Config

Leo Aparisi de Lannoy

February 21, 2023

Contents

1 Compilation

```
;; brew tap railwaycat/emacsmacport
;; brew install emacs-mac --with-mac-metal --with-natural-title-bar --with-native-compilation
↪  --with-xwidget
```

2 Basic

2.1 ID

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

2.2 Good defaults

```
(setq-default
 delete-by-moving-to-trash t                      ; Delete files to trash
 window-combination-resize t                      ; take new window space from all other windows (not
 ↪  just current)
 x-stretch-cursor t)                              ; Stretch cursor to the glyph width

(setq undo-limit 80000000                         ; Raise undo-limit to 80Mb
      evil-want-fine-undo t                       ; By default while in insert all changes are one
      ↪  big blob. Be more granular
      auto-save-default t                         ; Nobody likes to loose work, I certainly don't
      truncate-string-ellipsis "..."
      scroll-margin 2)                            ; It's nice to maintain a little margin

(display-time-mode 1)                             ; Enable time in the mode-line

(unless (string-match-p "^Power N/A" (battery))   ; On laptops...
  (display-battery-mode 1))                       ; it's nice to know how much power you have
```

```
(global-subword-mode 1)                           ; Iterate through CamelCase words
```

2.2.1 Browser

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

2.2.2 Which-key

```
(setq which-key-idle-delay 0.5 ;; Default is 1.0
      which-key-idle-secondary-delay 0.05) ;; Default is nil
(setq which-key-allow-multiple-replacements t)

(after! which-key
  (pushnew! which-key-replacement-alist
            '((""       . "\\`+?evil[-:]?\\(?:a-\\)?\\(.*\\)") . (nil . "·\\1"))
            '(("\\`g s" . "\\`evilem--?motion-\\(.*\\)")       . (nil . "·\\1"))))
```

2.3 Visual

2.3.1 Font

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

2.3.2 Theme

1. Catppuccin theme

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

2. Treemacs styling

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

2.3.3 Default visual

Transparency and fontification

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

```
;; set transparency
(set-frame-parameter (selected-frame) 'alpha '(99 99))
(add-to-list 'default-frame-alist '(alpha 99 99))
(add-to-list 'default-frame-alist '(fullscreen . maximized))
```

2.3.4 Starting image

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

2.3.5 Theme Magic

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

```
(after! theme-magic
  :commands theme-magic-from-emacs
  :config
  (defadvice! theme-magic--auto-extract-16-doom-colors ()
    :override #'theme-magic--auto-extract-16-colors
    (list
     (face-attribute 'default :background)
     (doom-color 'error)
     (doom-color 'success)
     (doom-color 'type)
     (doom-color 'keywords)
     (doom-color 'constants)
     (doom-color 'functions)
     (face-attribute 'default :foreground)
     (face-attribute 'shadow :foreground)
     (doom-blend 'base8 'error 0.1)
     (doom-blend 'base8 'success 0.1)
     (doom-blend 'base8 'type 0.1)
     (doom-blend 'base8 'keywords 0.1)
     (doom-blend 'base8 'constants 0.1)
     (doom-blend 'base8 'functions 0.1)
     (face-attribute 'default :foreground))))
```

2.4 Marginalia

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

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

2.5 File Templates

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

2.6 Editor config

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

3 Org-Mode

3.1 Defaults

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

3.2 Babel

```
(setq org-babel-default-header-args
      '((:session . "none")
        (:results . "replace")
        (:exports . "code")
        (:cache . "no")
        (:noweb . "no")
        (:hlines . "no")
        (:tangle . "no")
        (:comments . "link")))
```

3.3 Visuals

3.3.1 Org-modern

```
(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)
          ("NO"   :inverse-video t :inherit +org-todo-cancel))
        org-modern-footnote
        (cons nil (cadr org-script-display))
        org-modern-block-fringe nil
        org-modern-block-name
        '((t . t)
          ("src" "»" "«")
          ("example" "»–" "–«")
          ("quote" "" "")
          ("export" "" ""))
        org-modern-progress nil
        org-modern-priority nil
        org-modern-horizontal-rule (make-string 36 ?)
        org-modern-keyword
        '((t . t)
          ("title" . "")
          ("subtitle" . "")
          ("author" . "")
          ("email" . #("" 0 1 (display (raise -0.14))))
          ("date" . "")
          ("property" . "")
          ("options" . "")
          ("startup" . "")
          ("macro" . "")
          ("bind" . #("" 0 1 (display (raise -0.1))))
          ("bibliography" . "")
          ("print_bibliography" . #("" 0 1 (display (raise -0.1))))
          ("cite_export" . "")
          ("print_glossary" . #("" 0 1 (display (raise -0.1))))
          ("glossary_sources" . #("" 0 1 (display (raise -0.14))))
          ("include" . "")
          ("setupfile" . "")
          ("html_head" . "")
          ("html" . "")
          ("latex_class" . "")
          ("latex_class_options" . #("" 1 2 (display (raise -0.14))))
          ("latex_header" . "")
          ("latex_header_extra" . "")
          ("latex" . "")
          ("beamer_theme" . "")
          ("beamer_color_theme" . #("" 1 2 (display (raise -0.12))))
          ("beamer_font_theme" . "")
          ("beamer_header" . "")
          ("beamer" . "")
          ("attr_latex" . "")
          ("attr_html" . "")
          ("attr_org" . "")
          ("call" . #("" 0 1 (display (raise -0.15))))
          ("name" . "")
          ("header" . "›")
          ("caption" . "")
          ("results" . "")))
  (custom-set-faces! '(org-modern-statistics :inherit org-checkbox-statistics-todo)))
```

