Emacs Config

Jonathan

May 26, 2025

Contents

1 Doom Setup

```
;;; $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!

;; 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:
(setq doom-theme ’doom-one)

;; This determines the style of line numbers in effect. If set to ‘nil’, line
;; numbers are disabled. For relative line numbers, set this to ‘relative’.
(setq display-line-numbers-type t)

;; 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!

;; Whenever you reconfigure a package, make sure to wrap your config in an
;; ‘after!’ block, otherwise Doom’s defaults may override your settings. E.g.
;;
;;   (after! PACKAGE
;;     (setq x y))
;;
```

```
;; The exceptions to this rule:
;;
;;   - Setting file/directory variables (like ‘org-directory’)
;;   - Setting variables which explicitly tell you to set them before their
;;     package is loaded (see 'C-h v VARIABLE' to look up their documentation).
;;   - Setting doom variables (which start with 'doom-' or '+').
;;
;; Here are some additional functions/macros that will 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 c k').
;; This will open documentation for it, including demos of how they are used.
;; Alternatively, use ‘C-h o’ to look up a symbol (functions, variables, faces,
;; etc).
;;
;; You can also try 'gd' (or 'C-c c d') to jump to their definition and see how
;; they are implemented.
```

2 Org Mode

2.1 Org files

```
(setq org-directory "~/OneDrive/org/")
(setq org-agenda-files '("~/OneDrive/org/tasks.org"
    "~/OneDrive/org/shed.org"
    "~/OneDrive/org/journal.org"
                         "~/OneDrive/org/anniversaries.org"))
```

2.2 Capture

```
(defun my/read-quotes ()
  "Read quotes from ~/.emacs.d/quotes/quotes and return them as a list of tuples.
  Each tuple contains (QUOTE AUTHOR SOURCE).
  Skips the first quote block (which is used as a template)."
  (let ((quotes-file "~/.config/doom/quotes")
        (quotes-list nil))
    (when (file-exists-p quotes-file)
      (with-temp-buffer
        (insert-file-contents quotes-file)
        (let ((quote-blocks (split-string (buffer-string) "\n\n" t)))
          ;; Skip the first block (template) and process the rest
          (dolist (block (cdr quote-blocks))
            (let* ((lines (split-string block "\n" t))
                   ;; Extract the three components (quote, author, source)
                   (quote-text (string-trim (or (nth 0 lines) "")))
                   (author (string-trim (or (nth 1 lines) "")))
                   (source (string-trim (or (nth 2 lines) ""))))
              ;; Only add complete quotes to the list
              (when (and (not (string-empty-p quote-text))
                         (not (string-empty-p author)))
                (push (list quote-text author source) quotes-list)))))))
    (nreverse quotes-list))) ; Return the list in the original order

(defun my/random-quote ()
  "Return a random quote as a tuple (QUOTE AUTHOR SOURCE)."
  (let ((quotes (my/read-quotes)))
    (if (null quotes)
        nil  ; Return nil if no quotes are found
      (nth (random (length quotes)) quotes))))

(defun my/get-quote-string ()
  "Return a random quote formatted as an org-mode quote block for use in templates."
  (let* ((quote-tuple (my/random-quote))
         (quote-text (nth 0 quote-tuple))
         (author (nth 1 quote-tuple))
         (source (nth 2 quote-tuple))
         (attribution (if (string-empty-p source)
                          (format "    --- %s" author)
                        (format "    --- %s, %s" author source))))
    (if quote-tuple
        (format "#+BEGIN_QUOTE\n%s\n%s\n#+END_QUOTE"
                quote-text attribution)
        "No quotes found")))

  ;; (my/get-quote-string)

(after! org
  (setq org-capture-templates
        '(("x" "Export D&D Session")
          ("xd" "Export Dungeon" plain
           (file+olp "dnd-session.org" "Random Dungeons")
           "** %f%?\n#+INCLUDE: ./roam/%f"
           :immediate-finish t
           :jump-to-captured t)

          ("j" "Journal")
          ("jj" "Journal" entry
           (file+olp+datetree "journal.org" "Journal")
           "* Entry - %<%H:%M> %U\n%?"
           :empty-lines 1
           :kill-buffer t)
          ("jm" "Morning" plain
           (file+olp+datetree "journal.org" "Journal")
           "%(my/get-quote-string)"
           :prepend t
           :empty-lines 1
           :immediate-finish t
           :jump-to-captured t
           )

        ("b" "blog-post" entry (file+olp "~/repos/blog-home/blog.org" "blog")
         "* TODO %^{Title} %^g \n:PROPERTIES:\n:EXPORT_FILE_NAME: %^{Slug}\n:EXPORT_DAT
         :empty-lines-before 2)

        ("m" "Email Workflow")
           ("mf" "Follow Up" entry (file+olp "~/OneDrive/org/mail.org" "Follow Up")
            "* TODO Follow up with %:fromname on %a\nSCHEDULED:%t\n\n%i")
           ("mr" "Read Later" entry (file+olp "~/OneDrive/org/mail.org" "Read Later")
            "* TODO Read %a\nSCHEDULED:%t\n\n%i")

           ("s" "Sleep Entry" table-line
            (file+headline "sleep.org" "Data")
            "| |%^{Date}u|%^{Move (kcal)}|%^{Exercise (min)}|%^{Caffeine (mg)}|%^{Tim
            :immediate-finish t :jump-to-captured t)

           ("t" "Task" entry
            (file+headline "tasks.org" "Tasks")
            "** TODO %? %^g\n:PROPERTIES:\n:CREATED: %U\n:END:\n" :empty-lines 1)

           ("T" "Task with Deadline" entry
            (file+headline "tasks.org" "Tasks")
            "** TODO %?  %^g\nDEADLINE: %^t\n:PROPERTIES:\n:CREATED: %U\n:END:\n" :empt

(setq default-doom-org-capture-templates
      '(("t" "Personal todo" entry (file+headline +org-capture-todo-file "Inbox")
         "* [ ] %?\n%i\n%a" :prepend t)
        ("n" "Personal notes" entry (file+headline +org-capture-notes-file "Inbox")
         "* %u %?\n%i\n%a" :prepend t)
        ("j" "Journal" entry (file+olp+datetree +org-capture-journal-file)
         "* %U %?\n%i\n%a" :prepend t)
        ("p" "Templates for projects")
        ("pt" "Project-local todo" entry
         (file+headline +org-capture-project-todo-file "Inbox") "* TODO %?\n%i\n%a"
         :prepend t)
        ("pn" "Project-local notes" entry
         (file+headline +org-capture-project-notes-file "Inbox") "* %U %?\n%i\n%a"
         :prepend t)
        ("pc" "Project-local changelog" entry
         (file+headline +org-capture-project-changelog-file "Unreleased")
         "* %U %?\n%i\n%a" :prepend t)
        ("o" "Centralized templates for projects")
        ("ot" "Project todo" entry #'+org-capture-central-project-todo-file
         "* TODO %?\n %i\n %a" :heading "Tasks" :prepend nil)
        ("on" "Project notes" entry #'+org-capture-central-project-notes-file
         "* %U %?\n %i\n %a" :heading "Notes" :prepend t)
        ("oc" "Project changelog" entry #'+org-capture-central-project-changelog-file
         "* %U %?\n %i\n %a" :heading "Changelog" :prepend t)))
```

