Literate Doom Emacs Config

Jonathan Fung

Resources

444 Days of Literate Configuration

`https://www.reddit.com/r/emacs/comments/m9zxw6/444_days_of_literate_configuration_450_days_using/`

`https://github.com/tecosaur/emacs-config/`

Emacs 28 Info

`https://www.reddit.com/r/emacs/comments/kev7o8/aur_arch_linux_package_delivering_emacs_binaries/`

`emacs` was assimilating to 27.1, even with rm -rf–ing .emacs.d. I had to delete my extra Emacs 27.1 bin.

Just reinstalled doom in .emacs.d, and synced

Emacs 28 Fix: define-obsolete-function-alias

From `https://www.reddit.com/r/emacs/comments/kqb9s9/cannot_recompile_packagess_error_wrong_number_of/`

```
(defun make-obsolete (obsolete-name current-name &optional when)
  "Make the byte-compiler warn that function OBSOLETE-NAME is obsolete.
OBSOLETE-NAME should be a function name or macro name (a symbol).

The warning will say that CURRENT-NAME should be used instead.
If CURRENT-NAME is a string, that is the `use instead' message
\(it should end with a period, and not start with a capital).
WHEN should be a string indicating when the function
was first made obsolete, for example a date or a release number."
  (declare (advertised-calling-convention
            ;; New code should always provide the `when' argument.
            (obsolete-name current-name when) "23.1"))
  (put obsolete-name 'byte-obsolete-info
       ;; The second entry used to hold the `byte-compile' handler, but
       ;; is not used any more nowadays.
       (purecopy (list current-name nil when)))
  obsolete-name)

(defmacro define-obsolete-function-alias (obsolete-name current-name
                                                        &optional when docstring)
  "Set OBSOLETE-NAME's function definition to CURRENT-NAME and mark it obsolete.

\(define-obsolete-function-alias \\='old-fun \\='new-fun \"22.1\" \"old-fun's
↪  doc.\")

is equivalent to the following two lines of code:

\(defalias \\='old-fun \\='new-fun \"old-fun's doc.\")
\(make-obsolete \\='old-fun \\='new-fun \"22.1\")

WHEN should be a string indicating when the function was first
made obsolete, for example a date or a release number.

See the docstrings of `defalias' and `make-obsolete' for more details."
  (declare (doc-string 4)
           (advertised-calling-convention
            ;; New code should always provide the `when' argument.
            (obsolete-name current-name when &optional docstring) "23.1"))
  `(progn
     (defalias ,obsolete-name ,current-name ,docstring)
     (make-obsolete ,obsolete-name ,current-name ,when)))

(defun make-obsolete-variable (obsolete-name current-name &optional when
↪  access-type)
  "Make the byte-compiler warn that OBSOLETE-NAME is obsolete.
The warning will say that CURRENT-NAME should be used instead.
If CURRENT-NAME is a string, that is the `use instead' message.
WHEN should be a string indicating when the variable
was first made obsolete, for example a date or a release number.
ACCESS-TYPE if non-nil should specify the kind of access that will trigger
  obsolescence warnings; it can be either `get' or `set'."
  (declare (advertised-calling-convention
            ;; New code should always provide the `when' argument.
            (obsolete-name current-name when &optional access-type) "23.1"))
  (put obsolete-name 'byte-obsolete-variable
       (purecopy (list current-name access-type when)))
  obsolete-name)

(defmacro define-obsolete-variable-alias (obsolete-name current-name
```

Information

User Info

Some functionality uses this to identify you, e.g. GPG configuration, email clients, file templates and snippets.

```
(setq user-full-name  "Jonathan Fung"
      user-mail-address "jonathanfung2000@gmail.com")
```

Doom Info

Here are some additional functions/macros that could help you configure Doom:

- 'load!' for loading external *.el files relative to this one
- 'use-package' for configuring packages
- 'after!' for running code after a package has loaded
- 'add-load-path!' for adding directories to the 'load-path', relative to this file. Emacs searches the 'load-path' when you load packages with 'require' or 'use-package'.
- 'map!' for binding new keys

To get info about any of these functions/macros, move the cursor over the highlighted symbol at press 'K' (non-evil users must press 'C-c g k'). This will open documentation for it, including demos of how they are used.

You can also try 'gd' (or 'C-c g d') to jump to their definition and see how they are implemented.

Visual Interface

Fonts

Doom exposes five (optional) variables for controlling fonts in Doom. Here are the three important ones:

- 'doom-font'
- 'doom-variable-pitch-font'
- 'doom-big-font' – used for 'doom-big-font-mode'; use this for presentations or streaming.

They all accept either a font-spec, font string ("Input Mono-12"), or xlfd font string. You generally only need these two:

```
;; (setq doom-font (font-spec :family "Source Code Pro" :size 24))
;; (setq doom-big-font (font-spec :family "Source Code Pro" :size 36))

;; (setq doom-font (font-spec :family "JetBrains Mono" :weight 'light :size 24))
;; (setq doom-big-font (font-spec :family "JetBrains Mono" :weight 'light :size
↪  36))
;; (setq doom-variable-pitch-font (font-spec :family "Overpass" :weight 'bold
↪  :size 24))

;; (setq doom-variable-pitch-font (font-spec :family "Roboto" :size 24))

;; variable-pitch et al seems to inherit ":weight 'light" from doom-font
(setq doom-font (font-spec :family "JetBrains Mono" :size 16)
      ;; 24
      doom-big-font (font-spec :family "JetBrains Mono" :size 36)
      ;; doom-variable-pitch-font (font-spec :family "Source Sans Pro" :size 36)
      ;; doom-variable-pitch-font (font-spec :family "Roboto" :size 24)
      doom-variable-pitch-font (font-spec :family "IBM Plex Sans" :size 16 :weight
      ↪  'semibold)
      ;; doom-variable-pitch-font (font-spec :family "IBM Plex Sans Condensed"
      ↪  :size 24 :weight 'normal)
      ;; doom-variable-pitch-font (font-spec :family "IBM Plex Serif" :size 24
      ↪  :weight 'normal)
      doom-serif-font (font-spec :family "IBM Plex Mono" :weight 'light))
```

