```

3.3 Visuals

3.3.1 Org-modern

```
1 (package! org-modern)
```

```
(use-package! org-modern

  :after org

  :hook (org-mode . org-modern-mode)

  :config

  (setq org-modern-star '(" " " " " " " " " " " " " " " " " ")

        org-modern-table-vertical 1

        org-modern-table-horizontal 0.2

        org-modern-list '((43 . " ")

                           (45 . "-")

                           (42 . "•")))

  org-modern-todo-faces

  '(("TODO" :inverse-video t :inherit org-todo)

    ("PROJ" :inverse-video t :inherit +org-todo-project)

    ("STRT" :inverse-video t :inherit +org-todo-active)

    ("[-]" :inverse-video t :inherit +org-todo-active)

    ("HOLD" :inverse-video t :inherit +org-todo-onhold)

    ("WAIT" :inverse-video t :inherit +org-todo-onhold)

    ("[?]" :inverse-video t :inherit +org-todo-onhold)

    ("KILL" :inverse-video t :inherit +org-todo-cancel))
```



```

20      ("NO"      :inverse-video t :inherit +org-todo-cancel))
21  org-modern-footnote
22  (cons nil (cadr org-script-display))
23  org-modern-block-fringe nil
24  org-modern-block-name
25  '(t . t)
26      ("src" "»" "«")
27      ("example" "»-" "-«")
28      ("quote" " " " ")
29      ("export" " " " "))
30  org-modern-progress nil
31  org-modern-priority nil
32  org-modern-horizontal-rule (make-string 36 ?)
33  org-modern-keyword
34  '(t . t)
35      ("title" . " ")
36      ("subtitle" . " ")
37      ("author" . " ")
38      ("email" . #(" " 0 1 (display (raise -0.14))))
39      ("date" . " ")
40      ("property" . " ")
41      ("options" . " ")
42      ("startup" . " ")
43      ("macro" . " ")
44      ("bind" . #(" " 0 1 (display (raise -0.1))))
45      ("bibliography" . " ")
46      ("print_bibliography" . #(" " 0 1 (display (raise -0.1))))
47      ("cite_export" . " ")
48      ("print_glossary" . #(" " 0 1 (display (raise -0.1))))
49      ("glossary_sources" . #(" " 0 1 (display (raise -0.14))))
50      ("include" . " ")
51      ("setupfile" . " ")
52      ("html_head" . " ")
53      ("html" . " ")
54      ("latex_class" . " ")
55      ("latex_class_options" . #(" " 1 2 (display (raise -0.14))))
56      ("latex_header" . " ")
57      ("latex_header_extra" . " ")
58      ("latex" . " ")
59      ("beamer_theme" . " ")

```

```

60     ("beamer_color_theme" . #(" " 1 2 (display (raise -0.12))))
61     ("beamer_font_theme" . " ")
62     ("beamer_header" . " ")
63     ("beamer" . " ")
64     ("attr_latex" . " ")
65     ("attr_html" . " ")
66     ("attr_org" . " ")
67     ("call" . #(" " 0 1 (display (raise -0.15))))
68     ("name" . " ")
69     ("header" . ">")
70     ("caption" . " ")
71     ("results" . " ")))
72 (custom-set-faces! '(org-modern-statistics :inherit
  ↪  org-checkbox-statistics-todo)))

```

```

1 (after! spell-fu
2   (cl-pushnew 'org-modern-tag (alist-get 'org-mode +spell-excluded-faces-alist)))

```

3.3.2 General

```

1 (add-hook 'org-mode-hook #'org-pretty-mode)

```

```

1 (setq org-src-fontify-natively t
2     org-fontify-whole-heading-line t
3     org-fontify-done-headline t
4     org-fontify-quote-and-verse-blocks t
5     org-startup-with-inline-images t
6     org-startup-indented t
7
8     ;; Org styling, hide markup etc.
9     org-pretty-entities t
10    )
11
12 (setq org-ellipsis " "
13     org-hide-leading-stars t

```

```

14     org-priority-highest ?A
15     org-priority-lowest ?E
16     org-priority-faces
17     '((?A . 'all-the-icons-red)
18        (?B . 'all-the-icons-orange)
19        (?C . 'all-the-icons-yellow)
20        (?D . 'all-the-icons-green)
21        (?E . 'all-the-icons-blue)))
22
23
24 (setq org-inline-src-prettify-results '(" " . " "))
25
26 (setq doom-themes-org-fontify-special-tags nil)

```

```

1 (custom-set-faces!
2   '(outline-1 :weight extra-bold :height 1.25)
3   '(outline-2 :weight bold :height 1.15)
4   '(outline-3 :weight bold :height 1.12)
5   '(outline-4 :weight semi-bold :height 1.09)
6   '(outline-5 :weight semi-bold :height 1.06)
7   '(outline-6 :weight semi-bold :height 1.03)
8   '(outline-8 :weight semi-bold)
9   '(outline-9 :weight semi-bold))
10 (custom-set-faces!
11   '(org-document-title :height 1.2))

```

3.3.3 Org-appear

```

1 (package! org-appear)

```

```

1
2 (use-package! org-appear
3   :after org
4   :hook (org-mode . org-appear-mode)
5   :config

```

```

6  (setq org-appear-autoemphasis t
7      org-appear-autosubmarkers t
8      org-appear-autolinks t)
9  ;; for proper first-time setup, `org-appear--set-elements'
10  ;; needs to be run after other hooks have acted.
11  (run-at-time nil nil #'org-appear--set-elements))

```

3.3.4 Ligatures

```

1  (appendq! +ligatures-extra-symbols
2      (list :list_property " "
3          :em_dash      "--"
4          :ellipses     "... "
5          :arrow_right  "->"
6          :arrow_left   "<-"
7          :arrow_lr     "<->"
8          :properties   ":PROPERTIES:"
9          :end           ":END:"
10         :priority_a    "#A"
11         :priority_b    "#B"
12         :priority_c    "#C"
13         :priority_d    "#D"
14         :priority_e    "#E"))
15
16 (defadvice! +org-init-appearance-h--no-ligatures-a ()
17   :after #'org-init-appearance-h
18   (set-ligatures! 'org-mode nil)
19   (set-ligatures! 'org-mode
20       :list_property ":@"
21       :em_dash      "---"
22       :ellipsis     "... "
23       :arrow_right  "->"
24       :arrow_left   "<-"
25       :arrow_lr     "<->"
26       :properties   ":PROPERTIES:"
27       :end           ":END:"
28       :priority_a    "#A"
29       :priority_b    "#B"
30       :priority_c    "#C"))

```