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

3.3.2 General

```
(add-hook 'org-mode-hook #'+org-pretty-mode)
```

```
(setq org-src-fontify-natively t
      org-fontify-whole-heading-line t
      org-fontify-done-headline t
      org-fontify-quote-and-verse-blocks t
      org-startup-with-inline-images t
      org-startup-indented t

      ;; Org styling, hide markup etc.
      org-pretty-entities t
      )

(setq org-ellipsis "  "
      org-hide-leading-stars t
      org-priority-highest ?A
      org-priority-lowest ?E
      org-priority-faces
      '((?A . 'all-the-icons-red)
        (?B . 'all-the-icons-orange)
        (?C . 'all-the-icons-yellow)
        (?D . 'all-the-icons-green)
        (?E . 'all-the-icons-blue)))


(setq org-inline-src-prettify-results '("⟨" . "⟩"))

(setq doom-themes-org-fontify-special-tags nil)
```

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

3.3.3 Org-appear

```
(package! org-appear)
```

```

(use-package! org-appear
  :after org
  :hook (org-mode . org-appear-mode)
  :config
  (setq org-appear-autoemphasis t
        org-appear-autosubmarkers t
        org-appear-autolinks t)
  ;; for proper first-time setup, `org-appear--set-elements'
  ;; needs to be run after other hooks have acted.
  (run-at-time nil nil #'org-appear--set-elements))
```

3.3.4 Ligatures

```
(appendq! +ligatures-extra-symbols
          (list :list_property ""
                :em_dash       "–"
                :ellipses      "..."
                :arrow_right   "→"
                :arrow_left    "←"
                :arrow_lr      ""
                :properties    ""
                :end           ""
                :priority_a    #("" 0 1 (face all-the-icons-red))
                :priority_b    #("" 0 1 (face all-the-icons-orange))
                :priority_c    #("" 0 1 (face all-the-icons-yellow))
                :priority_d    #("" 0 1 (face all-the-icons-green))
                :priority_e    #("" 0 1 (face all-the-icons-blue))))

(defadvice! +org-init-appearance-h--no-ligatures-a ()
  :after #'+org-init-appearance-h
  (set-ligatures! 'org-mode nil)
  (set-ligatures! 'org-mode
    :list_property "::"
    :em_dash       "---"
    :ellipsis      "..."
    :arrow_right   "->"
    :arrow_left    "<-"
    :arrow_lr      "<->"
    :properties    ":PROPERTIES:"
    :end           ":END:"
    :priority_a    "[#A]"
    :priority_b    "[#B]"
    :priority_c    "[#C]"
    :priority_d    "[#D]"
    :priority_e    "[#E]"))
```

3.3.5 Latex improvement

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

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

```
;; (package! org-fragtog)
```

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

3.4 Bullets

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

3.5 Agenda

3.5.1 Visual

```
(after! org-agenda
  (setq org-agenda-deadline-faces
      '((1.001 . error)
        (1.0 . org-warning)
        (0.5 . org-upcoming-deadline)
        (0.0 . org-upcoming-distant-deadline))))
```

3.6 Super-Agenda

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

3.6.1 Config

```
(setq  org-agenda-tags-column 0
       org-agenda-block-separator ?
       org-agenda-time-grid
       '((daily today require-timed)
         (800 1000 1200 1400 1600 1800 2000)
         "  " "")
       org-agenda-current-time-string
       " now ")
```

3.6.2 Customize

3.7 Roam

3.7.1 Defaults

```

(use-package! org-roam
  :after org
  :config
  (setq                   org-enable-roam-support t
                          org-roam-directory (concat org-directory "/Roam")
                          org-roam-v2-ack t))

```

3.7.2 Daily

```

(setq org-roam-dailies-directory "daily/")

(setq org-roam-dailies-capture-templates
      '(("d" "default" entry
         "* %?"
```

```
          :target (file+head "%<%Y-%m-%d>.org"
                             "#+title: %<%Y-%m-%d>\n")))
```

3.7.3 Visuals

1. UI and visualization

```
(package! org-roam-ui)
(package! websocket)
```

```

(defadvice! doom-modeline--buffer-file-name-roam-aware-a (orig-fun)
  :around #'doom-modeline-buffer-file-name ; takes no args
  (if (s-contains-p org-roam-directory (or buffer-file-name ""))
      (replace-regexp-in-string
       "\\(?:^\\|.*/\\)\\([0-9]\\{4\\}\\)\\([0-9]\\{2\\}\\)\\([0-9]\\{2\\}\\)[0-9]*-"
       "(\\1-\\2-\\3) "
       (subst-char-in-string ?_ ?  buffer-file-name))
    (funcall orig-fun)))
(use-package! websocket
  :after org-roam)
(use-package! org-roam-ui
  :after org-roam
  :commands org-roam-ui-open
  :hook (org-roam . org-roam-ui-mode)
  :config
  (setq org-roam-ui-sync-theme t
        org-roam-ui-follow t
        org-roam-ui-update-on-save t
        org-roam-ui-open-on-start t)
  (require 'org-roam) ; in case autoloaded
  (defun org-roam-ui-open ()
    "Ensure the server is active, then open the roam graph."
    (interactive)
    (unless org-roam-ui-mode (org-roam-ui-mode 1))
    (browse-url--browser (format "http://localhost:%d" org-roam-ui-port))))
```

3.8 Ob-async

3.8.1 Julia support

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

3.8.2 Jupyter Integration

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

3.9 Org-Diff

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

```
(use-package! orgdiff
  :defer t
  :config
  (defun +orgdiff-nicer-change-colours ()
    (goto-char (point-min))
    ;; Set red/blue based on whether chameleon is being used
    (if (search-forward "%% make document follow Emacs theme" nil t)
        (setq red  (substring (doom-blend 'red 'fg 0.8) 1)
              blue (substring (doom-blend 'blue 'teal 0.6) 1))
      (setq red  "c82829"
            blue "00618a"))
    (when (and (search-forward "%DIF PREAMBLE EXTENSION ADDED BY LATEXDIFF" nil t)
               (search-forward "\\RequirePackage{color}" nil t))
      (when (re-search-forward "definecolor{red}{rgb}{1,0,0}" (cdr (bounds-of-thing-at-point 'line))
      ↪  t)
        (replace-match (format "definecolor{red}{HTML}{%s}" red)))
      (when (re-search-forward "definecolor{blue}{rgb}{0,0,1}" (cdr (bounds-of-thing-at-point 'line))
      ↪  t)
        (replace-match (format "definecolor{blue}{HTML}{%s}"))))))
```

