

# Config

Aseem Ratha

April 21, 2025

## Contents

<b>1</b>	<b>Header</b>	<b>2</b>
<b>2</b>	<b>Auto-Tangle</b>	<b>3</b>
<b>3</b>	<b>Appearance</b>	<b>3</b>
3.1	Theme . . . . .	3
3.2	Dashboard . . . . .	3
3.3	Initial Dimensions . . . . .	4
3.4	Fonts . . . . .	5
<b>4</b>	<b>Org Settings</b>	<b>5</b>
<b>5</b>	<b>Editor Settings</b>	<b>8</b>
5.1	General . . . . .	8
5.2	Copy-Paste . . . . .	9
5.3	Web-Dev . . . . .	10
5.4	Supercollider . . . . .	11
5.5	Odin . . . . .	11
5.6	Latex . . . . .	12
5.7	Python . . . . .	12
5.8	Copilot . . . . .	12
5.9	Clang . . . . .	13
5.10	Rust . . . . .	13
5.11	Dev Environment . . . . .	13
5.12	Telephone-line . . . . .	14
5.13	Indent-Guides . . . . .	15
5.14	Flyspell . . . . .	15
5.15	Leetcode . . . . .	15

5.16 PDF-Tools . . . . .	15
5.17 Vterm . . . . .	15
5.18 gptel . . . . .	15
5.19 Zoom . . . . .	15
<b>6 Keybinds</b>	<b>16</b>

## 1 Header

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

;; Place your private configuration here! Remember, you do not need to run 'doom
;; sync' after modifying this file!

;; Some functionality uses this to identify you, e.g. GPG configuration, email
;; clients, file templates and snippets. It is optional.
;; (setq user-full-name "John Doe"
;;       user-mail-address "john@doe.com")

;; Doom exposes five (optional) variables for controlling fonts in Doom:
;;
;; - 'doom-font' -- the primary font to use
;; - 'doom-variable-pitch-font' -- a non-monospace font (where applicable)
;; - 'doom-big-font' -- used for 'doom-big-font-mode'; use this for
;;   presentations or streaming.
;; - 'doom-symbol-font' -- for symbols
;; - 'doom-serif-font' -- for the 'fixed-pitch-serif' face
;;
;; See 'C-h v doom-font' for documentation and more examples of what they
;; accept. For example:
;;
;;(setq doom-font (font-spec :family "Fira Code" :size 12 :weight 'semi-light)
;;      doom-variable-pitch-font (font-spec :family "Fira Sans" :size 13))
;;
;; If you or Emacs can't find your font, use 'M-x describe-font' to look them
;; up, 'M-x eval-region' to execute elisp code, and 'M-x doom/reload-font' to
;; refresh your font settings. If Emacs still can't find your font, it likely
;; wasn't installed correctly. Font issues are rarely Doom issues!
```

## 2 Auto-Tangle

```
(add-hook 'org-mode-hook
  (lambda ()
    (when (string-equal (file-name-nondirectory buffer-file-name) "config.org")
      (add-hook 'after-save-hook 'org-babel-tangle nil 'make-it-local))))
```

## 3 Appearance

### 3.1 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. This is the default:
(add-to-list 'custom-theme-load-path (expand-file-name "themes/" doom-user-dir))
;; Set theme based on display type
(if (display-graphic-p)
  (setq doom-theme 'doom-solarized-transparent      )
  (setq doom-theme 'doom-solarized-transparent-term))
  ;; rogue, gruvbox, opera, sourcerer

(set-frame-parameter nil 'alpha-background 0.4) ; For current frame
```

### 3.2 Dashboard

```
(setopt initial-buffer-choice #'enlight)

(require 'grid)

(defvar enlight-lipsum "Test
Test")

(defface enlight-yellow-bold
  '((t (:foreground "#cabf00" :bold t)))
  "Yellow bold face")
```

```

(defvar enlight-calendar
  (progn
    (calendar)
    (prog1 (with-current-buffer (buffer-name (current-buffer))
      (buffer-string))
      (calendar-exit)))))

(use-package enlight
  :custom
  (enlight-content
    (concat
      (grid-get-row
        (list
          (grid-get-box
            (concat
              (grid-get-box
                `(
                  (:content
                    ,(concat
                      (grid-get-box `(
                        (:content ,(propertize "Doom Emacs" 'face 'bold)
                          :width 80 :align center)))
                    )
                  )
                  :width 80)
                )
                enlight-calendar "\n"
                (grid-get-row
                  `(
                    ,(concat
                      (propertize " " 'face 'bold)
                      "\n"
                      (enlight-menu
                        '("Org Mode"
                          ("Org-Agenda (current day)" (org-agenda nil "a") "a"))
                        ("Downloads"
                          ("Downloads folder" (dired "~/Downloads") "a"))
                        ("Other"
                          ("Projects" project-switch-project "p"))))))
                  )
                )
              )
            )
          )
        )
      )
    )
  )
)