Theme

Modus is included in Emacs 28.1

There are two ways to load a theme. Both assume the theme is installed and available. You can either set 'doom-theme' or manually load a theme with the 'load-theme' function.

Reference: `https://protesilaos.com/modus-themes`

As of 3/14/21, my current emacs version is GNU Emacs 28.0.50 (build 2, $x86_{64}$-pc-linux-gnu, GTK+ Version 3.24.24, cairo version 1.17.4) of 2021-01-30 so, modus is not in native emacs

```
;; https://github.com/hlissner/doom-emacs/issues/3967
;; (setq doom-theme 'modus-operandi)
(setq doom-theme 'modus-vivendi)

;; (require 'modus-themes)                  ; common code
;; (require 'modus-operandi-theme)          ; light theme
;; (require 'modus-vivendi-theme)           ; dark theme

;; (load-theme 'modus-vivendi)           ; Dark theme
;; (load-theme 'modus-operandi)          ; Light theme

;; (global-set-key (kbd "<f5>") (lambda () (interactive) (modus-themes-toggle)
↪  (set-face-background 'mode-line "default")))

;; for terminal use, to see window divides
(global-set-key (kbd "<f5>") 'modus-themes-toggle)

;; Set customization options to values of your choice
(setq modus-themes-slanted-constructs t
      modus-themes-bold-constructs t
      modus-themes-fringes 'intense ; {nil,'subtle,'intense}

      ;; Options for `modus-themes-lang-checkers': nil,
      ;; 'straight-underline, 'subtle-foreground,
      ;; 'subtle-foreground-straight-underline, 'intense-foreground,
      ;; 'intense-foreground-straight-underline, 'colored-background
      modus-themes-lang-checkers 'colored-background

      modus-themes-mode-line nil ; {nil,'3d,'moody}

      modus-themes-syntax nil ; Lots of options---continue reading the manual

      modus-themes-intense-hl-line t
      modus-themes-paren-match 'intense ; {nil,'subtle-bold,'intense,'intense-bold}

      modus-themes-links nil ; Lots of options---continue reading the manual
      modus-themes-no-mixed-fonts nil
      modus-themes-prompts 'subtle ; {nil,'subtle,'intense}
      modus-themes-completions 'opinionated ; {nil,'moderate,'opinionated}
      modus-themes-region 'no-extend ; {nil,'no-extend,'bg-only,'bg-only-no-extend}
      modus-themes-diffs nil ; {nil,'desaturated,'fg-only,'bg-only}
      modus-themes-org-blocks 'grayscale ; {nil,'grayscale,'rainbow}

      modus-themes-headings ; Lots of options---continue reading the manual
      '((1 . rainbow-section)
        ;; (2 . rainbow-line-no-bold)
        ;; (3 . no-bold)
        ;; (t . rainbow)
        ;; (t . rainbow-line)
        (t . rainbow-line-no-bold)
        ;; (t . highlight-no-bold)
        )
      modus-themes-variable-pitch-headings nil
      modus-themes-scale-headings nil
      modus-themes-scale-1 1.1
      modus-themes-scale-2 1.15
      modus-themes-scale-3 1.21
      modus-themes-scale-4 1.27
      modus-themes-scale-5 1.33)
```

TODO Display

```
                                        ;includes part of the file's directory
                                        ↪  name at the beginning of the shared
                                        ↪  buffer name to make unique
(setq uniquify-buffer-name-style 'forward)
;; this may do the same thing as uniquify-buffer...
(setq ivy-rich-path-style 'abbrev)

;; (setq display-line-numbers-type 'visual)
(setq display-line-numbers-type nil)

                                        ; just editted these line 12/24
;; idk what these 2 lines do
;; (add-to-list 'default-frame-alist '(font . "Source Code Pro-10"))
;; (set-face-attribute 'default t :font "Source Code Pro-10")
```

TODO Modeline

```
;; CAUTION
;; This might be fatal, might turn off all keymaps
;; (setq display-battery-mode t)

;; (setq display-time-mode t)
(display-time-mode)
(setq display-time-default-load-average nil)
(setq line-number-mode nil
      column-number-mode nil)
(set-face-background 'mode-line "default")

(setq doom-modeline-buffer-encoding nil)
;; (setq doom-modeline-buffer-encoding t)
(setq doom-modeline-buffer-file-name-style 'relative-from-project)
```

hl-line mode

```
(setq hl-line-mode nil)
(map! :n "SPC t h" #'hl-line-mode)

; meant to only have hl-line highlight on end of line
(defun my-hl-line-range-function () (cons (line-end-position)
↪  (line-beginning-position 2)))
;(setq hl-line-range-function #'my-hl-line-range-function)

; standard full-width
(defun my-hl-line-range ()
  "Used as value of `hl-line-range-function'."
  (cons (line-beginning-position) (line-end-position)))

(setq-default hl-line-range-function #'my-hl-line-range)
```