3.10 Pandoc import

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

```
(use-package! org-pandoc-import
  :after org)
```

3.11 Export

3.11.1 Preview

```
(map! :map org-mode-map

      :localleader
      :desc "View exported file" "v" #'org-view-output-file)

(defun org-view-output-file (&optional org-file-path)
  "Visit buffer open on the first output file (if any) found, using
  ↪  `org-view-output-file-extensions'"
  (interactive)
  (let* ((org-file-path (or org-file-path (buffer-file-name) ""))
         (dir (file-name-directory org-file-path))
         (basename (file-name-base org-file-path))
         (output-file nil))
    (dolist (ext org-view-output-file-extensions)
      (unless output-file
        (when (file-exists-p
               (concat dir basename "." ext))
          (setq output-file (concat dir basename "." ext)))))
    (if output-file
        (if (member (file-name-extension output-file) org-view-external-file-extensions)
            (browse-url-xdg-open output-file)
          (pop-to-buffer (or (find-buffer-visiting output-file)
                             (find-file-noselect output-file))))
      (message "No exported file found"))))

(defvar org-view-output-file-extensions '("pdf" "md" "rst" "txt" "tex" "html")
  "Search for output files with these extensions, in order, viewing the first that matches")
(defvar org-view-external-file-extensions '("html")
  "File formats that should be opened externally.")
```

3.12 Zotero Integration

```
(package! zotxt)
```

```
(use-package! zotxt
  :after org)
```

3.13 Org-Chef

```
(package! org-chef)
```

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

3.14 Bibtex-Integration

3.14.1 Citar

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

```
(use-package! citar
  :no-require
  :custom
  (org-cite-global-bibliography '("~/org/Lecture_Notes/MyLibrary.bib"))
  (citar-bibliography org-cite-global-bibliography)
  (citar-symbols
   `((file ,(all-the-icons-faicon "file-o" :face 'all-the-icons-green :v-adjust -0.1) . " ")
     (note ,(all-the-icons-material "speaker_notes" :face 'all-the-icons-blue :v-adjust -0.3) . " ")
     (link ,(all-the-icons-octicon "link" :face 'all-the-icons-orange :v-adjust 0.01) . " ")))
  (citar-symbol-separator "  "))
```

```
(use-package! oc-csl
  :after oc
  :config
  (setq org-cite-csl-styles-dir "~/Zotero/styles/"))
(after! oc
 (setq org-cite-export-processors '((t csl))))
```

```
(use-package! oc-csl-activate
  :after org
  :config
  (setq org-cite-activate-processor 'csl-activate)
  (setq org-cite-csl-activate-use-document-style t)
  (setq org-cite-csl-activate-use-document-locale t)
  (add-hook 'org-mode-hook
            (lambda ()
              (cursor-sensor-mode 1)
              (org-cite-csl-activate-render-all))))
```

3.14.2 Org-Roam integration

```
(use-package! citar-org-roam
  :after citar org-roam
  :config (citar-org-roam-mode))
(setq org-roam-capture-templates
      '(("d" "default" plain
         "%?"
         :target
         (file+head
          "%<%Y%m%d%H%M%S>-${slug}.org"
          "#+title: ${title}\n")
         :unnarrowed t)
        ("n" "literature note" plain
         "%?"
         :target
         (file+head
          "%(expand-file-name \"literature\" org-roam-directory)/${citekey}.org"
          "#+title: ${citekey}. ${title}.\n#+created: %U\n#+last_modified: %U\n\n")
         :unnarrowed t)))
(setq citar-org-roam-capture-template-key "n")
```

3.15 Latex templates

3.15.1 Preview

1. PNG

```
(after! org
  ;; ORG LATEX PREVIEW
  (setq org-format-latex-options
        (plist-put org-format-latex-options :background "Transparent"))
  (setq org-format-latex-options
        (plist-put org-format-latex-options :scale 1))
  (setq org-preview-latex-default-process 'dvisvgm)
  (setq org-preview-latex-image-directory "~/.cache/ltximg/")
  )
```

2. Header

```
(setq org-format-latex-header "\\documentclass[12pt]
{article}
\\usepackage[usenames]{xcolor}
\\usepackage{booktabs}
\\pagestyle{empty}             % do not remove
% The settings below are copied from fullpage.sty
\\setlength{\\textwidth}{\\paperwidth}
\\addtolength{\\textwidth}{-3cm}
\\setlength{\\oddsidemargin}{1.5cm}
\\addtolength{\\oddsidemargin}{-2.54cm}
\\setlength{\\evensidemargin}{\\oddsidemargin}
\\setlength{\\textheight}{\\paperheight}
\\addtolength{\\textheight}{-\\headheight}
\\addtolength{\\textheight}{-\\headsep}
\\addtolength{\\textheight}{-\\footskip}
\\addtolength{\\textheight}{-3cm}
\\setlength{\\topmargin}{1.5cm}
\\addtolength{\\topmargin}{-2.54cm}
% my custom stuff
\\usepackage{xfrac}
\\usepackage{siunitx}
\\usepackage{diffcoeff}
\\usepackage{nicematrix}
\\DeclareMathOperator{\\Var}{Var}
\\DeclareMathOperator{\\cov}{Cov}
\\DeclareMathOperator{\\E}{\\mathbb{E}}
\\DeclareMathOperator*{\\argmax}{arg\\,max}
\\DeclareMathOperator*{\\argmin}{arg\\,min}
")
```