```

### 3.3 Initial Dimensions

```

;; Set the initial window width and height (in columns and rows)
(setq initial-frame-alist

```

```
((width . 100) ; Width in characters  
 (height . 50))) ; Height in lines
```

### 3.4 Fonts

```
(setq doom-font (font-spec :family "JetBrainsMono Nerd Font" :size 14.0 :height 1)  
      ;;doom-variable-pitch-font (font-spec :family "Averia Serif Libre" :size 14.0)  
      ;;doom-mixed-pitch-font (font-spec :family "Averia Serif Libre" :size 30.0))  
      doom-variable-pitch-font (font-spec :family "EB Garamond" :size 20.0)  
      doom-mixed-pitch-font (font-spec :family "EB Garamond" :size 20.0))  
  
(use-package! mixed-pitch  
  :config  
  (setq mixed-pitch-set-height t)  
  (setq variable-pitch-serif-font doom-variable-pitch-font)  
  (set-face-attribute 'variable-pitch nil :height 0.9))  
  
(setq mixed-pitch-variable-pitch-cursor t)
```

## 4 Org Settings

```
;; If you use 'org' and don't want your org files in the default location below,  
;; change 'org-directory'. It must be set before org loads!  
(setq org-directory "/Users/aaratha/Library/CloudStorage/OneDrive-Personal/org/")  
  
(setq org-appear-mode 1)  
  
(setq org-hide-emphasis-markers t)  
  
;  
;; Automatically enable olivetti-mode when org-agenda starts  
(add-hook 'org-agenda-mode-hook #'olivetti-mode)  
  
(setq org-agenda-span 'month)  
  
;; (use-package writeroom-mode
```

```

;;    :config
;;    (setq writeroom-fullscreen-effect nil)
;;    (setq writeroom-width 64)
;;    )

;; (add-hook 'writeroom-mode-enable-hook (lambda () (text-scale-set 2)))
;; (add-hook 'writeroom-mode-disable-hook (lambda () (text-scale-set 0)))

(setq doc-view-continuous t)

(after! org
  ;; This is usually the default, but keep in mind it must be nil

  ;;; Titles and Sections
  ;; hide #+TITLE:
  ;;(setq org-hidden-keywords '(title))
  ;; set basic title font
  (set-face-attribute 'org-level-8 nil :weight 'bold :inherit 'default)
  ;; Low levels are unimportant => no scaling
  (set-face-attribute 'org-level-7 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-6 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-5 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-4 nil :inherit 'org-level-8)
  ;; Top ones get scaled the same as in LaTeX (\large, \Large, \LARGE)
  (set-face-attribute 'org-level-3 nil :inherit 'org-level-8 :height 1.2) ;\large
  (set-face-attribute 'org-level-2 nil :inherit 'org-level-8 :height 1.3) ;\Large
  (set-face-attribute 'org-level-1 nil :inherit 'org-level-8 :height 1.4) ;\LARGE
  ;; Only use the first 4 styles and do not cycle.
  (setq org-cycle-level-faces nil)
  (setq org-n-level-faces 4)
  ;; Document Title, (\huge)
  (set-face-attribute 'org-document-title nil
                     :height 1.5
                     ;; :foreground 'unspecified
                     :inherit 'org-level-8)
  (remove-hook 'org-mode-hook #'flyspell-mode)
  (remove-hook 'org-mode-hook #'org-indent-mode)
  (remove-hook 'org-mode-hook #'org-superstar-mode)