Line Spacing

```
(defun jf/toggle-line-spacing ()
  "Toggle line spacing between no extra space to extra half line height.
URL `http://ergoemacs.org/emacs/emacs_toggle_line_spacing.html'
Version 2017-06-02"
  (interactive)
  (if line-spacing
      (setq line-spacing nil)
    (setq line-spacing 5))
  (redraw-frame (selected-frame)))

(map! :n "SPC t v" 'jf/toggle-line-spacing)
```

evil-mode terminal cursors

https://github.com/h0d/term-cursor.el

```
;;; term-cursor.el --- Change cursor shape in terminal -*- lexical-binding: t;
↪  coding: utf-8; -*-

;; Version: 0.4
;; Author: h0d
;; URL: https://github.com/h0d
;; Keywords: terminals
;; Package-Requires: ((emacs "26.1"))

;;; Commentary:

;; Send terminal escape codes to change cursor shape in TTY Emacs.
;; Using VT520 DECSCUSR (cf
↪  https://invisible-island.net/xterm/ctlseqs/ctlseqs.html).
;; Does not interfere with GUI Emacs behavior.

;;; Code:

(defgroup term-cursor nil
  "Group for term-cursor."
  :group 'terminals
  :prefix 'term-cursor-)

;; Define escape codes for different cursors
(defcustom term-cursor-block-blinking "\e[1 q"
  "The escape code sent to terminal to set the cursor as a blinking box."
  :type 'string
  :group 'term-cursor)

(defcustom term-cursor-block-steady "\e[2 q"
  "The escape code sent to terminal to set the cursor as a steady box."
  :type 'string
  :group 'term-cursor)

(defcustom term-cursor-underline-blinking "\e[3 q"
  "The escape code sent to terminal to set the cursor as a blinking underscore."
  :type 'string
  :group 'term-cursor)

(defcustom term-cursor-underline-steady "\e[4 q"
  "The escape code sent to terminal to set the cursor as a steady underscore."
  :type 'string
  :group 'term-cursor)

(defcustom term-cursor-bar-blinking "\e[5 q"
  "The escape code sent to terminal to set the cursor as a blinking bar."
  :type 'string
  :group 'term-cursor)

(defcustom term-cursor-bar-steady "\e[6 q"
  "The escape code sent to terminal to set the cursor as a steady bar."
  :type 'string
  :group 'term-cursor)

;; Current cursor evaluation
(defcustom term-cursor-triggers (list 'blink-cursor-mode-hook
↪  'lsp-ui-doc-frame-hook)
  "Hooks to add when the variable watcher might not be enough.
That is, hooks to trigger `term-cursor--immediate'."
  :type 'list
```

Usability

Navigation

```
; Bind Zooms??
(map! :n "C-_" #'er/contract-region
      :n "C-+" #'er/expand-region)

;; ; unbind J,K,M
(map! :map evil-normal-state-map "J" nil
      "K" nil)
(map! :map evil-motion-state-map "M" nil
      "K" nil)

;; ; rebind J,K for scrolling
(map! :n "J" #'evil-scroll-line-up)
(map! :n "K" #'evil-scroll-line-down)

;; ; bind M for contexual lookup
(map! :n "M" #'+lookup/documentation)

;; ;; Make evil-mode up/down operate in screen lines instead of actual lines
(define-key evil-motion-state-map "j" 'evil-next-visual-line)
(define-key evil-motion-state-map "k" 'evil-previous-visual-line)
;; ;; Also in visual mode
(define-key evil-visual-state-map "j" 'evil-next-visual-line)
(define-key evil-visual-state-map "k" 'evil-previous-visual-line)
```

Screenshots

```
(defun screenshot-svg ()
  "Save a screenshot of the current frame as an SVG image.
Saves to a temp file and puts the filename in the kill ring."
  (interactive)
  (let* ((filename (make-temp-file "Emacs" nil ".svg"))
         (data (x-export-frames nil 'svg)))
    (with-temp-file filename
      (insert data))
    (kill-new filename)
    (message filename)))
```

pdf-tools

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

Custom Keybinds

```
;; Bind toggles
(global-set-key (kbd "<f2>") 'mixed-pitch-mode)
(global-set-key (kbd "<f3>") 'olivetti-mode)
(global-set-key (kbd "<f4>") 'toggle-rot13-mode)
(setq olivetti-body-width 110)
;; (global-set-key (kbd "U") 'undo-tree-redo)

;; Unbind language input switcher
(map! :map global-map "C-\\" nil)

;; Bind toggle for 80-char limit, buffer-wide
(map! :n "SPC t c" 'display-fill-column-indicator-mode)
(map! :n "C-\\" 'display-fill-column-indicator-mode)

;; currently do not use org-roam, need to delete
;; (setq org-roam-directory "~/emacs/org-roam")
;; (setq org-roam-index-file "index.org")
;; (define-key org-roam-mode-map (kbd "C-c n l") #'org-roam)
;; (define-key org-roam-mode-map (kbd "C-c n f") #'org-roam-find-file)
;; (define-key org-roam-mode-map (kbd "C-c n j") #'org-roam-jump-to-index)
;; (define-key org-roam-mode-map (kbd "C-c n b") #'org-roam-switch-to-buffer)
;; (define-key org-roam-mode-map (kbd "C-c n g") #'org-roam-graph)
;; (define-key org-mode-map (kbd "C-c n i") #'org-roam-insert)
;; (require 'org-roam-protocol)
```

undo-tree

```
(setq global-undo-tree-mode t)
```

save-some-buffers

```
(map! :n "SPC f a" 'save-some-buffers)

;; (map! :map org-agenda-mode-map "SPC f a" 'save-some-buffers)

(map! :map doom-leader-map "f a" 'save-some-buffers)
```

find-buffer-pdf

`https://www.reddit.com/r/emacs/comments/mzuaf6/how_open_a_pdf_file_with_pdftools_when_you_are/`

