Config

User Info

$DOOMDIR/config.el **-- lexical-binding: t; --**

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 information 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.
```

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 "Jet Brains Mono" :weight 'light :height 100))
(setq doom-font (font-spec :family "Source Code Pro" :height 100))
(setq doom-variable-pitch-font (font-spec :family "Source Sans Pro" :size 30))

; idk what this line does
(setq auto-mode-alist (cons '("\\.org$" . org-mode) auto-mode-alist))

;Ligatures for JetBrains Mono
;; (let ((alist '((?! . "\\(?:!\\(?:==\\|[!=]\\)\\)")
;;                (?# . "\\(?:#\\(?:###?\\|_(\\|[!#(:=?[_{]\\)\\)")
;;                (?$ . "\\(?:\\$>\\)")
;;                (?& . "\\(?:&&&?\\)")
;;                (?* . "\\(?:\\*\\(?:\\*\\*\\|[/>]\\)\\)")
;;                (?+ . "\\(?:\\+\\(?:\\+\\+\\|[+>]\\)\\)")
;;                (?- . "\\(?:-\\(?:-[>-]\\|<<\\|>>\\|[<>|~-]\\)\\)")
;;                (?. . "\\(?:\\.\\(?:\\.[.<]\\|[.=?-]\\)\\)")
;;                (?/ . "\\(?:/\\(?:\\*\\*\\|//\\|==\\|[*/=>]\\)\\)")
;;                (?: . "\\(?::\\(?:::\\|\\?>\\|[:<-?]\\)\\)")
;;                (?\; . "\\(?:;;\\)")
;;                (?< . "\\(?:<\\(?:!--\\|\\$>\\|\\*>\\|\\+>\\|-[<>|]\\|/>\\|<[<=-]\\|=\\(?
;;                (?= . "\\(?:=\\(?:!=\\|/=\\|:=\\|=[=>]\\|>>\\|[=>]\\)\\)")
;;                (?> . "\\(?:>\\(?:=>\\|>[=>-]\\|[]:=-]\\)\\)")
;;                (?? . "\\(?:\\?[.:=?]\\)")
;;                (?\[ . "\\(?:\\[\\(?:||]\\|[<|]\\)\\)")
;;                (?\ . "\\(?:\\\\/?\\)")
;;                (?\] . "\\(?:]#\\)")
;;                (?^ . "\\(?:\\^=\\)")
;;                (?_ . "\\(?:_\\(?:|?_\\)\\)")
;;                (?{ . "\\(?:{|\\)")
;;                (?| . "\\(?:|\\(?:->\\|=>\\||\\(?:|>\\|[=>-]\\)\\|[]=>|}-]\\)\\)")
;;                (?~ . "\\(?:~\\(?:~>\\|[=>@~-]\\)\\)"))))
;;   (dolist (char-regexp alist)
;;     (set-char-table-range composition-function-table (car char-regexp)
;;                           `([,(cdr char-regexp) 0 font-shape-gstring]))))
```

Theme

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.

```
;(setq doom-theme 'doom-dracula)

(load-theme 'modus-operandi t)
(setq modus-operandi-theme-rainbow-headings t)
(setq modus-operandi-theme-section-headings t )
(setq modus-operandi-theme-scale-headings t )
(setq modus-operandi-theme-slanted-constructs t )
(setq modus-operandi-theme-bold-constructs t )

;; (load-theme 'modus-vivendi t)
;; (setq modus-vivendi-theme-rainbow-headings t)
;; (setq modus-vivendi-theme-section-headings t )
(setq modus-vivendi-theme-scale-headings t )
(setq modus-vivendi-theme-slanted-constructs t )
(setq modus-vivendi-theme-bold-constructs t )
```

Toggle Themes

Define Custom functions and map heaven-and-hell to F5

```
(after! heaven-and-hell
  (setq heaven-and-hell-themes
        '((light . modus-operandi)
          (dark . doom-dracula)))
  ;; Optionall, load themes without asking for confirmation.
  (setq heaven-and-hell-load-theme-no-confirm t)
  (map!
   :g "<f5>" 'heaven-and-hell-toggle-theme
   ;; Sometimes loading default theme is broken. I couldn't figured that out yet.
   :leader "<f5>" 'heaven-and-hell-load-default-theme))

(add-hook 'after-init-hook 'heaven-and-hell-init-hook)

(defvar *haba-theme-dark* 'doom-dracula)
(defvar *haba-theme-light* 'modus-operandi)
(defvar *haba-current-theme* *haba-theme-dark*)

;; disable other themes before loading new one
(defadvice load-theme (before theme-dont-propagate activate)
  "Disable theme before loading new one."
  (mapcar #'disable-theme custom-enabled-themes))

(defun haba/next-theme (theme)
  (if (eq theme 'default)
      (disable-theme *haba-current-theme*)
```

```
    (progn
      (load-theme theme t)))
  (setq *haba-current-theme* theme))

(defun haba/toggle-theme ()
  (interactive)
  (cond ((eq *haba-current-theme* *haba-theme-dark*) (haba/next-theme *haba-theme-light*))
        ((eq *haba-current-theme* *haba-theme-light*) (haba/next-theme 'default))
        ((eq *haba-current-theme* 'default) (haba/next-theme *haba-theme-dark*))))
```