```

31 :priority_d    "[#D]"
32 :priority_e    "[#E]")

```

3.3.5 Latex improvement

```

1 (setq org-highlight-latex-and-related '(native script entities))

```

```

1 (require 'org-src)
2 (add-to-list 'org-src-block-faces '("latex" (:inherit default :extend t)))

```

```

1 ;; (package! org-fragtog)

```

```

1 ;; :hook (org-mode . org-fragtog-mode))

```

3.4 Bullets

```

1 (setq org-list-demote-modify-bullet '(("+" . "-") ("- " . "+") ("*" . "+") ("1."
  ↳ . "a.")))

```

3.5 Agenda

3.5.1 Visual

```

1 (after! org-agenda
2   (setq org-agenda-deadline-faces
3     '((1.001 . error)
4       (1.0 . org-warning)
5       (0.5 . org-upcoming-deadline)
6       (0.0 . org-upcoming-distant-deadline))))

```

3.6 Super-Agenda

```
1 ;; (package! org-super-agenda)
```

3.6.1 Config

```
1 (setq org-agenda-tags-column 0
2     org-agenda-block-separator ?
3     org-agenda-time-grid
4     '((daily today require-timed)
5       (800 1000 1200 1400 1600 1800 2000)
6       " " " " " ")
7     org-agenda-current-time-string
8     " now ")
```

3.6.2 Customize

3.7 Roam

3.7.1 Defaults

```
1
2 (use-package! org-roam
3   :after org
4   :config
5   (setq
6     org-enable-roam-support t
7     org-roam-directory (concat org-directory "/Roam")
8     org-roam-v2-ack t))
```

3.7.2 Daily

```
1
2 (setq org-roam-dailies-directory "daily/")
3
4 (setq org-roam-dailies-capture-templates
5     '(("d" "default" entry
```

```

6      "* %?"
7      :target (file+head "%<%Y-%m-%d>.org"
8                  "#+title: %<%Y-%m-%d>\n"))))

```

3.7.3 Visuals

1. UI and visualization

```

1 (package! org-roam-ui)
2 (package! websocket)

```

```

1
2 (defadvice! doom-modeline--buffer-file-name-roam-aware-a (orig-fun)
3   :around #'doom-modeline-buffer-file-name ; takes no args
4   (if (s-contains-p org-roam-directory (or buffer-file-name ""))
5       (replace-regexp-in-string
6         "\\(?:~\\|\\.*/\\|\\([0-9]\\{4\\}\\|\\([0-9]\\{2\\}\\|\\([0-9]\\{2\\}\\|
7         ↪ \\)[0-9]*-"
8         " (\\1-\\2-\\3) "
9         (subst-char-in-string ?_ ? buffer-file-name))
10      (funcall orig-fun)))
11
12 (use-package! websocket
13   :after org-roam)
14
15 (use-package! org-roam-ui
16   :after org-roam
17   :commands org-roam-ui-open
18   :hook (org-roam . org-roam-ui-mode)
19   :config
20   (setq org-roam-ui-sync-theme t
21         org-roam-ui-follow t
22         org-roam-ui-update-on-save t
23         org-roam-ui-open-on-start t)
24   (require 'org-roam) ; in case autoloaded
25   (defun org-roam-ui-open ()
26     "Ensure the server is active, then open the roam graph."
27     (interactive)
28     (unless org-roam-ui-mode (org-roam-ui-mode 1))
29     (browse-url--browser (format "http://localhost:%d" org-roam-ui-port))))

```

3.8 Ob-async

3.8.1 Julia support

```
1 (add-hook 'ob-async-pre-execute-src-block-hook
2       #'(lambda ()
3       (setq inferior-julia-program-name "/usr/local/bin/julia")))
```

3.8.2 Jupyter Integration

```
1 (setq ob-async-no-async-languages-alist '("jupyter-python" "jupyter-julia"))
```

3.9 Org-Diff

```
1
2 (package! org-diff
3   :recipe (:host github
4           :repo "tecosaur/orgdiff"))
```

```
1
2 (use-package! orgdiff
3   :defer t
4   :config
5   (defun +orgdiff-nicer-change-colours ()
6     (goto-char (point-min))
7     ;; Set red/blue based on whether chameleon is being used
8     (if (search-forward "%% make document follow Emacs theme" nil t)
9       (setq red (substring (doom-blend 'red 'fg 0.8) 1)
10           blue (substring (doom-blend 'blue 'teal 0.6) 1))
11       (setq red "c82829"
12           blue "00618a"))
13   (when (and (search-forward "%DIF PREAMBLE EXTENSION ADDED BY LATEXDIFF" nil
14     ↪ t)
```



```

14         (search-forward "\\RequirePackage{color}" nil t))
15     (when (re-search-forward "definecolor{red}{rgb}{1,0,0}" (cdr
16         ↪ (bounds-of-thing-at-point 'line)) t)
17         (replace-match (format "definecolor{red}{HTML}{%s}" red)))
18     (when (re-search-forward "definecolor{blue}{rgb}{0,0,1}" (cdr
19         ↪ (bounds-of-thing-at-point 'line)) t)
20         (replace-match (format "definecolor{blue}{HTML}{%s}"))))))

```

3.10 Pandoc import

```

1 (package! org-pandoc-import
2   :recipe (:host github
3           :repo "tecosaur/org-pandoc-import"
4           :files ("*.el" "filters" "preprocessors")))

```

```

1
2 (use-package! org-pandoc-import
3   :after org)

```

3.11 Export

3.11.1 Preview

```

1
2 (map! :map org-mode-map
3
4   :localleader
5   :desc "View exported file" "v" #'org-view-output-file)
6
7 (defun org-view-output-file (&optional org-file-path)
8   "Visit buffer open on the first output file (if any) found, using
9   ↪ `org-view-output-file-extensions'"
10  (interactive)
11  (let* ((org-file-path (or org-file-path (buffer-file-name) ""))
12         (dir (file-name-directory org-file-path))

```

```

12     (basename (file-name-base org-file-path))
13     (output-file nil))
14 (dolist (ext org-view-output-file-extensions)
15   (unless output-file
16     (when (file-exists-p
17           (concat dir basename "." ext))
18       (setq output-file (concat dir basename "." ext))))))
19 (if output-file
20   (if (member (file-name-extension output-file)
21             ↪ org-view-external-file-extensions)
22       (browse-url-xdg-open output-file)
23       (pop-to-buffer (or (find-buffer-visiting output-file)
24                         (find-file-noselect output-file))))
25   (message "No exported file found"))))
26 (defvar org-view-output-file-extensions '("pdf" "md" "rst" "txt" "tex" "html")
27   "Search for output files with these extensions, in order, viewing the first
28   ↪ that matches")
29 (defvar org-view-external-file-extensions '("html")
  "File formats that should be opened externally.")

```

3.12 Zotero Integration

