Config

Leo Aparisi de Lannoy

January 28, 2023

Contents

1 Basic

1.1 ID

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

1.2 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
```

1.2.1 Browser

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

1.2.2 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"))))
```

1.3 Visual

1.3.1 Font

```
(setq doom-font (font-spec :family "Iosevka" :size 13)
      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))
```

1.3.2 Theme

1. Catppuccin theme

```
(load-theme 'catppuccin t t)
(setq doom-theme 'catppuccin)
(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))
```

1.3.3 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 '(95 95))
(add-to-list 'default-frame-alist '(alpha 95 95))
(add-to-list 'default-frame-alist '(fullscreen . maximized))
```

1.3.4 Starting image

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

1.3.5 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))))

```

1.4 Marginalia

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

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

1.5 File Templates

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

1.6 Editor config

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

2 Org-Mode

2.1 Defaults

```
(setq org-directory "~/org/"
      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-export-in-background t                  ; run export processes in
      ↪ external emacs process
      org-fold-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 _{} / ^{}
```

2.2 Visuals

2.2.1 General

```
(setq org-ascii-charset 'utf-8)
  (setq org-src-fontify-natively t
        org-fontify-whole-heading-line t
```

```
        org-pretty-entities \nil
        org-ellipsis "  " ;; folding symbol
        org-fontify-done-headline t
        org-fontify-quote-and-verse-blocks t
        org-startup-with-inline-images t
        org-startup-indented t)

(lambda () (progn
             (setq left-margin-width 2)
             (setq right-margin-width 2)
             (set-window-buffer nil (current-buffer))))
(add-hook! 'org-mode-hook #'+org-pretty-mode)
(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))
```

2.2.2 Bullets

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

2.2.3 Org-appear

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

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

2.2.4 Ligatures

```
(appendq! +ligatures-extra-symbols
          `(:checkbox      ""
            :pending       ""
            :checkedbox    ""
            :list_property ""
            :em_dash       "-"
            :ellipses      "..."
            :arrow_right   "→"
            :arrow_left    "←"
            :title         ""
            :subtitle      ""
            :author        ""
            :date          ""
            :property      ""
            :options       ""
            :startup       ""
            :macro         ""
            :html_head     ""
            :html          ""
            :latex_class   ""
            :latex_header  ""
            :beamer_header ""
            :latex         ""
            :attr_latex    ""
            :attr_html     ""
            :attr_org      ""
```

```
            :begin_quote   ""
            :end_quote     ""
            :caption       ""
            :header        ">"
            :results       ""
            :begin_export  ""
            :end_export    ""
            :properties    ""
            :end           ""
            :priority_a   ,(propertize "" 'face 'all-the-icons-red)
            :priority_b   ,(propertize "" 'face 'all-the-icons-orange)
            :priority_c   ,(propertize "" 'face 'all-the-icons-yellow)
            :priority_d   ,(propertize "" 'face 'all-the-icons-green)
            :priority_e   ,(propertize "" 'face 'all-the-icons-blue)))
```

```
(set-ligatures! 'org-mode
  :merge t
  :checkbox      "[ ]"
  :pending       "[-]"
  :checkedbox    "[X]"
  :list_property "::"
  :em_dash       "---"
  :ellipsis      "..."
  :arrow_right   "->"
  :arrow_left    "<-"
  :title         "#+title:"
  :subtitle      "#+subtitle:"
  :author        "#+author:"
  :date          "#+date:"
  :property      "#+property:"
  :options       "#+options:"
  :startup      "#+startup:"
  :macro         "#+macro:"
  :html_head     "#+html_head:"
  :html          "#+html:"
  :latex_class   "#+latex_class:"
  :latex_header  "#+latex_header:"
  :beamer_header "#+beamer_header:"
```

```
  :latex          "#+latex:"
  :attr_latex     "#+attr_latex:"
  :attr_html      "#+attr_html:"
  :attr_org       "#+attr_org:"
  :begin_quote    "#+begin_quote"
  :end_quote      "#+end_quote"
  :caption        "#+caption:"
  :header         "#+header:"
  :begin_export   "#+begin_export"
  :end_export     "#+end_export"
  :results        "#+RESULTS:"
  :property       ":PROPERTIES:"
  :end            ":END:"
  :priority_a     "[#A]"
  :priority_b     "[#B]"
  :priority_c     "[#C]"
  :priority_d     "[#D]"
  :priority_e     "[#E]")
```

```
(plist-put +ligatures-extra-symbols :name "")
```