My own function

```
(defun jf/find-buffer-pdf ()
  (interactive)
  (find-file (concat
              "./"
              (car (split-string (buffer-name)
                                 "\\."))
              ".pdf")))

(map! :n "SPC o p" 'jf/find-buffer-pdf)
```

Org-Mode

Org

```
(setq org-directory "~/org/")

(setq org-ellipsis " ")
(setq org-startup-folded 'content)

(add-hook 'org-mode-hook (lambda () (org-superstar-mode 1)))
;; (setq org-superstar-headline-bullets-list
;;       '("" ("" ?) "" ""))

;; https://www.reddit.com/r/emacs/comments/lapujj/weekly_tipstricketc_thread/glvoi
↪  fj/
(setq org-superstar-headline-bullets-list '("" "" "" "" "" "" "" "" "" "o"))

(map! :n "SPC o l" 'link-hint-open-link-at-point)
```

a

b

c

1. d
    (a) e
        i. f
            A. g
            B. h
            C. i

TODO Org-Inline Task

https://mattduck.github.io/generic-css/demo/org-demo.html https://www.reddit.com/r/emacs/comments/3tpd5z/a_different_way_to_use_org_xpost_rorgmode/ https://github.com/amluto/org-mode/blob/master/lisp/org-inlinetask.el https://irreal.org/blog/?p=8418

```
;; seems to break doom config ?
;; (require 'org-inlinetask)
```

Org Agenda + Super Agenda

Setup (Super) Agenda

```
;; https://www.reddit.com/r/orgmode/comments/6q6cdk/adding_files_to_the_agenda_lis
↪  t_recursively/
;; doom doctor: org-agenda-file-regexp seems to be void
;; (setq org-agenda-files (apply 'append
;;                    (mapcar
;;                     (lambda (directory)
;;                    (directory-files-recursively
;;                     directory org-agenda-file-regexp))
;;                     '("~/School/W21/" "~/org/"))))

;; Need to manually update based on school term
(setq org-agenda-files '("~/org"
                         "~/org/blog"
                         "~/org/voxpop"
                         "~/org/resources"
                         "~/School/S21/ENGR_165_Manuf"
                         "~/School/S21/MSE_165C_Phase"
                         "~/School/S21/MSE_189C_Snr"
                         "~/School/S21/STATS_120C_Prob"
                         "~/rust/effex"
                         ))

(setq org-tag-faces
      '(("Phase" . "gold2")
        ("Nano" . "lime green")
        ("Manuf" . "red2")
        ("Snr" . "medium orchid")
        ("Stats" . "dodger blue")))

(setq org-agenda-start-day "+0"
      org-agenda-span 1) ;; for use with day-by-day view

(setq org-agenda-timegrid-use-ampm t)
(setq org-agenda-time-grid
      (quote
       ((daily today require-timed)
        (400 1200 1600 2000 2400)
        "  " "----------------------")))  ; 2400 is the next day

(setq org-super-agenda-date-format "%A, %e %b")
(setq org-super-agenda-header-separator ?-)
;; (setq org-super-agenda-header-separator "")
(org-super-agenda-mode)

(map! :map org-super-agenda-header-map "k" nil
      "j" nil)

                                        ; removes 'agenda' prefix coming from
                                        ↪  agenda.org
                                        ; also adds in effort level
                                        ; should be (todo   . " %i %-12:c") if
                                        ↪  using multiple files
(setq org-agenda-prefix-format
      '(
        ;; (agenda . "%i %-7T%?-12t% s")
        (agenda . "%i %?-12t% s")
```

Day-by-Day + Regular (Super) Agenda Views

```
(setq org-agenda-custom-commands
      '(("z" "Super View, Everyday"
         (
          (agenda "" ((org-super-agenda-groups
                       '((:name ""
                          :time-grid t
                          :date today
                          :deadline today
                          ;; :scheduled today
                          :order 0
                          :discard (:anything t)
                          )))))
          (alltodo "" ((org-agenda-overriding-header (concat
                                                      (make-string 1 ?\n)
                                                      "Today is " (org-read-date
                                                      ↪  nil nil "+0d")
                                                      ))
                       (org-super-agenda-groups
                        '(
                          (:name "Overdue"
                           :deadline past
                           :order 0)
                          (:name "Scheduled"
                           :auto-planning t
                           :order 0)
                          (:name "========\n Personal"
                           :tag "Person"
                           :order 10)
                          (:name "Email"
                           :tag "Email"
                           :order 15)
                          (:discard (:anything t))
                          ))))
          ))))

(defun jf/org-agenda-day-by-day ()
  (interactive)
  (org-agenda nil "z"))
(map! :n "SPC o v" 'jf/org-agenda-day-by-day)

(defun jf/org-agenda-regular-view ()
  (interactive)
  (org-agenda nil "a"))
(map! :n "SPC o c" 'jf/org-agenda-regular-view)
```