```

1 (package! zotxt)

```

```

1 (use-package! zotxt
2   :after org)
3

```

3.13 Org-Chef

```

1 (package! org-chef)

```

```

1 (use-package! org-chef
2   :commands (org-chef-insert-recipe org-chef-get-recipe-from-url))

```

3.14 Bibtex-Integration

3.14.1 Citar

```

1 (package! org-cite-csl-activate :recipe (:host github :repo
↳ "andras-simonyi/org-cite-csl-activate"))

```

```

1 (use-package! citar
2   :no-require
3   :custom
4   (org-cite-global-bibliography '("~/org/Lecture_Notes/MyLibrary.bib"))
5   (org-cite-insert-processor 'citar)
6   (org-cite-follow-processor 'citar)
7   (org-cite-activate-processor 'citar)
8   (citar-bibliography org-cite-global-bibliography)
9   (citar-symbols
10    `((file ,(all-the-icons-faicon "file-o" :face 'all-the-icons-green :v-adjust
↳ -0.1) . " ")
11      (note ,(all-the-icons-material "speaker_notes" :face 'all-the-icons-blue
↳ :v-adjust -0.3) . " ")
12      (link ,(all-the-icons-octicon "link" :face 'all-the-icons-orange :v-adjust
↳ 0.01) . " ")))
13   (citar-symbol-separator " "))
14
15 (use-package! oc-csl
16   :after oc
17   :config
18   (setq org-cite-csl-styles-dir "~/Zotero/styles"))
19
20
21 (after! oc
22   (setq org-cite-export-processors '((t csl))))

```

```

1 (use-package! oc-csl-activate
2   :after org
3   :config
4   (setq org-cite-activate-processor 'csl-activate)
5   (setq org-cite-csl-activate-use-document-style t)
6   (setq org-cite-csl-activate-use-document-locale t)
7   (add-hook 'org-mode-hook
8     (lambda ()
9       (cursor-sensor-mode 1)
10      (org-cite-csl-activate-render-all))))

```

3.14.2 Org-Roam integration

```

1 (package! citar-org-roam :recipe (:host github :repo
  ↳ "emacs-citar/citar-org-roam"))

```

```

1 (use-package! citar-org-roam
2   :after citar org-roam
3   :config (citar-org-roam-mode))
4 (setq org-roam-capture-templates
5   '(("d" "default" plain
6     "%?"
7     :target
8     (file+head
9      "%<%Y%m%d%H%M%S>-${slug}.org"
10     "#+title: ${title}\n")
11     :unnarrowed t)
12   ("n" "literature note" plain
13     "%?"
14     :target
15     (file+head
16      "%(expand-file-name \"literature\" org-roam-directory)/${citekey}.org"
17     "#+title: ${citekey}. ${title}.\n#+created: %U\n#+last_modified:
18     ↳ %U\n\n")
19     :unnarrowed t)))
20 (setq citar-org-roam-capture-template-key "n")

```

3.15 Latex templates

3.15.1 Preview

1. PNG

```
1 (after! org
2   ;; ORG LATEX PREVIEW
3   (setq org-format-latex-options
4     (plist-put org-format-latex-options :background "Transparent"))
5   (setq org-format-latex-options
6     (plist-put org-format-latex-options :scale 1))
7   (setq org-preview-latex-default-process 'dvisvgm)
8   (setq org-preview-latex-image-directory "~/cache/ltximg/")
9   )
```

2. Header

```
1 (setq org-format-latex-header "\\documentclass[12pt]
2 {article}
3 \\usepackage[usenames]{xcolor}
4 \\usepackage{booktabs}
5 \\pagestyle{empty}           % do not remove
6 % The settings below are copied from fullpage.sty
7 \\setlength{\\textwidth}{\\paperwidth}
8 \\addtolength{\\textwidth}{-3cm}
9 \\setlength{\\oddsidemargin}{1.5cm}
10 \\addtolength{\\oddsidemargin}{-2.54cm}
11 \\setlength{\\evensidemargin}{\\oddsidemargin}
12 \\setlength{\\textheight}{\\paperheight}
13 \\addtolength{\\textheight}{-\\headheight}
14 \\addtolength{\\textheight}{-\\headsep}
15 \\addtolength{\\textheight}{-\\footskip}
16 \\addtolength{\\textheight}{-3cm}
17 \\setlength{\\topmargin}{1.5cm}
18 \\addtolength{\\topmargin}{-2.54cm}
19 % my custom stuff
20 \\usepackage{xfrac}
21 \\usepackage{siunitx}
22 \\usepackage{diffcoeff}
23 \\usepackage{nicematrix}
```

```

24 \DeclareMathOperator{\Var}{Var}
25 \DeclareMathOperator{\Cov}{Cov}
26 \DeclareMathOperator{\E}{\mathbb{E}}
27 \DeclareMathOperator*{\argmax}{arg\,,max}
28 \DeclareMathOperator*{\argmin}{arg\,,min}
29 ")
30

```

3.15.2 Article

```

1 (with-eval-after-load 'ox-latex
2 (add-to-list 'org-latex-classes
3   '("article"
4     "\documentclass[c]{article}
5   \usepackage[american]{babel}
6   \usepackage[margin=1.25in]{geometry}
7   \usepackage{parskip}
8   \usepackage{booktabs}
9   \usepackage{float}
10  \usepackage{microtype}
11  \usepackage{graphicx}
12  \usepackage{mathtools}
13  \usepackage{wrapfig}
14  \usepackage{amsthm}
15  \usepackage{amssymb}
16  \usepackage{newpxtext}
17  \usepackage[varbb]{newpxmath}
18  \usepackage{xfrac}
19  \usepackage{siunitx}
20  \usepackage{caption}
21  \captionsetup[labelfont=bf,font={small,singlespacing}}
22  \usepackage{subcaption}
23  \usepackage{cancel}
24  \usepackage{setspace}
25  \usepackage{xcolor}
26  \usepackage{diffcoeff}
27  \usepackage{nicematrix}
28  \usepackage{enumitem}
29  \usepackage{acronym}

```