3.15.2 Article

```
(with-eval-after-load 'ox-latex
(add-to-list 'org-latex-classes
             '("article"
               "\\documentclass[c]{article}
\\usepackage[american]{babel}
\\usepackage[margin=1.25in]{geometry}
\\usepackage{parskip}
\\usepackage{booktabs}
\\usepackage{float}
\\usepackage{microtype}
\\usepackage{graphicx}
\\usepackage{mathtools}
\\usepackage{wrapfig}
\\usepackage{amsthm}
\\usepackage{amssymb}
\\usepackage{newpxtext}
\\usepackage[varbb]{newpxmath}
\\usepackage{xfrac}
\\usepackage{siunitx}
\\usepackage{caption}
\\captionsetup{labelfont=bf,font={small,singlespacing}}
\\usepackage{subcaption}
\\usepackage{cancel}
\\usepackage{setspace}
\\usepackage{xcolor}
\\usepackage{diffcoeff}
\\usepackage{nicematrix}
\\usepackage{enumitem}
\\usepackage{acronym}
\\usepackage{xurl}
\\onehalfspacing{}
\\DeclareMathOperator{\\Var}{Var}
\\DeclareMathOperator{\\cov}{Cov}
\\DeclareMathOperator{\\E}{\\mathbb{E}}
\\DeclareMathOperator*{\\argmax}{arg\\,max}
\\DeclareMathOperator*{\\argmin}{arg\\,min}
\\newcommand{\\Et}[2]{\\E_{#2} \\left[#1\\right]}
\\newcommand{\\Covt}[3]{\\cov_{#3}\\left(#1, #2\\right)}
\\newcommand{\\Vart}[2]{\\Var_{#2} \\left[#1\\right]}
\\DeclarePairedDelimiter\\abs{\\lvert}{\\rvert}
\\DeclarePairedDelimiter\\norm{\\lVert}{\\rVert}
\\DeclarePairedDelimiterX\\innerp[2]{\\langle}{\\rangle}{#1,#2}
\\DeclarePairedDelimiterX\\braket[3]{\\langle}{\\rangle}%
{#1\\,\\delimsize\\vert\\,\\mathopen{}#2\\,\\delimsize\\vert\\,\\mathopen{}#3}
\\providecommand\\given{}
\\DeclarePairedDelimiterXPP\\Prob[1]{\\mathbb{P}} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1}
\\DeclarePairedDelimiterXPP\\condE[1]{\\E} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1}
\\DeclarePairedDelimiterXPP\\condVar[2]{\\Var} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1,#2}
\\DeclarePairedDelimiterXPP\\condCov[2]{\\cov} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1,#2}
\\theoremstyle{plain}% default
\\newtheorem{thm}{Theorem}
\\newtheorem{lem}[thm]{Lemma}
\\newtheorem{prop}[thm]{Proposition}
\\newtheorem*{cor}{Corollary}
\\theoremstyle{definition}
\\newtheorem{defn}{Definition}
\\newtheorem{exmp}{Example}
\\providecommand*{\\defnautorefname}{Definition}
\\theoremstyle{remark}
\\newtheorem*{rem}{Remark}
\\newtheorem*{note}{Note}
\\newtheorem{case}{Case}

\\renewcommand{\\leq}{\\leqslant}
\\renewcommand{\\geq}{\\geqslant}
\\definecolor{bgcolorminted}{HTML}{f9f5d7}
\\usepackage{hyperref}
\\usepackage[]{cleveref}

[NO-DEFAULT-PACKAGES]
[PACKAGES]
[EXTRA]"
               ("\\section{%s}" . "\\section*{%s}")
               ("\\subsection{%s}" . "\\subsection*{%s}")
               ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
               ("\\paragraph{%s}" . "\\paragraph*{%s}"))))
```

3.15.3 Beamer

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

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

Define Beamer class:

```

(with-eval-after-load 'ox-latex
(add-to-list 'org-latex-classes
             '("beamer"
               "\\documentclass[c]{beamer}
\\usepackage[american]{babel}
\\usetheme[progressbar=frametitle, titleformat=smallcaps, numbering=fraction]{metropolis}
\\usepackage{booktabs}
\\usepackage{float}
\\usepackage{mathtools}
\\usepackage{amsthm}
\\usepackage{amssymb}
\\usepackage[varbb]{newpxmath}
\\usepackage[]{xfrac}
\\usepackage{siunitx}
\\usepackage{graphicx}
\\usepackage{caption}
\\captionsetup{labelfont=bf,font={small,singlespacing}}
\\usepackage{subcaption}
\\usepackage{cancel}
\\usepackage{setspace}
\\usepackage{xcolor}
\\usepackage{diffcoeff}
\\usepackage{nicematrix}
\\usepackage{acronym}
\\usepackage{appendixnumberbeamer}
\\usepackage{dirtytalk}
\\usepackage{xurl}
\\DeclareMathOperator{\\Var}{Var}
\\DeclareMathOperator{\\cov}{Cov}
\\DeclareMathOperator{\\E}{\\mathbb{E}}
\\DeclareMathOperator*{\\argmax}{arg\\,max}
\\DeclareMathOperator*{\\argmin}{arg\\,min}
\\newcommand{\\Et}[2]{\\E_{#2} \\left[#1\\right]}
\\newcommand{\\Covt}[3]{\\cov_{#3}\\left(#1, #2\\right)}
\\newcommand{\\Vart}[2]{\\Var_{#2} \\left[#1\\right]}
\\DeclarePairedDelimiter\\abs{\\lvert}{\\rvert}
\\DeclarePairedDelimiter\\norm{\\lVert}{\\rVert}
\\DeclarePairedDelimiterX\\innerp[2]{\\langle}{\\rangle}{#1,#2}
\\DeclarePairedDelimiterX\\braket[3]{\\langle}{\\rangle}%
{#1\\,\\delimsize\\vert\\,\\mathopen{}#2\\,\\delimsize\\vert\\,\\mathopen{}#3}
\\providecommand\\given{}
```

```
\\DeclarePairedDelimiterXPP\\Prob[1]{\\mathbb{P}} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1}
\\DeclarePairedDelimiterXPP\\condE[1]{\\E} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1}
\\DeclarePairedDelimiterXPP\\condVar[2]{\\Var} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1,#2}
\\DeclarePairedDelimiterXPP\\condCov[2]{\\cov} (){}{
\\renewcommand\\given{\\nonscript\\:\\delimsize\\vert\\nonscript\\:\\mathopen{}}
#1,#2}
\\theoremstyle{plain}% default
\\newtheorem{thm}{Theorem}
\\newtheorem{lem}[thm]{Lemma}
\\newtheorem{prop}[thm]{Proposition}
\\newtheorem*{cor}{Corollary}
\\theoremstyle{definition}
\\newtheorem{defn}{Definition}
\\newtheorem{exmp}{Example}
\\providecommand*{\\defnautorefname}{Definition}
\\theoremstyle{remark}
\\newtheorem*{rem}{Remark}
\\newtheorem{case}{Case}


\\definecolor{dblue}{HTML}{2E3440}
\\definecolor{umber}{HTML}{8FBCBB}
\\definecolor{alertcolor}{HTML}{BF616A}
\\definecolor{examplecolor}{HTML}{EBCB8B}

\\definecolor{pale}{HTML}{ECEFF4}
\\definecolor{bluish}{HTML}{88C0D0}
\\definecolor{cream}{HTML}{D8DEE9}
\\definecolor{bgcolorminted}{HTML}{f9f5d7}
\\setbeamercolor{progress bar}{fg=bluish,bg=cream}
\\setbeamercolor{frametitle}{fg=umber,bg=pale}
\\setbeamercolor{normal text}{fg=dblue,bg=pale}
\\setbeamercolor{alerted text}{fg=alertcolor,bg=pale}
\\setbeamercolor{example text}{fg=examplecolor}
\\setbeamercovered{dynamic}

\\usecolortheme{rose}
\\usepackage{hyperref}
\\usepackage{cleveref}
[NO-DEFAULT-PACKAGES]
[PACKAGES]
[EXTRA]"
               ("\\section{%s}" . "\\section*{%s}")
               ("\\subsection{%s}" . "\\subsection*{%s}")
               ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
```

```
                ("\\paragraph{%s}" . "\\paragraph*{%s}")
                ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
```