2.2.5 Pretty tables

```
(package! org-pretty-table :recipe (:host github :repo "Fuco1/org-pretty-table")
 ↪  :pin "7bd68b420d3402826fea16ee5099d04aa9879b78")
```

```

(use-package! org-pretty-table
  :after org
  :commands (org-pretty-table-mode global-org-pretty-table-mode))
```

2.2.6 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)
```

```
(use-package! org-fragtog
  :hook (org-mode . org-fragtog-mode))
```

2.3 Agenda

2.3.1 Defaults

```
;; org-agenda-config
(after! org-agenda
  (setq org-agenda-files (list "~/org/agenda.org"
                               "~/org/todo.org")))
```

2.3.2 Visual

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

2.4 Super-Agenda

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

2.4.1 Config

```

(after! org-super-agenda
  :commands org-super-agenda-mode)

(after! org-agenda
  (org-super-agenda-mode))

(setq org-agenda-skip-scheduled-if-done t
      org-agenda-skip-deadline-if-done t
      org-agenda-include-deadlines t
      org-agenda-block-separator nil
      org-agenda-tags-column 100 ;; from testing this seems to be a good value
      org-agenda-compact-blocks t)
```

2.4.2 Customize

```
(setq org-agenda-custom-commands
      '(("o" "Overview"
         ((agenda "" ((org-agenda-span 'day)
                      (org-super-agenda-groups
                       '((:name "Today"
                          :time-grid t
                          :date today
                          :todo "TODAY"
                          :scheduled today
                          :order 1)))))
          (alltodo "" ((org-agenda-overriding-header "")
                       (org-super-agenda-groups
                        '((:name "Next to do"
                           :todo "NEXT"
                           :order 1)
```

```
                        (:name "Important"
                         :tag "Important"
                         :priority "A"
                         :order 6)
                        (:name "Due Today"
                         :deadline today
                         :order 2)
                        (:name "Due Soon"
                         :deadline future
                         :order 8)
                        (:name "Overdue"
                         :deadline past
                         :face error
                         :order 7)
                        (:name "Assignments"
                         :tag "Assignment"
                         :order 10)
                        (:name "Issues"
                         :tag "Issue"
                         :order 12)
                        (:name "Emacs"
                         :tag "Emacs"
                         :order 13)
                        (:name "Projects"
                         :tag "Project"
                         :order 14)
                        (:name "Research"
                         :tag "Research"
                         :order 15)
                        (:name "To read"
                         :tag "Read"
                         :order 30)
                        (:name "Waiting"
                         :todo "WAITING"
                         :order 20)
                        (:name "University"
                         :tag "uni"
                         :order 32)
                        (:name "Trivial"
                         :priority<= "E"
```

```
                            :tag ("Trivial" "Unimportant")
                            :todo ("SOMEDAY" )
                            :order 90)
                           (:discard (:tag ("Chore" "Routine" "Daily")))))))))))
```

2.5 Roam

2.5.1 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))

```

```
(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)))
```

2.5.2 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"))))
```

2.5.3 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))))
```

2.6 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}"))))))
```

2.7 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)
```

2.8 Export

2.8.1 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.")
```

2.9 Zotero Integration

```
(package! zotxt)
```

```

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

2.10 Org-Chef

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

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

2.11 Bibtex-Integration

2.11.1 Citar

```
(package! citeproc)
(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"))
  (org-cite-insert-processor 'citar)
  (org-cite-follow-processor 'citar)
  (org-cite-activate-processor 'citar)
  (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! citeproc
  :defer t)

 (use-package! pdf-tool
 :hook (pdf-tools-enabled . pdf-view-themed-minor-mode ))
;;; Org-Cite configuration

(use-package! oc
  :after org citar
  :config
  (require 'ox)
  (setq org-cite-global-bibliography org-cite-global-bibliography)
  ;; setup export processor; default csl/citeproc-el, with biblatex for latex
  (setq org-cite-export-processors
        '((t csl))))

  ;;; Org-cite processors
(use-package! oc-biblatex
  :after oc)

(use-package! oc-csl
  :after oc
  :config
```

```
  (setq org-cite-csl-styles-dir "~/Zotero/styles"))

(use-package! oc-natbib
  :after oc)
```

```
(use-package! oc-csl-activate
  :after oc
  :config
  (setq org-cite-csl-activate-use-document-style t)
  (setq org-cite-csl-activate-use-citar-cache t)
  (defun +org-cite-csl-activate/enable ()
    (interactive)
    (setq org-cite-activate-processor 'csl-activate)
    (add-hook 'org-mode-hook '((lambda () (cursor-sensor-mode 1))
    ↪  org-cite-csl-activate-render-all))
    (defadvice! +org-cite-csl-activate-render-all-silent (orig-fn)
      :around #'org-cite-csl-activate-render-all
      (with-silent-modifications (funcall orig-fn)))
    (when (eq major-mode 'org-mode)
      (with-silent-modifications
        (save-excursion
          (goto-char (point-min))
          (org-cite-activate (point-max)))
        (org-cite-csl-activate-render-all)))
    (fmakunbound #'+org-cite-csl-activate/enable)))
```