```

30 \usepackage{xurl}
31 \onehalfspacing{}
32 \DeclareMathOperator{\Var}{Var}
33 \DeclareMathOperator{\cov}{Cov}
34 \DeclareMathOperator{\E}{\mathbb{E}}
35 \DeclareMathOperator*{\argmax}{arg\,,max}
36 \DeclareMathOperator*{\argmin}{arg\,,min}
37 \newcommand{\Et}[2]{\E_{#2} \left[#1\right]}
38 \newcommand{\Covt}[3]{\cov_{#3}\left(#1, #2\right)}
39 \newcommand{\Vart}[2]{\Var_{#2} \left[#1\right]}
40 \DeclarePairedDelimiter\abs{\lvert}{\rvert}
41 \DeclarePairedDelimiter\norm{\lVert}{\rVert}
42 \DeclarePairedDelimiterX\innerp[2]{\langle}{\rangle}{#1,#2}
43 \DeclarePairedDelimiterX\braket[3]{\langle}{\rangle}%
44 {#1\,,\delimsize\vert\,,\mathopen{ }#2\,,\delimsize\vert\,,\mathopen{ }#3}
45 \providecommand\given{}
46 \DeclarePairedDelimiterXPP\Prob[1]{\mathbb{P}} (){}{
47 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{ }
48 #1}
49 \DeclarePairedDelimiterXPP\condE[1]{\E} (){}{
50 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{ }
51 #1}
52 \DeclarePairedDelimiterXPP\condVar[2]{\Var} (){}{
53 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{ }
54 #1,#2}
55 \DeclarePairedDelimiterXPP\condCov[2]{\cov} (){}{
56 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{ }
57 #1,#2}
58 \theoremstyle{plain}% default
59 \newtheorem{thm}{Theorem}
60 \newtheorem{lem}[thm]{Lemma}
61 \newtheorem{prop}[thm]{Proposition}
62 \newtheorem*{cor}{Corollary}
63 \theoremstyle{definition}
64 \newtheorem{defn}{Definition}
65 \newtheorem{exmp}{Example}
66 \providecommand*\defnautorefname{Definition}
67 \theoremstyle{remark}
68 \newtheorem*{rem}{Remark}
69 \newtheorem*{note}{Note}

```

```

70  \newtheorem{case}{Case}
71
72  \renewcommand{\leq}{\leqslant}
73  \renewcommand{\geq}{\geqslant}
74  \definecolor{bgcolorminted}{HTML}{2E3440}
75  \usepackage{hyperref}
76  \usepackage[]{cleveref}
77  [NO-DEFAULT-PACKAGES]
78  [PACKAGES]
79  [EXTRA]
80  \usemintedstyle{nord}"
81      ("\\section{%s}" . "\\section*{%s}")
82      ("\\subsection{%s}" . "\\subsection*{%s}")
83      ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
84      ("\\paragraph{%s}" . "\\paragraph*{%s}"))))

```

3.15.3 Beamer

```

1  (setq org-beamer-frame-level 2)

```

```

1  (setq org-beamer-theme "[progressbar=frametitle, titleformat=smallcaps,
    ↪ numbering=fraction]metropolis")

```

Define Beamer class:

```

1
2  (with-eval-after-load 'ox-latex
3  (add-to-list 'org-latex-classes
4      ("beamer"
5       "\\documentclass[c]{beamer}
6       \\usepackage[american]{babel}
7       \\usetheme[progressbar=frametitle, titleformat=smallcaps,
    ↪ numbering=fraction]{metropolis}
8       \\usepackage{booktabs}
9       \\usepackage{float}
10      \\usepackage{mathtools}

```



```

11 \usepackage{amsthm}
12 \usepackage{amssymb}
13 \usepackage[varbb]{newpxmath}
14 \usepackage[]{xfrac}
15 \usepackage{siunitx}
16 \usepackage{graphicx}
17 \usepackage{caption}
18 \captionsetup{labelfont=bf,font={small,singlespacing}}
19 \usepackage{subcaption}
20 \usepackage{cancel}
21 \usepackage{setspace}
22 \usepackage{xcolor}
23 \usepackage{diffcoeff}
24 \usepackage{nicematrix}
25 \usepackage{acronym}
26 \usepackage{appendixnumberbeamer}
27 \usepackage{dirtytalk}
28 \usepackage{xurl}
29 \DeclareMathOperator{\Var}{Var}
30 \DeclareMathOperator{\Cov}{Cov}
31 \DeclareMathOperator{\E}{\mathbb{E}}
32 \DeclareMathOperator*{\argmax}{arg\,,max}
33 \DeclareMathOperator*{\argmin}{arg\,,min}
34 \newcommand{\Et}[2]{\E_{#2} \left[#1\right]}
35 \newcommand{\Covt}[3]{\Cov_{#3} \left(#1, #2\right)}
36 \newcommand{\Vart}[2]{\Var_{#2} \left[#1\right]}
37 \DeclarePairedDelimiter\abs{\lvert}{\rvert}
38 \DeclarePairedDelimiter\norm{\lVert}{\rVert}
39 \DeclarePairedDelimiterX\innerp[2]{\langle}{\rangle}{#1,#2}
40 \DeclarePairedDelimiterX\braket[3]{\langle}{\rangle}%
41 {#1\,,\delimsize\vert\,,\mathopen{#2\,,\delimsize\vert\,,\mathopen{#3}}
42 \providecommand\given{}
43 \DeclarePairedDelimiterXPP\Prob[1]{\mathbb{P}} (){}{
44 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{}}
45 #1}
46 \DeclarePairedDelimiterXPP\condE[1]{\E} (){}{
47 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{}}
48 #1}
49 \DeclarePairedDelimiterXPP\condVar[2]{\Var} (){}{
50 \renewcommand\given{\nonscript\:\delimsize\vert\nonscript\:\mathopen{}}

```

```

51 #1,#2}
52 \\DeclarePairedDelimiterXPP\\condCov[2]{\\cov} (){}{
53 \\renewcommand\\given{\\nonscript\\:\\delimsiz\\vert\\nonscript\\:\\mathopen{}}
54 #1,#2}
55 \\theoremstyle{plain}% default
56 \\newtheorem{thm}{Theorem}
57 \\newtheorem{lem}[thm]{Lemma}
58 \\newtheorem{prop}[thm]{Proposition}
59 \\newtheorem*{cor}{Corollary}
60 \\theoremstyle{definition}
61 \\newtheorem{defn}{Definition}
62 \\newtheorem{exmp}{Example}
63 \\providecommand*{\\defnautorefname}{Definition}
64 \\theoremstyle{remark}
65 \\newtheorem*{rem}{Remark}
66 \\newtheorem{case}{Case}
67
68
69 \\definecolor{dblue}{HTML}{2E3440}
70 \\definecolor{umber}{HTML}{8FBCBB}
71 \\definecolor{alertcolor}{HTML}{BF616A}
72 \\definecolor{examplecolor}{HTML}{EBCB8B}
73
74 \\definecolor{pale}{HTML}{ECEFF4}
75 \\definecolor{bluish}{HTML}{88C0D0}
76 \\definecolor{cream}{HTML}{D8DEE9}
77 \\definecolor{bgcolorminted}{HTML}{2E3440}
78 \\setbeamercolor{progress bar}{fg=bluish,bg=cream}
79 \\setbeamercolor{frametitle}{fg=umber,bg=pale}
80 \\setbeamercolor{normal text}{fg=dblue,bg=pale}
81 \\setbeamercolor{alerted text}{fg=alertcolor,bg=pale}
82 \\setbeamercolor{example text}{fg=examplecolor}
83 \\setbeamercovered{dynamic}
84
85 \\usecolortheme{rose}
86 \\usepackage{hyperref}
87 \\usepackage{cleveref}
88 [NO-DEFAULT-PACKAGES]
89 [PACKAGES]
90 [EXTRA]

```