3.15.4 Export

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

```
(setq org-latex-tables-booktabs t
      org-latex-hyperref-template "\\providecolor{url}{HTML}{81a1c1}
\\providecolor{link}{HTML}{d08770}
\\providecolor{cite}{HTML}{d08770}
\\hypersetup{
pdfauthor={%a},
pdftitle={%t},
pdfkeywords={%k},
pdfsubject={%d},
pdfcreator={%c},
pdflang={%L},
breaklinks=true,
colorlinks=true,
linkcolor=link,
urlcolor=url,
citecolor=cite
}
\\urlstyle{same}
%% hide links styles in toc
\\NewCommandCopy{\\oldtoc}{\\tableofcontents}
\\renewcommand{\\tableofcontents}{\\begingroup\\hypersetup{hidelinks}\\oldtoc\\endgroup}
"
      org-latex-reference-command "\\cref{%s}")
```

1. Preview

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

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

2. Code blocks

```
(package! engrave-faces)
```

```
;; (setq org-latex-listings 'minted
;;       org-latex-packages-alist '(("" "minted")))
;; (setq org-latex-minted-options '(("breaklines" "true")
;;                                  ("breakanywhere" "true")
;;                                  ("bgcolor" "bgcolorminted")
;;                                  ("linenos" "true")))
(use-package! engrave-faces-latex
  :after ox-latex)
(setq org-latex-src-block-backend 'engraved)
(setq org-latex-engraved-theme 't)
```