Relative (Super) Agenda Views

```
;; from https://github.com/alphapapa/org-super-agenda/issues/59
;; function is needed to always eval relative dates
(defun jf/org-agenda-relative-deadline ()
  (interactive)
  (let ((org-super-agenda-groups
         `(
           (:name "Past"
            :deadline past)
           (:name "Next Items"
            :todo "NEXT")
           (:name "Clean up Notes"
            :todo "NOTE")
           (:name "Today's Time Blocks"
            :and (:todo "BLOCK"
                  :date today))
           (:name "Today"
            :deadline today)
           (:name "Tomorrow (+1)"
            ;; before acts as <
            :deadline (before ,(org-read-date nil nil "+2d")))
           (:name "Tomorrow Tomorrow (+2)"
            ;; if today is 1, should show (before (1+3)) = 1, 2,3
            :deadline (before ,(org-read-date nil nil "+3d")))
           (:name "Day After Tomorrow Tomorrow (+3)"
            :deadline (before ,(org-read-date nil nil "+4d")))
           (:name "Within a Week (+4..6)"
            :deadline (before ,(org-read-date nil nil "+7d")))
           (:name "Within 30 Days (+7..30)"
            :deadline (before ,(org-read_date nil nil "+31d")))
           (:name "========\n Personal"
            :tag "Person"
            :order 10)
           (:name "Email"
            :tag "Email"
            :order 15)
           (:discard (:anything t))
           )))
    (org-agenda nil "t")
    ;; (org-agenda-list)
    ;; this allows time grid to show with TODO, but doesn't catch
    ;; NEXT, Personal, and doesn't extend to 30 days
    ;; Text is also red for some reason
    ))

;; see https://github.com/alphapapa/org-super-agenda/issues/153
;; for a combined deadline-scheduled view with repeating items

(defun jf/org-agenda-relative-scheduled ()
  (interactive)
  (let ((org-super-agenda-groups
         `(
           (:name "Past"
            :scheduled past)
           (:name "Next Items"
            :todo "NEXT")
           (:name "Clean up Notes"
            :todo "NOTE")
           (:name "Today's Time Blocks"
```

Hydra Map for Agenda Views

```
(defhydra jf/hydra-agenda (:color blue
                           :hint nil)
  "
^Relative^      ^Absolute^      ^Time-Grid^
^^^-------------------------------------
_d_: deadline   _e_: everyday   _w_: regular
_s_: scheduled
  "
  ("d" jf/org-agenda-relative-deadline)
  ("s" jf/org-agenda-relative-scheduled)
  ("e" jf/org-agenda-day-by-day)
  ("w" jf/org-agenda-regular-view)
  )

;; (map! :n "SPC a" 'jf/hydra-agenda/body)
(map! :map doom-leader-map "a" 'jf/hydra-agenda/body)
```

Org-Todo

```
(setq org-todo-keywords
      '((sequence "TODO(t)" "NEXT(n)" "NOTE(m)" "BLOCK(b)" "STRT(s)" "HOLD(h)" "|"
      ↪  "DONE(d)" "KILL(k)")
        (sequence "[ ](T)" "[+](P)" "[-](S)" "[?](W)" "|" "[X](D)")))

(setq org-todo-keyword-faces
      '(("[-]" . +org-todo-active)
        ("STRT" . +org-todo-active)
        ("BLOCK" . +org-todo-active)
        ("NEXT" . +org-todo-active)
        ("[?]" . +org-todo-onhold)
        ("WAIT" . +org-todo-onhold)
        ("HOLD" . +org-todo-onhold)
        ("PROJ" . +org-todo-project)))

;; sort todos by deadline earliest first, then priority high first
(setq org-agenda-sorting-strategy
      '((agenda habit-down time-up priority-down category-keep)
        (todo deadline-up priority-down category-keep)
        (tags priority-down category-keep)
        (search category-keep)) )

;; when set to t, toggling a repeating TODO item to DONE will reset the TODO
↪  prefix to the previous one
;; implemented when switching BLOCK -> DONE, which returns it to BLOCK and not TODO
;; reason: super-agenda selector is based on TODO name
(setq org-todo-repeat-to-state t)
```

Org Capture

```
(setq org-capture-templates
      '(("t" "Agenda TODO" entry (file "~/org/Agenda.org")
        "* TODO %? \n DEADLINE: %t" :prepend t)
        ("e" "email" entry (file+headline "~/org/Agenda.org" "Emails")
         "* TODO Reply: %? \n - %a" :prepend t)
        ("d" "designboard" entry (file "~/org/designboard.org")
         "* %? \n- %t" :prepend t)
      ))

(map! :n "SPC z" 'org-capture)
```