```

91 \\usemintedstyle{nord}"
92     ("\\section{%s}" . "\\section*{%s}")
93     ("\\subsection{%s}" . "\\subsection*{%s}")
94     ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
95     ("\\paragraph{%s}" . "\\paragraph*{%s}")
96     ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))

```

3.15.4 Export

```

1 (setq org-latex-pdf-process '("LC_ALL=en_US.UTF-8 latexmk -xelatex -f
  ↪ -shell-escape -interaction=nonstopmode -output-directory=%o %f"))

```

```

1 (setq org-latex-tables-booktabs t
2     org-latex-hyperref-template "\\providecolor{url}{HTML}{81a1c1}
3   \\providecolor{link}{HTML}{d08770}
4   \\providecolor{cite}{HTML}{d08770}
5   \\hypersetup{
6     pdfauthor={%a},
7     pdftitle={%t},
8     pdfkeywords={%k},
9     pdfsubject={%d},
10    pdfcreator={%c},
11    pdflang={%L},
12    breaklinks=true,
13    colorlinks=true,
14    linkcolor=link,
15    urlcolor=url,
16    citecolor=cite
17  }
18  \\urlstyle{same}
19  %% hide links styles in toc
20  \\NewCommandCopy{\\oldtoc}{\\tableofcontents}
21  \\renewcommand{\\tableofcontents}{\\begingroup\\hypersetup{hidelinks}\\oldtoc\\e_
  ↪ ndgroup}
22  "
23  org-latex-reference-command "\\cref{%s}")

```

1. Preview

```
1 ;; Use pdf-tools to open PDF files
2 (setq TeX-view-program-selection '((output-pdf "PDF Tools"))
3   TeX-source-correlate-start-server t)
```

```
1 ;; Update PDF buffers after successful LaTeX runs
2 (add-hook! 'TeX-after-compilation-finished-functions
3   #'TeX-revert-document-buffer)
```

2. Code blocks

```
1 (setq org-latex-listings 'minted
2   org-latex-packages-alist '((" " "minted")))
3 (setq org-latex-minted-options '(("breaklines" "true")
4   ("breakanywhere" "true")
5   ("bgcolor" "bgcolorminted")
6   ("linenos" "true")))
```

3.16 Capture

3.16.1 Doct

```
1 (package! doct)
```

Prettify the captures:

```
1 (after! org-capture
2
3   (defun +doct-icon-declaration-to-icon (declaration)
4     "Convert :icon declaration to icon"
5     (let ((name (pop declaration))
6       (set (intern (concat "all-the-icons-" (plist-get declaration :set))))
7       (face (intern (concat "all-the-icons-" (plist-get declaration
8         ↪ :color)))))
9       (v-adjust (or (plist-get declaration :v-adjust) 0.01)))
10    (apply set `((name :face ,face :v-adjust ,v-adjust)))))
```

```

10
11 (defun +doct-iconify-capture-templates (groups)
12   "Add declaration's :icon to each template group in GROUPS."
13   (let ((templates (doct-flatten-lists-in groups)))
14     (setq doct-templates (mapcar (lambda (template)
15                                   (when-let* ((props (nthcdr (if (= (length
16                                     ↪ template) 4) 2 5) template))
17                                     (spec (plist-get (plist-get
18                                       ↪ props :doct) :icon)))
19                                     (setf (nth 1 template) (concat
20                                       ↪ (+doct-icon-declaration-to-icon spec)
21                                       ↪ "\t"
22                                       ↪ (nth 1
23                                         ↪ template)
24                                       ↪ te))))
25                                   template)
26     templates))))
27
28 (setq doct-after-conversion-functions '(+doct-iconify-capture-templates))
29
30 (defvar +org-capture-recipies "~/org/recipies.org")
31
32 (defun set-org-capture-templates ()
33   (setq org-capture-templates
34     (doct `(("Personal todo" :keys "t"
35              :icon ("checklist" :set "octicon" :color "green")
36              :file +org-capture-todo-file
37              :prepend t
38              :headline "Inbox"
39              :type entry
40              :template ("* TODO %?"
41                        "%i %a"))
42              ("Personal note" :keys "n"
43              :icon ("sticky-note-o" :set "faicon" :color "green")
44              :file +org-capture-todo-file
45              :prepend t
46              :headline "Inbox"
47              :type entry
48              :template ("* %?"
49                        "%i %a"))

```

```

45     ("Email" :keys "e"
46       :icon ("envelope" :set "faicon" :color "blue")
47       :file +org-capture-todo-file
48       :prepend t
49       :headline "Inbox"
50       :type entry
51       :template ("* TODO %^{type|reply to|contact} %\\3 %? :email:"
52                 "Send an email %^{urgancy|soon|ASAP|anon|at some
53                   ↳ point|eventually} to %^{recipiant}"
54                   "about %^{topic}"
55                   "%U %i %a"))
56 ("Meeting" :keys "m"
57   :icon ("users" :set "faicon")
58   :file "~/org/meeting.org"
59   :prepend t
60   :headline "Meetings"
61   :type entry
62   :template ("* %^{Topic}
63             + Attendees: %^{Attendees},Leo
64             + Date: %U
65             ** Notes
66             + %?
67             ** Actions
68             + [ ] ")
69 ("Interesting" :keys "i"
70   :icon ("eye" :set "faicon" :color "lcyan")
71   :file +org-capture-todo-file
72   :prepend t
73   :headline "Interesting"
74   :type entry
75   :template ("* [ ] %^{desc}%? :%{i-type}:"
76             "%i %a")
77   :children (("Webpage" :keys "w"
78                 :icon ("globe" :set "faicon" :color "green")
79                 :desc "%(org-cliplink-capture) "
80                 :i-type "read:web")
81             ("Article" :keys "a"
82                 :icon ("file-text" :set "octicon" :color "yellow")
83                 :desc ""
84                 :i-type "read:reaserch"))

```