```

```

)

(setq org-startup-indented nil)
(after! org
  (remove-hook 'org-mode-hook #'org-indent-mode))

;; Basic Setup
;; Auto-start Superstar with Org
(add-hook 'org-mode-hook
  (lambda ()
    (org-superstar-mode 1)))
(add-hook 'org-mode-hook 'adaptive-wrap-prefix-mode)
(add-hook 'org-mode-hook (lambda () (display-line-numbers-mode -1)))

;;(setq org-superstar-leading-bullet ?\s)
;; If you use Org Indent you also need to add this, otherwise the
;; above has no effect while Indent is enabled.
(setq org-hide-leading-stars nil)
;; (setq org-startup-indented t)
;; (setq org-adapt-indentation t)
(setq org-indent-mode-turns-on-hiding-stars nil)
;; (setq org-superstar-remove-leading-stars t)

;; turn off org-indent-mode
(setq org-startup-indented nil)
(remove-hook 'org-mode-hook #'org-indent-mode)

(with-eval-after-load 'org-superstar
  (set-face-attribute 'org-superstar-item nil :height 0.9))
;; Set different bullets, with one getting a terminal fallback.
(setq org-superstar-headline-bullets-list
  '(" " " " "))

(setq org-superstar-item-bullet-alist '((?* . ?) (?+ . ?) (?- . ?)))
;; Stop cycling bullets to emphasize hierarchy of headlines.
;;(setq org-superstar-cycle-headline-bullets nil)

;; ; Configure Superstar only for terminal
;; (unless (display-graphic-p)

```

```

;; (use-package! org-superstar
;;   :hook (org-mode . org-superstar-mode)
;;   :config
;;   (setq org-superstar-leading-fallback " "
;;         org-superstar-item-bullet-alist '(((* . ?) (+ . ?) (- . ?)))))

;; make latex previews have a transparent background
;; (plist-put org-format-latex-options :background "Transparent")

;; citations
;; :custom
;; (org-cite-csl-styles-dir
;;  (expand-file-name "~/Zotero/styles/"))

```

## 5 Editor Settings

### 5.1 General

```

(pixel-scroll-precision-mode 1)

(map! :leader
      :desc "Treemacs"
      "e" #'treemacs)

(setq treemacs-width 20)

(setq window-divider-default-right-width 3)

(require 'mouse)
(xterm-mouse-mode t)
(global-set-key [mouse-4] (lambda ()
                           (interactive)
                           (scroll-down 1)))
(global-set-key [mouse-5] (lambda ()
                           (interactive)
                           (scroll-up 1)))
(setq mouse-sel-mode t)

```

```

(setq magit-define-global-key-bindings nil)

(use-package olivetti
  :custom
  (olivetti-body-width 64))

(map! :leader ; Modifies the leader key
      :desc "Lsp Format Buffer" ; Optional: description for the command
      "l" #'lsp-format-buffer) ; Example: bind "SPC k e" to 'eval-buffer'

(vertico-mouse-mode 1)

(setq text-scale-mode-step 1)
(setq doom-font-increment 1)

(defun my-change-window-divider ()
  (let ((display-table (or buffer-display-table
                           standard-display-table
                           (make-display-table))))
    (set-display-table-slot display-table 5 ?)
    (set-window-display-table (selected-window) display-table)))

(when (not (display-graphic-p))
  ;; Apply window divider settings only in terminal Emacs
  (add-hook 'window-configuration-change-hook 'my-change-window-divider))

(window-divider-mode -1)

```

## 5.2 Copy-Paste

```

;; (setq shell-file-name "/bin/fish")

;; (setq select-enable-clipboard t)
;; (setq x-select-enable-clipboard-manager nil)

;; (setq wl-copy-process nil)

;; (defun my/copy-to-clipboard (text &optional push)
;;   (setq wl-copy-process (make-process :name "wl-copy"

```

```

;; :buffer nil
;; :command '("wl-copy" "-f" "-n")))
;; (process-send-string wl-copy-process text)
;; (process-send-eof wl-copy-process))

;; (defun my/paste-from-clipboard ()
;;   (if (and wl-copy-process (process-live-p wl-copy-process))
;;       nil
;;       (shell-command-to-string "wl-paste -n")))

;; (setq interprogram-cut-function 'my/copy-to-clipboard)
;; (setq interprogram-paste-function 'my/paste-from-clipboard)