Org Export - Latex

```
(setq org-latex-classes '(("article" "\\documentclass[11pt]{article}"
                           ("\\section{%s}" . "\\section*{%s}")
                           ("\\subsection{%s}" . "\\subsection*{%s}")
                           ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
                           ("\\paragraph{%s}" . "\\paragraph*{%s}")
                           ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))
                          ("report" "\\documentclass[11pt]{report}"
                           ("\\part{%s}" . "\\part*{%s}")
                           ("\\chapter{%s}" . "\\chapter*{%s}")
                           ("\\section{%s}" . "\\section*{%s}")
                           ("\\subsection{%s}" . "\\subsection*{%s}")
                           ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
                          ("book" "\\documentclass[11pt]{book}"
                           ("\\part{%s}" . "\\part*{%s}")
                           ("\\chapter{%s}" . "\\chapter*{%s}")
                           ("\\section{%s}" . "\\section*{%s}")
                           ("\\subsection{%s}" . "\\subsection*{%s}")
                           ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
                          ("notes"
                           "\\documentclass[8pt]{article}
  \\usepackage[letterpaper, portrait, margin=1in]{geometry}
  \\usepackage[utf8]{inputenc}
  \\usepackage[T1]{fontenc}
  \\usepackage{amsmath}
  \\usepackage{amssymb}
  \\usepackage{hyperref}
  \\usepackage{tcolorbox}
% http://tug.ctan.org/macros/latex/contrib/minted/minted.pdf
  \\usepackage[cache=false]{minted}
  \\setminted{breaklines=true, breakanywhere=true}
% \\usemintedstyle{paraiso-light} % pygmentize -L styles
  \\usemintedstyle{emacs} % pygmentize -L styles
% https://tex.stackexchange.com/questions/112559/box-around-minted-environment
% https://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/tcolorbox/tcolo
↪  rbox.pdf
  \\BeforeBeginEnvironment{minted}{\\begin{tcolorbox}[colframe=black!85!white,
↪  colback=black!5!white, boxrule=0.3mm]}
  \\AfterEndEnvironment{minted}{\\end{tcolorbox}}
  \\usepackage{enumitem}
  \\setitemize{itemsep=0.5pt}
  \\usepackage{lastpage}
  \\usepackage{fancyhdr}
  \\pagestyle{fancy}
  \\fancyhf{}
  \\usepackage{titling} % allows \thetitle \theauthor \thedate
  \\rhead{\\theauthor}
  \\lhead{\\thetitle}
  \\rfoot{\\thepage{} of \\pageref{LastPage}}
  \\linespread{1}
  \\setlength{\\parindent}{0pt}
  \\setlength{\\parskip}{0.5em plus 0.1em minus 0.2em}
  \\hypersetup{pdfborder=0 0 0}
  \\setcounter{secnumdepth}{0}"
                           ("\\section{%s}" . "\\section*{%s}")
                           ("\\subsection{%s}" . "\\subsection*{%s}")
                           ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
                           ("\\paragraph{%s}" . "\\paragraph*{%s}")
                           ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
```

Org-Indent

```
(map! :n "SPC t i" #'org-indent-mode)
```

TODO Autoinsert

Taken from: `https://emacs.stackexchange.com/questions/34651/how-can-i-create-custom-org-mode-templates`

```
(use-package autoinsert
  :init
  ;; Don't want to be prompted before insertion:
  (setq auto-insert-query nil)

  (setq auto-insert-directory (locate-user-emacs-file "templates"))
  (add-hook 'find-file-hook 'auto-insert)
  (auto-insert-mode 1)

  ;; directory is ~/.emacs.d/local/etc/templates
  :config
  (define-auto-insert "\\.org?$" "default-autoinsert.org"))
```

Source Block Markup

`https://www.reddit.com/r/emacs/comments/kzo09h/emacs_configuration_created_for_heavy_orgmode/`
`https://github.com/lijigang/config-orgmode-within-doom`

```
  (defvar rasmus/ob-header-symbol ?
    "Symbol used for babel headers")

  (defun rasmus/org-prettify-symbols ()
    (interactive)
    (mapc (apply-partially 'add-to-list 'prettify-symbols-alist)
          (cl-reduce 'append
                     (mapcar (lambda (x) (list x (cons (upcase (car x)) (cdr x))))
                             `(("#+begin_src" . ?) ;;
                               ("#+end_src"   . ?) ;;
                               ("#+header:" . ,rasmus/ob-header-symbol)
                               ("#+begin_quote" . ?)
                               ("#+end_quote" . ?)))))
    (turn-on-prettify-symbols-mode))

  (add-hook 'org-mode-hook #'rasmus/org-prettify-symbols)
```

Applications

Swiper - In-Buffer Fuzzy Finder

From r/emacs: By default if you have visual line mode on swiper scans every visual line, which can be really slow in large files. This forces swiper to revert back to searching only every actual line even if the user is

using visual line mode

Note: seems to only find one occurrence in each file line, user needs to scan main buffer for thorough results.

```
(setq swiper-use-visual-line nil)
(setq swiper-use-visual-line-p (lambda (a) nil))
```

Dired - File Manager

```
(add-hook 'dired-mode-hook
      (lambda ()
        (dired-hide-details-mode)
        ))
; add this into above hook to default to sorting by edit time
; (dired-sort-toggle-or-edit)
```

Treemacs - Sidebar Directory Viewer

Bind external (zathura, etc.) opening for treemacs

```
(map! :n "SPC o o" #'treemacs-visit-node-in-external-application)
(map! :n "SPC o t" #'treemacs)
(setq treemacs-position 'right
      treemacs-width 25
      treemacs-indentation 1)
```