```

84         ("\\tRecipie" :keys "r"
85           :icon ("spoon" :set "faicon" :color "dorange")
86           :file +org-capture-recipes
87           :headline "Unsorted"
88           :template "%(org-chef-get-recipe-from-url)")
89         ("Information" :keys "i"
90           :icon ("info-circle" :set "faicon" :color "blue")
91           :desc ""
92           :i-type "read:info")
93         ("Idea" :keys "I"
94           :icon ("bubble_chart" :set "material" :color
95             ↪ "silver")
96           :desc ""
97           :i-type "idea"))))
98     ("Tasks" :keys "k"
99       :icon ("inbox" :set "octicon" :color "yellow")
100       :file +org-capture-todo-file
101       :prepend t
102       :headline "Tasks"
103       :type entry
104       :template ("* TODO %? %^G%{extra}"
105                 "%i %a")
106       :children (("General Task" :keys "k"
107                   :icon ("inbox" :set "octicon" :color "yellow")
108                   :extra "")
109                 ("Task with deadline" :keys "d"
110                   :icon ("timer" :set "material" :color "orange"
111                     ↪ :v-adjust -0.1)
112                   :extra "\\nDEADLINE: %^{Deadline:}t")
113                 ("Scheduled Task" :keys "s"
114                   :icon ("calendar" :set "octicon" :color "orange")
115                   :extra "\\nSCHEDULED: %^{Start time:}t"))))
116     ("Project" :keys "p"
117       :icon ("repo" :set "octicon" :color "silver")
118       :prepend t
119       :type entry
120       :headline "Inbox"
121       :template ("* %{time-or-todo} %?"
122                 "%i"
123                 "%a")

```

```

122         :file ""
123         :custom (:time-or-todo "")
124         :children (("Project-local todo" :keys "t"
125             :icon ("checklist" :set "octicon" :color "green")
126             :time-or-todo "TODO"
127             :file +org-capture-project-todo-file)
128             ("Project-local note" :keys "n"
129             :icon ("sticky-note" :set "faicon" :color
130                 ↵ "yellow")
131             :time-or-todo "%U"
132             :file +org-capture-project-notes-file)
133             ("Project-local changelog" :keys "c"
134             :icon ("list" :set "faicon" :color "blue")
135             :time-or-todo "%U"
136             :heading "Unreleased"
137             :file +org-capture-project-changelog-file)))
138     ("\\tCentralised project templates"
139     :keys "o"
140     :type entry
141     :prepend t
142     :template ("* %{:time-or-todo} %?"
143         "%i"
144         "%a")
145     :children (("Project todo"
146         :keys "t"
147         :prepend nil
148         :time-or-todo "TODO"
149         :heading "Tasks"
150         :file +org-capture-central-project-todo-file)
151         ("Project note"
152         :keys "n"
153         :time-or-todo "%U"
154         :heading "Notes"
155         :file +org-capture-central-project-notes-file)
156         ("Project changelog"
157         :keys "c"
158         :time-or-todo "%U"
159         :heading "Unreleased"
160         :file +org-capture-central-project-changelog-file ↵
161         ↵ )))))))

```



```

160
161 (set-org-capture-templates)
162 (unless (display-graphic-p)
163   (add-hook! 'server-after-make-frame-hook
164     (defun org-capture-reinitialise-hook ()
165       (when (display-graphic-p)
166         (set-org-capture-templates)
167         (remove-hook 'server-after-make-frame-hook
168           #'org-capture-reinitialise-hook))))))

```

```

1 (defun org-mks-pretty (table title &optional prompt specials)
2   "Select a member of an alist with multiple keys. Prettified.
3
4 TABLE is the alist which should contain entries where the car is a string.
5 There should be two types of entries.
6
7 1. prefix descriptions like (\"a\" \"Description\")
8   This indicates that `a' is a prefix key for multi-letter selection, and
9   that there are entries following with keys like \"ab\", \"ax\"...
10
11 2. Select-able members must have more than two elements, with the first
12   being the string of keys that lead to selecting it, and the second a
13   short description string of the item.
14
15 The command will then make a temporary buffer listing all entries
16 that can be selected with a single key, and all the single key
17 prefixes. When you press the key for a single-letter entry, it is selected.
18 When you press a prefix key, the commands (and maybe further prefixes)
19 under this key will be shown and offered for selection.
20
21 TITLE will be placed over the selection in the temporary buffer,
22 PROMPT will be used when prompting for a key. SPECIALS is an
23 alist with (\"key\" \"description\") entries. When one of these
24 is selected, only the bare key is returned."
25 (save-window-excursion
26   (let ((inhibit-quit t)
27         (buffer (org-switch-to-buffer-other-window "*Org Select*"))
28         (prompt (or prompt "Select: ")))

```

```

29     case-fold-search
30     current)
31 (unwind-protect
32   (catch 'exit
33     (while t
34       (setq-local evil-normal-state-cursor (list nil))
35       (erase-buffer)
36       (insert title "\n\n")
37       (let ((des-keys nil)
38             (allowed-keys '("\C-g"))
39             (tab-alternatives '("\s" "\t" "\r"))
40             (cursor-type nil))
41         ;; Populate allowed keys and descriptions keys
42         ;; available with CURRENT selector.
43         (let ((re (format "\\~%s\\(.\\)\\\\"
44                           (if current (regexp-quote current) "")))
45             (prefix (if current (concat current " ") "")))
46           (dolist (entry table)
47             (pcase entry
48               ;; Description.
49               (`(, (and key (pred (string-match re))) ,desc)
50                (let ((k (match-string 1 key)))
51                  (push k des-keys)
52                  ;; Keys ending in tab, space or RET are equivalent.
53                  (if (member k tab-alternatives)
54                      (push "\t" allowed-keys)
55                      (push k allowed-keys))
56                  (insert (propertize prefix 'face
57                                     ↪ 'font-lock-comment-face) (propertize k 'face 'bold)
58                           (propertize ">" 'face 'font-lock-comment-face) " "
59                           desc "..." "\n"))))
60               ;; Usable entry.
61               (`(, (and key (pred (string-match re))) ,desc . ,_)
62                (let ((k (match-string 1 key)))
63                  (insert (propertize prefix 'face
64                                     ↪ 'font-lock-comment-face) (propertize k 'face 'bold)
65                           " " desc "\n")
66                  (push k allowed-keys)))
67                (_ nil))))
68         ;; Insert special entries, if any.

```