3.16 Capture

3.16.1 Doct

```
(package! doct)
```

Prettify the captures:

```
(after! org-capture

  (defun +doct-icon-declaration-to-icon (declaration)
    "Convert :icon declaration to icon"
    (let ((name (pop declaration))
          (set  (intern (concat "all-the-icons-" (plist-get declaration :set))))
          (face (intern (concat "all-the-icons-" (plist-get declaration :color))))
          (v-adjust (or (plist-get declaration :v-adjust) 0.01)))
      (apply set `(,name :face ,face :v-adjust ,v-adjust))))

  (defun +doct-iconify-capture-templates (groups)
    "Add declaration's :icon to each template group in GROUPS."
    (let ((templates (doct-flatten-lists-in groups)))
      (setq doct-templates (mapcar (lambda (template)
                                     (when-let* ((props (nthcdr (if (= (length template) 4) 2 5)
                                     ↪   template))
                                                 (spec (plist-get (plist-get props :doct) :icon)))
                                       (setf (nth 1 template) (concat (+doct-icon-declaration-to-icon
                                       ↪   spec)
                                                                      "\t"
                                                                      (nth 1 template))))
                                     template)
                                   templates))))

  (setq doct-after-conversion-functions '(+doct-iconify-capture-templates))

  (defvar +org-capture-recipies  "~/org/recipies.org")

  (defun set-org-capture-templates ()
    (setq org-capture-templates
          (doct `(("Personal todo" :keys "t"
                   :icon ("checklist" :set "octicon" :color "green")
                   :file +org-capture-todo-file
                   :prepend t
                   :headline "Inbox"
```

```
 :type entry
 :template ("* TODO %?"
            "%i %a"))
("Personal note" :keys "n"
 :icon ("sticky-note-o" :set "faicon" :color "green")
 :file +org-capture-todo-file
 :prepend t
 :headline "Inbox"
 :type entry
 :template ("* %?"
            "%i %a"))
("Email" :keys "e"
 :icon ("envelope" :set "faicon" :color "blue")
 :file +org-capture-todo-file
 :prepend t
 :headline "Inbox"
 :type entry
 :template ("* TODO %^{type|reply to|contact} %\\3 %? :email:"
            "Send an email %^{urgancy|soon|ASAP|anon|at some point|eventually} to
            ↪   %^{recipiant}"
            "about %^{topic}"
            "%U %i %a"))
("Meeting" :keys "m"
 :icon ("users" :set "faicon")
 :file "~/org/meeting.org"
 :preprend t
 :headline "Meetings"
 :type entry
 :template ("* %^{Topic}
              + Attendees:  %^{Attendees},Leo
              + Date: %U
              ** Notes
                 +  %?
              ** Actions
                 + [ ]     "))
("Interesting" :keys "i"
 :icon ("eye" :set "faicon" :color "lcyan")
 :file +org-capture-todo-file
 :prepend t
 :headline "Interesting"
 :type entry
 :template ("* [ ] %{desc}%? :%{i-type}:"
            "%i %a")
 :children (("Webpage" :keys "w"
             :icon ("globe" :set "faicon" :color "green")
             :desc "%(org-cliplink-capture) "
             :i-type "read:web")
            ("Article" :keys "a"
             :icon ("file-text" :set "octicon" :color "yellow")
             :desc ""
             :i-type "read:reaserch")
```

```
                ("\tRecipie" :keys "r"
                 :icon ("spoon" :set "faicon" :color "dorange")
                 :file +org-capture-recipies
                 :headline "Unsorted"
                 :template "%(org-chef-get-recipe-from-url)")
                ("Information" :keys "i"
                 :icon ("info-circle" :set "faicon" :color "blue")
                 :desc ""
                 :i-type "read:info")
                ("Idea" :keys "I"
                 :icon ("bubble_chart" :set "material" :color "silver")
                 :desc ""
                 :i-type "idea")))
          ("Tasks" :keys "k"
           :icon ("inbox" :set "octicon" :color "yellow")
           :file +org-capture-todo-file
           :prepend t
           :headline "Tasks"
           :type entry
           :template ("* TODO %? %^G%{extra}"
                      "%i %a")
           :children (("General Task" :keys "k"
                       :icon ("inbox" :set "octicon" :color "yellow")
                       :extra "")
                      ("Task with deadline" :keys "d"
                       :icon ("timer" :set "material" :color "orange" :v-adjust -0.1)
                       :extra "\nDEADLINE: %^{Deadline:}t")
                      ("Scheduled Task" :keys "s"
                       :icon ("calendar" :set "octicon" :color "orange")
                       :extra "\nSCHEDULED: %^{Start time:}t")))
          ("Project" :keys "p"
           :icon ("repo" :set "octicon" :color "silver")
           :prepend t
           :type entry
           :headline "Inbox"
           :template ("* %{time-or-todo} %?"
                      "%i"
                      "%a")
           :file ""
           :custom (:time-or-todo "")
           :children (("Project-local todo" :keys "t"
                       :icon ("checklist" :set "octicon" :color "green")
                       :time-or-todo "TODO"
                       :file +org-capture-project-todo-file)
                      ("Project-local note" :keys "n"
                       :icon ("sticky-note" :set "faicon" :color "yellow")
                       :time-or-todo "%U"
                       :file +org-capture-project-notes-file)
                      ("Project-local changelog" :keys "c"
                       :icon ("list" :set "faicon" :color "blue")
                       :time-or-todo "%U"
```

```
                               :heading "Unreleased"
                               :file +org-capture-project-changelog-file))
                  ("\tCentralised project templates"
                   :keys "o"
                   :type entry
                   :prepend t
                   :template ("* %{time-or-todo} %?"
                              "%i"
                              "%a")
                   :children (("Project todo"
                               :keys "t"
                               :prepend nil
                               :time-or-todo "TODO"
                               :heading "Tasks"
                               :file +org-capture-central-project-todo-file)
                              ("Project note"
                               :keys "n"
                               :time-or-todo "%U"
                               :heading "Notes"
                               :file +org-capture-central-project-notes-file)
                              ("Project changelog"
                               :keys "c"
                               :time-or-todo "%U"
                               :heading "Unreleased"
                               :file +org-capture-central-project-changelog-file)))))))

  (set-org-capture-templates)
  (unless (display-graphic-p)
    (add-hook! 'server-after-make-frame-hook
              (defun org-capture-reinitialise-hook ()
                (when (display-graphic-p)
                  (set-org-capture-templates)
                  (remove-hook 'server-after-make-frame-hook
                               #'org-capture-reinitialise-hook))))))
```

```
(defun org-mks-pretty (table title &optional prompt specials)
  "Select a member of an alist with multiple keys. Prettified.

TABLE is the alist which should contain entries where the car is a string.
There should be two types of entries.

1. prefix descriptions like (\"a\" \"Description\")
   This indicates that `a' is a prefix key for multi-letter selection, and
   that there are entries following with keys like \"ab\", \"ax\"...

2. Select-able members must have more than two elements, with the first
   being the string of keys that lead to selecting it, and the second a
   short description string of the item.

The command will then make a temporary buffer listing all entries
that can be selected with a single key, and all the single key
```

```
prefixes.  When you press the key for a single-letter entry, it is selected.
When you press a prefix key, the commands (and maybe further prefixes)
under this key will be shown and offered for selection.

TITLE will be placed over the selection in the temporary buffer,
PROMPT will be used when prompting for a key.  SPECIALS is an
alist with (\"key\" \"description\") entries.  When one of these
is selected, only the bare key is returned."
  (save-window-excursion
    (let ((inhibit-quit t)
          (buffer (org-switch-to-buffer-other-window "*Org Select*"))
          (prompt (or prompt "Select: "))
          case-fold-search
          current)
      (unwind-protect
          (catch 'exit
            (while t
              (setq-local evil-normal-state-cursor (list nil))
              (erase-buffer)
              (insert title "\n\n")
              (let ((des-keys nil)
                    (allowed-keys '("\C-g"))
                    (tab-alternatives '("\s" "\t" "\r"))
                    (cursor-type nil))
                ;; Populate allowed keys and descriptions keys
                ;; available with CURRENT selector.
                (let ((re (format "\\`%s\\(.\\)\\'"
                                  (if current (regexp-quote current) "")))
                      (prefix (if current (concat current " ") "")))
                  (dolist (entry table)
                    (pcase entry
                      ;; Description.
                      (`(,(and key (pred (string-match re))) ,desc)
                       (let ((k (match-string 1 key)))
                         (push k des-keys)
                         ;; Keys ending in tab, space or RET are equivalent.
                         (if (member k tab-alternatives)
                             (push "\t" allowed-keys)
                           (push k allowed-keys))
                         (insert (propertize prefix 'face 'font-lock-comment-face) (propertize k
                         ↪   'face 'bold) (propertize "›" 'face 'font-lock-comment-face) "  " desc
                         ↪   "..." "\n")))
                      ;; Usable entry.
                      (`(,(and key (pred (string-match re))) ,desc . ,_)
                       (let ((k (match-string 1 key)))
                         (insert (propertize prefix 'face 'font-lock-comment-face) (propertize k
                         ↪   'face 'bold) "   " desc "\n")
                         (push k allowed-keys)))
                      (_ nil))))
                ;; Insert special entries, if any.
                (when specials
                  (insert "\n")
                  (pcase-dolist (`(,key ,description) specials)
                    (insert (format "%s   %s\n" (propertize key 'face '(bold all-the-icons-red))
                    ↪  description))
                    (push key allowed-keys)))
                ;; Display UI and let user select an entry or
                ;; a sub-level prefix.
                (goto-char (point-min))
                (unless (pos-visible-in-window-p (point-max))
                  (org-fit-window-to-buffer))
                (let ((pressed (org--mks-read-key allowed-keys
                                                  prompt
                                                  (not (pos-visible-in-window-p (1- (point-max)))))))
                  (setq current (concat current pressed))
                  (cond
                   ((equal pressed "\C-g") (user-error "Abort"))
                   ;; Selection is a prefix: open a new menu.
                   ((member pressed des-keys))
                   ;; Selection matches an association: return it.
                   ((let ((entry (assoc current table)))
                      (and entry (throw 'exit entry))))
                   ;; Selection matches a special entry: return the
                   ;; selection prefix.
                   ((assoc current specials) (throw 'exit current))
                   (t (error "No entry available")))))))
        (when buffer (kill-buffer buffer))))))
(advice-add 'org-mks :override #'org-mks-pretty)
```