2.12 Latex templates

2.12.1 Preview

1. PNG

```
(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 3))
```

```
(setq org-preview-latex-default-process 'dvisvgm)
```

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}
")

```

2.12.2 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[authoryear,longnamesfirst]{natbib}
\\usepackage{xurl}
\\definecolor{mint}{HTML}{d73a49}
\\usepackage[colorlinks=true, allcolors= mint]{hyperref}
\\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}{gray}{0.9}
[NO-DEFAULT-PACKAGES]
```

```
[PACKAGES]
[EXTRA]
\\usemintedstyle{vs}"
               ("\\section{%s}" . "\\section*{%s}")
               ("\\subsection{%s}" . "\\subsection*{%s}")
               ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
               ("\\paragraph{%s}" . "\\paragraph*{%s}"))))
```

2.12.3 Beamer

```
(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[authoryear]{natbib}
\\usepackage{xurl}
\\bibliographystyle{ecta}
\\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}{4c4f69}
\\definecolor{umber}{HTML}{dc8a78}
\\definecolor{alertcolor}{HTML}{dd7878}
\\definecolor{examplecolor}{HTML}{209fb5}

\\definecolor{pale}{HTML}{eff1f5}
\\definecolor{bluish}{HTML}{8c8fa1}
\\definecolor{cream}{HTML}{e6e9ef}
\\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}
\\definecolor{bgcolorminted}{gray}{0.9}
[NO-DEFAULT-PACKAGES]
[PACKAGES]
[EXTRA]
\\usemintedstyle{vs}"
               ("\\section{%s}" . "\\section*{%s}")
               ("\\subsection{%s}" . "\\subsection*{%s}")
               ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
               ("\\paragraph{%s}" . "\\paragraph*{%s}")
               ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
```

2.12.4 Export

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

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

```
(setq org-latex-listings 'minted
      org-latex-packages-alist '(("" "minted")))
(setq org-latex-minted-options '(("breaklines" "true")
                                 ("breakanywhere" "true")
                                 ("bgcolor" "bgcolorminted")
                                 ("linenos" "true")))
```

2.13 Capture

2.13.1 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))))
```

```
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
                                      ↪  template) 4) 2 5) template))
16                                                 (spec (plist-get (plist-get
                                                  ↪  props :doct) :icon)))
17                                       (setf (nth 1 template) (concat
                                          ↪  (+doct-icon-declaration-to-icon spec)
18                                                                      "\t"
19                                                                      (nth 1
                                                                         ↪  templa
                                                                         ↪  te))))
20                                     template)
21                                   templates))))
22
23  (setq doct-after-conversion-functions '(+doct-iconify-capture-templates))
24
25  (defvar +org-capture-recipies  "~/Org/recipies.org")
26
27  (defun set-org-capture-templates ()
28    (setq org-capture-templates
29          (doct `(("Personal todo" :keys "t"
30                   :icon ("checklist" :set "octicon" :color "green")
31                   :file +org-capture-todo-file
32                   :prepend t
33                   :headline "Inbox"
34                   :type entry
35                   :template ("* TODO %?"
36                              "%i %a"))
37                  ("Personal note" :keys "n"
38                   :icon ("sticky-note-o" :set "faicon" :color "green")
39                   :file +org-capture-todo-file
40                   :prepend t
41                   :headline "Inbox"
42                   :type entry
43                   :template ("* %?"
44                              "%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"))
                ("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)
```

3 Company

Improve the history size:

```
(after! company
  (setq company-idle-delay 0.5
        company-minimum-prefix-length 2)
  (setq company-show-numbers t)
  (add-hook! 'evil-normal-state-entry-hook #'company-abort)) ;; 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))
```

4 LSP

4.1 Digestif

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

4.2 LTex

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

```
(use-package! lsp-ltex
  :hook (text-mode . (lambda ()
                       (require 'lsp-ltex)
                       (lsp)))  ; or lsp-deferred
  :config
  (setq lsp-ltex-version "15.2.0"))  ; make sure you have set this, see below
```

5 VLFI

```
(package! vlfi)
```

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

6 PDF-Tools

6.1 Fix annotation bug

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

7 Option key Fix

```
(defun iensu/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)))
```

8 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)
```