```

64         (when specials
65             (insert "                \n")
66             (pcase-dolist (`(,key ,description) specials)
67                 (insert (format "%s    %s\n" (propertize key 'face '(bold
68                     ↪ all-the-icons-red)) description))
69                 (push key allowed-keys)))
70             ;; Display UI and let user select an entry or
71             ;; a sub-level prefix.
72             (goto-char (point-min))
73             (unless (pos-visible-in-window-p (point-max))
74                 (org-fit-window-to-buffer))
75             (let ((pressed (org--mks-read-key allowed-keys
76                 prompt
77                 (not (pos-visible-in-window-p
78                     ↪ (1- (point-max)))))))
79                 (setq current (concat current pressed))
80                 (cond
81                     ((equal pressed "\C-g") (user-error "Abort"))
82                     ;; Selection is a prefix: open a new menu.
83                     ((member pressed des-keys)
84                      ;; Selection matches an association: return it.
85                      ((let ((entry (assoc current table)))
86                          (and entry (throw 'exit entry))))
87                      ;; Selection matches a special entry: return the
88                      ;; selection prefix.
89                      ((assoc current specials) (throw 'exit current))
90                      (t (error "No entry available"))))))
91             (when buffer (kill-buffer buffer))))))
92 (advice-add 'org-mks :override #'org-mks-pretty)

```

4 Company

Improve the history size:

```

1  (after! company
2    (setq company-idle-delay 0.2
3          company-minimum-prefix-length 3)
4    (setq company-show-numbers t)) ;; make aborting less annoying.
5  (setq-default history-length 1000)

```

```
6 (setq-default prescient-history-length 1000)
```

```
1 (set-company-backend!  
2   '(text-mode  
3     markdown-mode  
4     gfm-mode)  
5   '(:seperate  
6     company-ispell  
7     company-files  
8     company-yasnippet))
```

5 LSP

5.1 Digestif

```
1 ;; (after! lsp-mode  
2 ;;   (setq lsp-tex-server 'digestif))
```

5.2 LTeX

```
1 ;; (package! lsp-ltex)  
2 ;; (package! eglot-ltex :recipe (:host github :repo  
3   ↪ "emacs-languagetool/eglot-ltex"))
```

```
1 ;; (use-package! lsp-ltex  
2 ;;   :hook (text-mode . (lambda ()  
3 ;;                         (require 'lsp-ltex)  
4 ;;                         (lsp))) ; or lsp-deferred  
5 ;;   :init  
6 ;;   (setq lsp-ltex-version "15.2.0")) ; make sure you have set this, see below  
7 ;; (use-package! eglot-ltex  
8 ;;   :hook (text-mode . (lambda ()
```

```

9 ;;                                     (require 'eglot-ltex)
10 ;;                                     (call-interactively #'eglot)))
11 ;; :init
12 ;; (setq eglot-languagetool-server-path "/opt/homebrew/Cellar/ltex-ls/15.2.0"))

```

6 VLFI

```

1 (package! vlfi)

```

```

1 (use-package! vlfi-setup
2   :defer-incrementally vlfi-tune vlfi-base vlfi-write vlfi-search vlfi-occur
   ↪ vlfi-follow vlfi-ediff vlfi)

```

7 PDF-Tools

7.1 Fix annotation bug

```

1 (defun my-fix-tablist ()
2   (interactive)
3   (unload-feature 'tablist-filter t)
4   (load-file (find-library-name "tablist-filter")))

```

7.2 Dark mode

```

1 ;; (add-hook 'pdf-tools-enabled-hook 'pdf-view-midnight-minor-mode)

```

8 Option key Fix

```

1 (defun switch-left-and-right-option-keys ()
2   "Switch left and right option keys.
3   On some external keyboards the left and right option keys are swapped,
4   this command switches the keys so that they work as expected."
5   (interactive)
6   (let ((current-left mac-option-modifier)
7         (current-right mac-right-option-modifier))
8     (setq mac-option-modifier current-right
9           mac-right-option-modifier current-left)))

```

9 Centaur

```

1 ;; (after! centaur-tabs
2 ;;   (centaur-tabs-mode -1)
3 ;;   (setq centaur-tabs-height 36
4 ;;         centaur-tabs-set-icons t
5 ;;         centaur-tabs-modified-marker "o"
6 ;;         centaur-tabs-close-button "x"
7 ;;         centaur-tabs-set-bar 'above
8 ;;         centaur-tabs-gray-out-icons 'buffer)
9 ;;   (centaur-tabs-change-fonts "P22 Underground Book" 160))
10 ;; (setq x-underline-at-descent-line t)

```

10 Email

10.1 mu4e

```

1 ;; add to $DOOMDIR/config.el
2 (after! mu4e
3   (setq sendmail-program (executable-find "msmtp")
4         send-mail-function #'smtpmail-send-it
5         message-sendmail-f-is-evil t
6         message-sendmail-extra-arguments '("--read-envelope-from")
7         message-send-mail-function #'message-send-mail-with-sendmail)
8   ;; how often to call it in seconds:
9   (setq mu4e-sent-messages-behavior 'sent ;; Save sent messages

```

```

10     mu4e-headers-auto-update t                ; avoid to type `g' to update
11     mu4e-compose-signature-auto-include nil    ; I don't want a message
        ↳ signature
12     mu4e-use-fancy-chars t                    ; allow fancy icons for mail
        ↳ threads
13     mu4e-context-policy 'pick-first           ;; Start with the first context
14     mu4e-compose-context-policy 'ask) ;; Always ask which context to use
        ↳ when composing a new mail
15     (setq mu4e-update-interval (* 1 60))
16     (setq mu4e-attachment-dir "~/Downloads")
17     (set-email-account! "gmail"
18         '((mu4e-sent-folder      . "/gmail/Sent Mail")
19           (mu4e-drafts-folder    . "/gmail/Drafts")
20           (mu4e-trash-folder     . "/gmail/Trash")
21           (mu4e-refile-folder    . "/gmail/All Mail")
22           (smtpmail-smtp-user    . "leoaparis@gmail.com")
23           (user-mail-address     . "leoaparis@gmail.com") ;;
        ↳ only needed for mu < 1.4
24           (mu4e-compose-signature . "---\nLeo Aparisi de Lannoy"))
25     t)
26     (set-email-account! "U Chicago"
27         '((mu4e-sent-folder      . "/UChicago/Sent Mail")
28           (mu4e-drafts-folder    . "/UChicago/Drafts")
29           (mu4e-trash-folder     . "/UChicago/Trash")
30           (mu4e-refile-folder    . "/UChicago/All Mail")
31           (smtpmail-smtp-user    .
        ↳ "laparisidelannoy@uchicago.edu")
32           (user-mail-address     .
        ↳ "laparisidelannoy@uchicago.edu") ;; only needed
        ↳ for mu < 1.4
33           (mu4e-compose-signature . "---\nLeo Aparisi de Lannoy"))
34     t)
35     (setq +mu4e-gmail-accounts '(("leoaparis@gmail.com" . "/gmail")))
36     (setq mu4e-compose-dont-reply-to-self t)
37     (setq mu4e-compose-format-flowed t)
38     ;; Add a unified inbox shortcut
39     (add-to-list
40       'mu4e-bookmarks
41       '(:name "Unified inbox" :query "maildir:/*inbox/" :key ?i) t)
42     (add-hook 'mu4e-compose-mode-hook 'company-mode)

```

43

)

10.2 Notification