```

### 5.3 Web-Dev

```

;; (package! lsp-tailwindcss :recipe (:host github :repo "merrickluo/lsp-tailwindcss"))

(use-package! lsp-mode
  :init
  ;; Exclude Vue language server for tsx and ts files
  (setq lsp-disabled-clients '(vue-semantic-server))
  :hook
  ((typescript-mode . lsp-deferred)
   (web-mode . lsp-deferred))
  :config
  ;; Configure TypeScript language server
  (lsp-register-client
    (make-lsp-client :new-connection (lsp-stdio-connection
                                         (lambda () '("typescript-language-server" "--stdio"
                                                       :major-modes '(typescript-mode web-mode)
                                                       :priority 1
                                                       :server-id 'typescript-ls)))
    :after! web-mode
    (add-to-list 'auto-mode-alist '("\\.tsx\\'" . web-mode))
    (add-to-list 'auto-mode-alist '("\\.ts\\'" . typescript-mode)))

```

```

(use-package! vue-mode
  :mode "\\.vue\\\""
  :hook (vue-mode . prettier-js-mode)
  :config
  (add-hook 'vue-mode-hook #'lsp)
  (setq prettier-js-args '("--parser vue")))

;; Additional optional configurations for LSP formatting
(use-package! lsp-mode
  :custom
  (lsp-vetur-format-default-formatter-css "none")
  (lsp-vetur-format-default-formatter-html "none")
  (lsp-vetur-format-default-formatter-js "none")
  (lsp-vetur-validation-template nil))

```

## 5.4 Supercollider

```

;; Super Collider Configs
;; (setq exec-path (append exec-path '("/Applications/SuperCollider.app/Contents/MacOS"))

;; (add-to-list 'load-path "/Users/aaratha/Library/Application Support/SuperCollider/da
;; (require 'sclang)

;; (setenv "PATH" (concat (getenv "PATH") ":/Users/aaratha/.ghcup/bin"))
;; (setq exec-path (append exec-path '("/Users/aaratha/.ghcup/bin")))

;; (setq haskell-process-type 'ghci)
;; (setq haskell-process-ghci "/Users/aaratha/.ghcup/bin/ghci")

```

## 5.5 Odin

```

;; Odin Configs
;; (use-package! odin-mode
;;   :mode ("\\.odin\\\" . odin-mode)
;;   :config
;;   ;; Add any additional configuration here
;;   )

;; (setenv "PATH" (concat (getenv "PATH") ":/Users/aaratha/ols"))

```

```

;; (setq exec-path (append exec-path '("/Users/aaratha/ols")))

;; (use-package! lsp-mode
;;   :commands (lsp lsp-deferred)
;;   :hook (odin-mode . lsp) ;; Start lsp when odin-mode is active
;;   :config
;;   (add-to-list 'lsp-language-id-configuration '(odin-mode . "odin"))
;;   (lsp-register-client
;;     (make-lsp-client
;;       :new-connection (lsp-stdio-connection "/Users/aaratha/ols/ols")
;;       :major-modes '(odin-mode)
;;       :server-id 'odin-ls)))

```

## 5.6 Latex

```

(setenv "PATH" (concat (getenv "PATH") ":/usr/local/texlive/2024/bin/universal-darwin")
(setq exec-path (append exec-path '("/usr/local/texlive/2024/bin/universal-darwin")))

(setq bibtex-dialect 'biblatex)

```

## 5.7 Python

```
(setq python-python-command "/Users/aaratha/.pyenv/shims/python")
```

## 5.8 Copilot

```

;; accept completion from copilot and fallback to company
(use-package! copilot
  :hook (prog-mode . copilot-mode)
  :bind (:map copilot-completion-map
             ("M-<tab>" . 'copilot-accept-completion)
             ("M-TAB" . 'copilot-accept-completion)
             ("C-TAB" . 'copilot-accept-completion-by-word)
             ("C-<tab>" . 'copilot-accept-completion-by-word)))

(after! (evil copilot)
  ;; Define the custom function that either accepts the completion or does the default
  (defun my/copilot-tab-or-default ()
    (interactive)
    (if (and (bound-and-true-p copilot-mode)
              ;; Add any other conditions to check for active copilot suggestions if needed
              )
        (my/copilot-tab-or-default)
        (default))))
```

```

        )
(copilot-accept-completion)
(evil-insert 1))) ; Default action to insert a tab. Adjust as needed.

;; Bind the custom function to <tab> in Evil's insert state
(evil-define-key 'insert 'global (kbd "<tab>") 'my/copilot-tab-or-default))

(add-hook 'emacs-lisp-mode-hook (lambda () (setq-local copilot-indent-offset 2)))