TODO Notmuch - Email Client

```
;define function that syncs mbsync and refreshes notmuch
(defun jf/sync-email ()
  "Lists the contents of the current directory."
  (interactive)
  (shell-command "mbsync -a && notmuch new"))

; bind notmuch-hello view
(map! :n "SPC o n" #'notmuch-hello)
; bind custom function to sync mbsync and notmuch
(map! :n "SPC r s" 'jf/sync-email)

;; attempt to fix notmuch formatting
(setq notmuch-search-result-format
  '(("date" . "%12s ")
    ("count" . "%-6s ")
    ("authors" . "%-15s ")
    ("subject" . "%-10s ")
    ("tags" . "(%s)"))
)
(defun jf/establish-notmuch ()
  (interactive)
(setq notmuch-saved-searches '((:name "Personal"
                                :query "tag:inbox AND
                                ↪  to:jonathanfung2000@gmail.com AND
                                ↪  date:nov_3_2020..today AND NOT tag:delete")
                               (:name "UCI"
                                :query "tag:inbox AND to:fungjm@uci.edu AND
                                ↪  date:nov_3_2020..today AND NOT tag:delete")
                               (:name "Clean Gen Inbox"
                                :query "tag:inbox AND date:nov_3_2020..today AND
                                ↪  NOT to:fungjm@uci.edu AND NOT
                                ↪  to:jonathanfung2000@gmail.com")
                               (:name "Flagged"
                                :query "tag:inbox AND tag:flagged")
                               (:name "Inbox"
                                :query "tag:inbox"))))

(map! :n "SPC r e" 'jf/establish-notmuch)

; this sets cursor of notmuch-hellow to first saved search
(add-hook 'notmuch-hello-refresh-hook
          (lambda ()
            (if (and (eq (point) (point-min))
                     (search-forward "Saved searches:" nil t))
                (progn
                  (forward-line)
                  (widget-forward 1))
              (if (eq (widget-type (widget-at)) 'editable-field)
                  (beginning-of-line)))))

(setq notmuch-hello-sections '(notmuch-hello-insert-saved-searches))
```

TODO Inkscape

```
;;; scimax-inkscape.el --- Using inkscape in org-mode

;;; Commentary:
;;
;; This library provides a new org-mode link for inkscape svg files. When you
;; click on an inkscape link, it will open the figure in inkscape. A thumbnail
;; image will be placed on the inkscape link.
;;
;; Export to HTML:
;; (browse-url (let ((org-export-before-processing-hook
↪  '(scimax-inkscape-preprocess)))
;;   (org-html-export-to-html)))
;;
;; (org-open-file (let ((org-export-before-processing-hook
↪  '(scimax-inkscape-preprocess)))
;;   (org-latex-export-to-pdf)))
;;
;; inkscape does not allow you to create empty files. We save the template in a
;; variable and create them on demand.

(defcustom scimax-inkscape-thumbnail-width 300
  "Width of thumbnails in pts."
  :group 'scimax-inkscape
  :type 'integer)

(defcustom scimax-inkscape-template-svg
  "<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"no\"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->
<svg
   xmlns:dc=\"http://purl.org/dc/elements/1.1/\"
   xmlns:cc=\"http://creativecommons.org/ns#\"
   xmlns:rdf=\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\"
   xmlns:svg=\"http://www.w3.org/2000/svg\"
   xmlns=\"http://www.w3.org/2000/svg\"
   xmlns:sodipodi=\"http://sodipodi.sourceforge.net/DTD/sodipodi-0.dtd\"
   xmlns:inkscape=\"http://www.inkscape.org/namespaces/inkscape\"
   width=\"6in\"
   height=\"4in\"
   viewBox=\"0 100 152.4 201.6\"
   version=\"1.1\"
   id=\"svg8\"
   inkscape:version=\"0.92.2 (5c3e80d, 2017-08-06)\"
   sodipodi:docname=\"drawing.svg\">
  <defs
     id=\"defs2\" />
  <sodipodi:namedview
     id=\"base\"
     pagecolor=\"#ffffff\"
     bordercolor=\"#666666\"
     borderopacity=\"1.0\"
     inkscape:pageopacity=\"0.0\"
     inkscape:pageshadow=\"2\"
     inkscape:zoom=\"1\"
     inkscape:cx=\"400\"
     inkscape:cy=\"214.9\"
     inkscape:document-units=\"in\"
     inkscape:current-layer=\"layer1\"
     showgrid=\"false\"
```

TODO writer-word-goals

`https://www.reddit.com/r/emacs/comments/l2a22l/have_you_writers_ambitions_write_a_little_write/`
`https://github.com/ag91/writer-word-goals`

- does not allow for killing/stopping processes
  - killing buffer seems to stop it
- can also apply multiple trackers to the same buffer

```
;;; wwg.el --- writer word goals

;; Copyright (C) 2021 Andrea

;; Author: Andrea andrea-dev@hotmail.com>
;; Version: 0.0.0
;; Package-Version: 20210121
;; Keywords: writing

;; This program is free software; you can redistribute it and/or modify
;; it under the terms of the GNU General Public License as published by
;; the Free Software Foundation, either version 3 of the License, or
;; (at your option) any later version.

;; This program is distributed in the hope that it will be useful,
;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;; GNU General Public License for more details.

;; You should have received a copy of the GNU General Public License
;; along with this program.  If not, see <http://www.gnu.org/licenses/>.

;;; Commentary:

;; A word number countdown for your writing goals.
;;
;; This mode helps you staying focused on the number of words you
;; setup for your goals. The more you write the closer you get to your
;; self-set goal for this session.
;;
;; See documentation on https://github.com/ag91/writer-word-goals

;;; Code:

(defgroup wwg nil
  "Options specific to wwg."
  :tag "wwg"
  :group 'wwg)

(defvar wwg/active-timer-alist nil "Alist of (buffer . timer) bindings to cleanup
↪  achieved targets.")

(defcustom wwg/monitor-period 15 "How many seconds before checking if a writer has
↪  reached the target number of words. Defaults to a minute." :group 'wwg)

(defvar wwg/monitor-function 'wwg/check-count-and-beep-with-message-if-finished
↪  "The function to monitor the target was reached in buffer. It takes two
↪  arguments: a number (target) and the buffer. It should return any value when
↪  it finds the target satisfied for cleanup purposes.")

(defun wwg/check-count-and-beep-with-message-if-finished (target-count buffer)
  "Beep if TARGET-COUNT was reached in BUFFER."
  (let* ((total-so-far (with-current-buffer buffer (count-words (point-min)
  ↪  (point-max))))
         (remaining-words (- target-count total-so-far)))
    (if (<= remaining-words 0)
        (progn
          (beep)
          (message
```