4 Company

Improve the history size:

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

```
(set-company-backend!
  '(text-mode
    markdown-mode
    gfm-mode)
  '(:seperate
    company-ispell
    company-files
    company-yasnippet))
```

5 LSP

5.1 Digestif

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

5.2 LTex

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

```
;; (use-package! lsp-ltex
;;   :hook (text-mode . (lambda ()
;;                        (require 'lsp-ltex)
;;                        (lsp)))  ; or lsp-deferred
;;   :init
;;   (setq lsp-ltex-version "15.2.0"))  ; make sure you have set this, see below
;; (use-package! eglot-ltex
;;   :hook (text-mode . (lambda ()
;;                        (require 'eglot-ltex)
;;                        (call-interactively #'eglot)))
;;   :init
;;   (setq eglot-languagetool-server-path "/opt/homebrew/Cellar/ltex-ls/15.2.0"))
```

6 VLFI

```
(package! vlfi)
```

```
(use-package! vlf-setup
  :defer-incrementally vlf-tune vlf-base vlf-write vlf-search vlf-occur vlf-follow vlf-ediff vlf)
```

7 PDF-Tools

7.1 Fix annotation bug

```
(defun my-fix-tablist ()
  (interactive)
  (unload-feature 'tablist-filter t)
  (load-file (find-library-name "tablist-filter")))
```

7.2 Dark mode

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

8 Option key Fix

```
(defun switch-left-and-right-option-keys ()
  "Switch left and right option keys.
     On some external keyboards the left and right option keys are swapped,
     this command switches the keys so that they work as expected."
  (interactive)
  (let ((current-left  mac-option-modifier)
        (current-right mac-right-option-modifier))
    (setq mac-option-modifier       current-right
          mac-right-option-modifier current-left)))
```

9 Centaur

```
;; (after! centaur-tabs
;;   (centaur-tabs-mode -1)
;;   (setq centaur-tabs-height 36
;;         centaur-tabs-set-icons t
;;         centaur-tabs-modified-marker "o"
;;         centaur-tabs-close-button "×"
;;         centaur-tabs-set-bar 'above
;;         centaur-tabs-gray-out-icons 'buffer)
;;   (centaur-tabs-change-fonts "P22 Underground Book" 160))
;; (setq x-underline-at-descent-line t)
```

10 Email

10.1 mu4e

```
;; add to $DOOMDIR/config.el
(after! mu4e
  (setq sendmail-program (executable-find "msmtp")
        send-mail-function #'smtpmail-send-it
        message-sendmail-f-is-evil t
        message-sendmail-extra-arguments '("--read-envelope-from")
        message-send-mail-function #'message-send-mail-with-sendmail)
  ;; how often to call it in seconds:
  (setq   mu4e-sent-messages-behavior 'sent ;; Save sent messages
          mu4e-headers-auto-update t                ; avoid to type `g' to update
          mu4e-compose-signature-auto-include nil   ; I don't want a message signature
          mu4e-use-fancy-chars t                   ; allow fancy icons for mail threads
          mu4e-context-policy 'pick-first   ;; Start with the first context
          mu4e-compose-context-policy 'ask) ;; Always ask which context to use when composing a new
          ↪  mail
  (setq mu4e-update-interval (* 1 60))
  (setq mu4e-attachment-dir "~/Downloads")
  (set-email-account! "gmail"
                      '((mu4e-sent-folder       . "/gmail/[Gmail]/Sent Mail")
                        (mu4e-drafts-folder     . "/gmail/[Gmail]/Drafts")
                        (mu4e-trash-folder      . "/gmail/[Gmail]/Trash")
                        (mu4e-refile-folder     . "/gmail/[Gmail]/All Mail")
                        (smtpmail-smtp-user     . "leoaparisi@gmail.com")
                        (mu4e-compose-signature . "---\nLeo Aparisi de Lannoy"))
                      t)
  (set-email-account! "U Chicago"
                      '((mu4e-sent-folder       . "/UChicago/Sent Mail")
                        (mu4e-drafts-folder     . "/UChicago/Drafts")
                        (mu4e-trash-folder      . "/UChicago/Trash")
                        (mu4e-refile-folder     . "/UChicago/All Mail")
                        (smtpmail-smtp-user     . "laparisidelannoy@uchicago.edu")
                        (mu4e-compose-signature . "---\nLeo Aparisi de Lannoy"))
                      t)
  (setq +mu4e-gmail-accounts '(("leoaparisi@gmail.com" . "/gmail/[Gmail]")))
  (setq mu4e-compose-dont-reply-to-self t)
  ;; Add a unified inbox shortcut
  (add-to-list
   'mu4e-bookmarks
   '(:name "Unified inbox" :query "maildir:/.*inbox/" :key ?i) t)
(add-hook 'mu4e-compose-mode-hook 'company-mode)
  )
```