```

## 5.9 Clang

```

(setq clang-format-executable "/opt/homebrew/opt/llvm/bin/clang-format")

(setq-hook! 'cpp-mode-hook +format-with "clang-format")
(setq-hook! 'glsl-mode-hook +format-with "clang-format")

(add-to-list 'auto-mode-alist '("\\.vs\\\\" . glsl-mode))
(add-to-list 'auto-mode-alist '("\\.fs\\\\" . glsl-mode))

```

## 5.10 Rust

```

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

(add-to-list 'exec-path " ~/.cargo/bin")

```

## 5.11 Dev Environment

```

(defun my/setup-dev-environment ()
  "Set up development environment with three vertical splits, treemacs, and vterm."
  (interactive)
  ;; Delete other windows to start fresh
  (delete-other-windows)

  ;; Create two vertical splits first
  (split-window-right)
  (split-window-right)

  ;; Move to the rightmost window
  (other-window 2))

```

```

;; Start vterm in the rightmost window
(+vterm/here nil)

;; Adjust the width of the terminal window
(let ((vterm-width 30)) ;; Adjust the desired width here
  (enlarge-window-horizontally (- vterm-width (window-width)))))

;; Move to the leftmost window to set up treemacs
(other-window -2)

;; Open treemacs
(treemacs)

;; Move to the coding window (middle window)
(other-window 1)

;; Adjust all window widths
(balance-windows-area)

;; Bind the command to a key (optional)
(map! :leader
      :desc "Setup dev environment"
      "d e" #'my/setup-dev-environment)

```

## 5.12 Telephone-line

```

(require 'telephone-line)
(set-face-attribute 'telephone-line-accent-active
                    nil
                    :background
                    "#2a4b57")
(set-face-attribute 'telephone-line-evil-normal
                    nil
                    :background
                    "#21523c")
(set-face-attribute 'telephone-line-evil-insert
                    nil
                    :background
                    "#752d2d")

```

```
(telephone-line-mode t)
```

### 5.13 Indent-Guides

```
(custom-set-faces
  '(highlight-indent-guides-character-face ((t (:foreground "#335057"))))
  '(highlight-indent-guides-top-character-face ((t (:foreground "#6898a3"))))
  '(highlight-indent-guides-stack-character-face ((t (:foreground "#4d7882")))))
```

### 5.14 Flyspell

```
(setq ispell-program-name "/opt/homebrew/bin/aspell")
(flyspell-mode-off)
```

### 5.15 Leetcode

```
(setq leetcode-prefer-language "cpp")
```

### 5.16 PDF-Tools

```
;; fix blurry pdfs on mac
(setq pdf-view-use-scaling t)
```

### 5.17 Vterm

```
(with-eval-after-load 'evil
  (evil-set-initial-state 'vterm-mode 'insert))
```

### 5.18 gptel

```
(setq gptel-api-key "sk-proj-10Vha6XWKnjQdHW2JNhIT3BlbkFJdKJ8X6BZysd5m1Jtjicw")
```

### 5.19 Zoom

```
(custom-set-variables
  '(zoom-size '(0.618 . 0.618)))
```

## 6 Keybinds

```
(map! :leader
      :desc "Olivetti mode"
      "t o" #'olivetti-mode)

(defun my-visual-to-eol ()
  "Start visual mode and select to the end of the line."
  (interactive)
  (evil-visual-char)
  (end-of-visual-line)
  (when (and (eolp) (not (bolp))) ; Check if at the end of a non-empty line
    (evil-backward-char)))

(map! :leader
      :desc "Visual to end of line"
      "v" #'my-visual-to-eol)

(map! :leader
      :desc "View Org Agenda"
      "SPC" #'org-agenda-list)

;; run build.sh
(defun run-build-script ()
  (interactive)
  (vterm)
  (vterm-send-string "./build.sh\n"))

(map! :leader
      :desc "Run build script in vterm"
      "r" #'run-build-script)

(defun my-zen-mode ()
  "Toggle my zen mode."
  (interactive)
  (if (or olivetti-mode mixed-pitch-mode)
      (progn
        (olivetti-mode -1)
        (mixed-pitch-mode -1))
      (olivetti-mode 1))
```

```
(mixed-pitch-mode 1)))  
  
(map! :leader  
      :desc "My zen mode"  
      "t z" #'my-zen-mode)  
  
(map! :leader  
      :desc "Toggle Ace Window"  
      "W" #'ace-window)
```