```

1 (mu4e-alert-set-default-style 'notifier)
2 (add-hook 'after-init-hook #'mu4e-alert-enable-notifications)

```

11 RSS

```

1 (add-hook! 'elfeed-search-mode-hook #'elfeed-update)
2 (after! elfeed
3   (setq elfeed-goodies/entry-pane-position 'bottom)
4   (setq rmh-elfeed-org-files '("~/org/elfeed.org")))

```

11.1 Visual

```

1 (after! elfeed
2   (setq elfeed-search-filter "@1-week-ago +unread"
3         elfeed-search-print-entry-function '+rss/elfeed-search-print-entry
4         elfeed-search-title-min-width 80
5         elfeed-show-entry-switch #'switch-to-buffer
6         elfeed-show-entry-delete #' +rss/delete-pane
7         elfeed-show-refresh-function #' +rss/elfeed-show-refresh--better-style
8         shr-max-image-proportion 0.6)
9
10  (add-hook! 'elfeed-show-mode-hook (hide-mode-line-mode 1))
11  (add-hook! 'elfeed-search-update-hook #'hide-mode-line-mode)
12
13  (defface elfeed-show-title-face '((t (:weight ultrabold :slant italic :height
14    ↳ 1.5)))
15    "title face in elfeed show buffer"
16    :group 'elfeed)
17  (defface elfeed-show-author-face `((t (:weight light)))
18    "title face in elfeed show buffer"

```



```

18   :group 'elfeed)
19   (set-face-attribute 'elfeed-search-title-face nil
20                       :foreground 'nil
21                       :weight 'light)
22
23   (defadvice! +rss-elfeed-wrap-h-nicer ()
24     "Enhances an elfeed entry's readability by wrapping it to a width of
25     `fill-column' and centering it with `visual-fill-column-mode'."
26     :override #'+rss-elfeed-wrap-h
27     (setq-local truncate-lines nil
28                 shr-width 140
29                 visual-fill-column-center-text t
30                 default-text-properties '(line-height 1.2))
31     (let ((inhibit-read-only t)
32           (inhibit-modification-hooks t))
33       (setq-local shr-current-font '(:family "Lato" :height 1.2))
34       (set-buffer-modified-p nil)))
35
36   (defun +rss/elfeed-search-print-entry (entry)
37     "Print ENTRY to the buffer."
38     (let* ((elfeed-goodies/tag-column-width 40)
39            (elfeed-goodies/feed-source-column-width 30)
40            (title (or (elfeed-meta entry :title) (elfeed-entry-title entry) ""))
41            (title-faces (elfeed-search--faces (elfeed-entry-tags entry)))
42            (feed (elfeed-entry-feed entry))
43            (feed-title
44              (when feed
45                (or (elfeed-meta feed :title) (elfeed-feed-title feed)))))
46            (tags (mapcar #'symbol-name (elfeed-entry-tags entry)))
47            (tags-str (concat (mapconcat 'identity tags ", ")))
48            (title-width (- (window-width) elfeed-goodies/feed-source-column-width
49                           elfeed-goodies/tag-column-width 4))
50
51            (tag-column (elfeed-format-column
52                        tags-str (elfeed-clamp (length tags-str)
53                                                elfeed-goodies/tag-column-width
54                                                elfeed-goodies/tag-column-width)
55                        :left))
56            (feed-column (elfeed-format-column
57                          feed-title (elfeed-clamp
58                                     elfeed-goodies/feed-source-column-width

```

```

58                                     elfeed-goodies/feed-source-col_
                                     ↪ umn-width
59                                     elfeed-goodies/feed-source-col_
                                     ↪ umn-width)
60                                     :left)))
61
62     (insert (propertize feed-column 'face 'elfeed-search-feed-face) " ")
63     (insert (propertize tag-column 'face 'elfeed-search-tag-face) " ")
64     (insert (propertize title 'face title-faces 'kbd-help title))
65     (setq-local line-spacing 0.2)))
66
67 (defun +rss/elfeed-show-refresh--better-style ()
68   "Update the buffer to match the selected entry, using a mail-style."
69   (interactive)
70   (let* ((inhibit-read-only t)
71          (title (elfeed-entry-title elfeed-show-entry))
72          (date (seconds-to-time (elfeed-entry-date elfeed-show-entry)))
73          (author (elfeed-meta elfeed-show-entry :author))
74          (link (elfeed-entry-link elfeed-show-entry))
75          (tags (elfeed-entry-tags elfeed-show-entry))
76          (tagsstr (mapconcat #'symbol-name tags ", "))
77          (nicedate (format-time-string "%a, %e %b %Y %T %Z" date))
78          (content (elfeed-deref (elfeed-entry-content elfeed-show-entry)))
79          (type (elfeed-entry-content-type elfeed-show-entry))
80          (feed (elfeed-entry-feed elfeed-show-entry))
81          (feed-title (elfeed-feed-title feed))
82          (base (and feed (elfeed-compute-base (elfeed-feed-url feed)))))
83     (erase-buffer)
84     (insert "\n")
85     (insert (format "%s\n\n" (propertize title 'face 'elfeed-show-title-face)))
86     (insert (format "%s\t" (propertize feed-title 'face
87                               ↪ 'elfeed-search-feed-face)))
88     (when (and author elfeed-show-entry-author)
89       (insert (format "%s\n" (propertize author 'face
90                               ↪ 'elfeed-show-author-face))))
91     (insert (format "%s\n\n" (propertize nicedate 'face
92                               ↪ 'elfeed-log-date-face)))
93     (when tags
94       (insert (format "%s\n"
95                       (propertize tagsstr 'face 'elfeed-search-tag-face))))

```

```

93      ;; (insert (propertize "Link: " 'face 'message-header-name))
94      ;; (elfeed-insert-link link link)
95      ;; (insert "\n")
96      (cl-loop for enclosure in (elfeed-entry-enclosures elfeed-show-entry)
97                do (insert (propertize "Enclosure: " 'face 'message-header-name))
98                do (elfeed-insert-link (car enclosure))
99                do (insert "\n"))
100      (insert "\n")
101      (if content
102        (if (eq type 'html)
103            (elfeed-insert-html content base)
104            (insert content))
105        (insert (propertize "(empty)\n" 'face 'italic)))
106      (goto-char (point-min)))
107
108    )

```