Display

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

; 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")
```

Org

```
(setq org-directory "~/org/")
(setq display-line-numbers-type 'relative)

(add-hook 'org-mode-hook 'pandoc-mode)
;(add-hook 'after-save-hook #'pandoc-convert-to-pdf)
```

Org Agenda

```
(setq org-agenda-files '("~/org/Agenda.org"))
(setq org-tag-faces
      '(("Poly" . "gold2") ("Cer" . "lime green") ("Xray" . "red2")
        ("Snr" . "medium orchid") ("Stat_112" . "dodger blue")))

(setq org-agenda-start-day "+0")

(setq org-agenda-custom-commands
       '(("u" "Super view"
          ((agenda "" ((org-super-agenda-groups
                        '((:name "Next Items"
                           :time-grid t
                           :tag ("NEXT" "outbox"))
```

```
                          (:name "School"
                           :tag ("Poly" "Cer" "Xray" "Snr"))
                          (:name "Personal"
                           :tag "Person")
                          )))))
           (alltodo "" ((org-agenda-overriding-header "Projects")
                     (org-super-agenda-groups
                      '((:tag "Person")
                        (:discard (:anything t)))))))))
(setq org-agenda-custom-commands
      '(("z" "Super View"
         ((agenda "" ((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
                          ;;        :order 7)
                          (:name "Personal"
                                 :tag "Person"
                                 :order 10)
                          (:name "Email"
                                 :tag "Email"
                                 :order 15)
                          (:discard (:anything t))))))))))
```

Org Capture

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

Custom Functions

```
; Set Toggle for rot13 cipher
(defun my-rot13-toggle ()
    (interactive)
    (toggle-rot13-mode)
    (redraw-display)
    )

; define custom horizonal-vertical split switch
(defun toggle-window-split ()
  (interactive)
  (if (= (count-windows) 2)
      (let* ((this-win-buffer (window-buffer))
             (next-win-buffer (window-buffer (next-window)))
             (this-win-edges (window-edges (selected-window)))
             (next-win-edges (window-edges (next-window)))
             (this-win-2nd (not (and (<= (car this-win-edges)
                                         (car next-win-edges))
                                     (<= (cadr this-win-edges)
                                         (cadr next-win-edges)))))
             (splitter
              (if (= (car this-win-edges)
                     (car (window-edges (next-window))))
                  'split-window-horizontally
                'split-window-vertically)))
        (delete-other-windows)
        (let ((first-win (selected-window)))
          (funcall splitter)
          (if this-win-2nd (other-window 1))
          (set-window-buffer (selected-window) this-win-buffer)
          (set-window-buffer (next-window) next-win-buffer)
          (select-window first-win)
          (if this-win-2nd (other-window 1))))))
```

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)

; unbind |
(map! :map evil-motion-state-map "|" nil)
; bind | to custom function
(map! :n "|" 'toggle-window-split)

;; Make evil-mode up/down operate in screen lines instead of logical 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)
```

Treemacs

Bind external opening for treemacs

```
(map! :n "SPC o o" #'treemacs-visit-node-in-external-application)
(setq treemacs-position 'right
      treemacs-width 25
      treemacs-indentation 1)
```

Pandoc

Bind pdf-export in pandoc

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

Notmuch

```
; define function that syncs mbsync and refreshes notmuch
(defun 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" 'sync-email)

; attempt to fix notmuch formatting
;; (setq notmuch-search-result-format
;;   (("date" . "%12s ")
;;     ("count" . "%-7s ")
;;     ("authors" . "%-15s ")
;;     ("subject" . "%-20s ")
;;     ("tags" . "(%s)"))
;; )

(setq notmuch-search-result-format '(("authors" . "%-40s")
     ("subject" . "%s")))

(setq notmuch-saved-searches '((:name "inbox" :query "tag:inbox")
                               (:name "unread" :query "tag:inbox AND tag:unread")
                               (:name "uci" :query "tag:inbox AND to:fungjm@uci.edu")))
```

Custom Keybinds

```
;; Bind toggles
(global-set-key (kbd "<f2>") 'mixed-pitch-mode)
(global-set-key (kbd "<f3>") 'olivetti-mode)
(setq olivetti-body-width 90)
(global-set-key (kbd "<f4>") 'my-rot13-toggle)
;; (global-set-key (kbd "U") 'undo-tree-redo)

; 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)
```

Elgantt

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