Emacsclient + Sessions

Workspace + emacsclient

Stops new emacsclient frames from creating new workspaces

```
(after! persp-mode
(setq persp-emacsclient-init-frame-behaviour-override "main"))
```

TODO Initial Buffer

```
;; (setq initial-buffer-choice t)
```

TODO Desktop

```
;; (setq desktop-auto-save-timeout 300)
;; (setq desktop-dirname "~/.emacs.d/.local/etc")
;; (setq desktop-base-file-name "desktop")
;; (setq desktop-load-locked-desktop t)
;; (desktop-save-mode 1)
;; (add-hook 'server-after-make-frame-hook 'desktop-read)
```

Programming

Formatting

```
(setq +format-on-save-enabled-modes
      '(not emacs-lisp-mode sql-mode tex-mode latex-mode julia-mode))
```

TODO LSP

```
                                        ; Rust
;; (setq lsp-rust-server 'rust-analyzer)
(map! :n "SPC t u" #'lsp-ui-doc-mode)

;; (after! rustic
;; (setq rustic-lsp-server 'rust-analyzer))

;; (after! lsp-rust
;;   (setq lsp-rust-server 'rust-analyzer))

;; (setq lsp-disabled-clients '(rls))
```

Julia

https://github.com/julia-vscode/LanguageServer.jl/issues/651

```
;; used when using Pkg.add("LanguageServer")
(setq lsp-julia-package-dir nil)

                                        ; for some reason emacs can't find the
                                        ↪  julia bin in PATH
(setq julia-repl-executable-records
      '((default "~/julia-1.6.0/bin/julia")
        ))

(after! julia-repl
  (julia-repl-set-terminal-backend 'vterm)
  )

(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . t)
   (julia . t)
   (python . t)
   (jupyter . t)))

(setq org-babel-default-header-args:jupyter-julia '((:async . "no")
                                                    (:session . "jl")
                                                    (:kernel . "julia-1.6")))
;; just using a snippet instead
;; (add-to-list 'org-structure-template-alist
;; '("j" . "src jupyter-julia :session jl :results graphics"))

;; required to get lsp to work
;; https://github.com/non-Jedi/lsp-julia/issues/35
(setq lsp-enable-folding t)


(map! :map julia-repl-mode-map "M-RET" 'julia-repl-send-region-or-line)
```

STRT Packages to Look At

STRT Transclusion

https://github.com/nobiot/org-transclusion

STRT Annotate

```
;(annotate-mode)
```

STRT Elgantt

```
;; enable elgantt - https://github.com/legalnonsense/elgantt/
;; (add-to-list 'load-path (concat user-emacs-directory "elgantt/")) ;; Or
↪  wherever it is located
;; (require 'elgantt)
```

HOLD Packages Not Used Right Now

HOLD header-line

```
;; (defun toggle-header-line-format ()
;;     "Toggle buffer-local var header-line-format as pseudo-top margin"
;;     (setq header-line-format (if (eq header-line-format nil) t nil))
;;     (interactive)
;;     (redraw-display))
;; (global-set-key (kbd "<f6>") 'toggle-header-line-format)
; use with set-face-font header-line
;(set-face-background 'header-line "white")
```

HOLD Pandoc

Bind pdf-export in pandoc

Note: Deprecated in favor of Org Export - Latex

```
;(map! :n "SPC r r" #'pandoc-convert-to-pdf)
```

HOLD Projectile

```
; unbind SPC p F
;(map! :map doom-leader-map "p F" nil)
; rebind SPC p F to search all projects' files
;(map! :n "SPC p F" #'projectile-find-file-in-known-projects)
```

HOLD mu4e

```
;; (add-to-list 'load-path "/usr/share/emacs/site-lisp/mu4e")
;; ;; Each path is relative to `+mu4e-mu4e-mail-path', which is ~/.mail by default
;; (set-email-account! "Personal"
;;   '((mu4e-sent-folder       . "/gmail/[Gmail].Sent Mail")
;;     ;(mu4e-drafts-folder     . "/gmail/Drafts")
;;     (mu4e-trash-folder      . "/gmail/[Gmail].Trash")
;;     (mu4e-refile-folder     . "/gmail/[Gmail].All Mail")
;;     (smtpmail-smtp-user     . "jonathanfung2000@gmail.com")
;;     ;; (mu4e-compose-signature . "---\nHenrik Lissner"))
;;   t))
;; (set-email-account! "UCI"
;;   '((mu4e-sent-folder       . "/uci/[Gmail].Sent Mail")
;;     ;(mu4e-drafts-folder     . "/gmail/Drafts")
;;     (mu4e-trash-folder      . "/uci/[Gmail].Trash")
;;     (mu4e-refile-folder     . "/uci/[Gmail].All Mail")
;;     (smtpmail-smtp-user     . "fungjm@uci.edu")
;;     ;; (mu4e-compose-signature . "---\nHenrik Lissner"))
;;   t)
```

HOLD Org-Krita

```
;; (use-package! org-krita
;;   :config
;;   (add-hook 'org-mode-hook 'org-krita-mode))
```