10.2 Notification

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

11 RSS

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

11.1 Visual

```
(after! elfeed
  (setq elfeed-search-filter "@1-week-ago +unread"
        elfeed-search-print-entry-function '+rss/elfeed-search-print-entry
        elfeed-search-title-min-width 80
        elfeed-show-entry-switch #'switch-to-buffer
        elfeed-show-entry-delete #'+rss/delete-pane
        elfeed-show-refresh-function #'+rss/elfeed-show-refresh--better-style
        shr-max-image-proportion 0.6)

  (add-hook! 'elfeed-show-mode-hook (hide-mode-line-mode 1))
  (add-hook! 'elfeed-search-update-hook #'hide-mode-line-mode)
  (defface elfeed-show-title-face '((t (:weight ultrabold :slant italic :height 1.5)))
    "title face in elfeed show buffer"
    :group 'elfeed)
  (defface elfeed-show-author-face `((t (:weight light)))
    "title face in elfeed show buffer"
    :group 'elfeed)
  (set-face-attribute 'elfeed-search-title-face nil
                      :foreground 'nil
                      :weight 'light)

  (defadvice! +rss-elfeed-wrap-h-nicer ()
    "Enhances an elfeed entry's readability by wrapping it to a width of
`fill-column' and centering it with `visual-fill-column-mode'."
    :override #'+rss-elfeed-wrap-h
    (setq-local truncate-lines nil
                shr-width 140
                visual-fill-column-center-text t
                default-text-properties '(line-height 1.2))
    (let ((inhibit-read-only t)
          (inhibit-modification-hooks t))
       (setq-local shr-current-font '(:family "Lato" :height 1.2))
      (set-buffer-modified-p nil)))

  (defun +rss/elfeed-search-print-entry (entry)
    "Print ENTRY to the buffer."
    (let* ((elfeed-goodies/tag-column-width 40)
           (elfeed-goodies/feed-source-column-width 30)
           (title (or (elfeed-meta entry :title) (elfeed-entry-title entry) ""))
           (title-faces (elfeed-search--faces (elfeed-entry-tags entry)))
           (feed (elfeed-entry-feed entry))
           (feed-title
            (when feed
              (or (elfeed-meta feed :title) (elfeed-feed-title feed))))
           (tags (mapcar #'symbol-name (elfeed-entry-tags entry)))
           (tags-str (concat (mapconcat 'identity tags ",")))
           (title-width (- (window-width) elfeed-goodies/feed-source-column-width
                           elfeed-goodies/tag-column-width 4))

           (tag-column (elfeed-format-column
                        tags-str (elfeed-clamp (length tags-str)
                                               elfeed-goodies/tag-column-width
                                               elfeed-goodies/tag-column-width)
                        :left))
           (feed-column (elfeed-format-column
                         feed-title (elfeed-clamp elfeed-goodies/feed-source-column-width
                                                  elfeed-goodies/feed-source-column-width
                                                  elfeed-goodies/feed-source-column-width)
                         :left)))

      (insert (propertize feed-column 'face 'elfeed-search-feed-face) " ")
      (insert (propertize tag-column 'face 'elfeed-search-tag-face) " ")
      (insert (propertize title 'face title-faces 'kbd-help title))
      (setq-local line-spacing 0.2)))

  (defun +rss/elfeed-show-refresh--better-style ()
    "Update the buffer to match the selected entry, using a mail-style."
    (interactive)
    (let* ((inhibit-read-only t)
           (title (elfeed-entry-title elfeed-show-entry))
           (date (seconds-to-time (elfeed-entry-date elfeed-show-entry)))
           (author (elfeed-meta elfeed-show-entry :author))
           (link (elfeed-entry-link elfeed-show-entry))
           (tags (elfeed-entry-tags elfeed-show-entry))
           (tagsstr (mapconcat #'symbol-name tags ", "))
           (nicedate (format-time-string "%a, %e %b %Y %T %Z" date))
           (content (elfeed-deref (elfeed-entry-content elfeed-show-entry)))
           (type (elfeed-entry-content-type elfeed-show-entry))
           (feed (elfeed-entry-feed elfeed-show-entry))
           (feed-title (elfeed-feed-title feed))
           (base (and feed (elfeed-compute-base (elfeed-feed-url feed)))))
      (erase-buffer)
      (insert "\n")
      (insert (format "%s\n\n" (propertize title 'face 'elfeed-show-title-face)))
      (insert (format "%s\t" (propertize feed-title 'face 'elfeed-search-feed-face)))
      (when (and author elfeed-show-entry-author)
        (insert (format "%s\n" (propertize author 'face 'elfeed-show-author-face))))
      (insert (format "%s\n\n" (propertize nicedate 'face 'elfeed-log-date-face)))
      (when tags
        (insert (format "%s\n"
                        (propertize tagsstr 'face 'elfeed-search-tag-face))))
      ;; (insert (propertize "Link: " 'face 'message-header-name))
      ;; (elfeed-insert-link link link)
      ;; (insert "\n")
      (cl-loop for enclosure in (elfeed-entry-enclosures elfeed-show-entry)
               do (insert (propertize "Enclosure: " 'face 'message-header-name))
               do (elfeed-insert-link (car enclosure))
               do (insert "\n"))
      (insert "\n")
      (if content
          (if (eq type 'html)
              (elfeed-insert-html content base)
            (insert content))
        (insert (propertize "(empty)\n" 'face 'italic)))
      (goto-char (point-min))))

  )
```