2.3 Org Publish

2.3.1 Projects

```
  (setq dnd-org-dir "~/OneDrive/org/roam/dnd/")
```

```
(setq dnd-out-dir "~/Documents/dnd-session")
(setq org-publish-project-alist
      `(
   ("dnd-campaign"
         :base-directory ,dnd-org-dir
         :base-extension "org"
         :publishing-directory ,dnd-out-dir
         :recursive t
         :publishing-function org-html-publish-to-html
         :html-doctype "html5"
         :html-html5-fancy t
         :with-toc t
         :section-numbers nil
         :html-head "<link rel=\"stylesheet\" href=\"./dnd-theme.css\" type=\"text/cs
                     <link href=\"https://fonts.googleapis.com/css2?family=Cinzel:wgh
                           rel=\"stylesheet\">
       <link href=\"https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/al
            rel=\"stylesheet\">"
         :html-preamble "<div class='campaign-header'>Home Brew Campaign</div>
                        <div class='nav-menu'>
                          <a href='index.html'>Home</a> |
                          <a href='random_locations_and_encounters.html'>Random</a>
                        </div>"
         :html-postamble "<div class='footer'>Campaign notes prepared by %a</div>"
         :auto-sitemap t
         :sitemap-title "D&D Campaign Index"
         :sitemap-filename "index.org"
         :sitemap-sort-files anti-chronologically
         )

        ("dnd-static"
         :base-directory ,dnd-org-dir
         :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|map"
         :publishing-directory ,dnd-out-dir
         :recursive t
         :publishing-function org-publish-attachment
         )

        ("dnd-website" :components ("dnd-campaign" "dnd-static"))
  ))
```

3 Org Roam

3.1 Enable org-roam

```
(use-package org-roam
  :ensure t
  :custom
  (org-roam-directory "~/OneDrive/org/roam")
  (org-roam-completion-everywhere t)
  :config
  (org-roam-setup))
```

3.2 Capture Templates

```
(setq org-roam-capture-templates
      '(("d" "D&D")
 ("dn" "new" plain
         "#+FILETAGS: %^{Tags}g\n\n* ${title}\n%?"
         :target (file+head "dnd/${slug}.org" "#+TITLE: ${title}\n")
  :immediate-finish t :jump-to-captured t
         :unnarrowed t)

        ("n" "note" plain
         "* Notes\n%?"
         :target (file+head "%<%Y%m%d%H%M%S>-${slug}.org"
                            "#+title: ${title}\n")
         :unnarrowed t)))
```

4 Org LaTeX

```
(after! ox-latex
(add-to-list
 'org-latex-classes
 '("dndbook"
   "
  \\documentclass[10pt,twoside,twocolumn,openany,print,justified]{dndbook}
  \\usepackage[english]{babel}
  \\usepackage[utf8]{inputenc}
```

```
      "
   ("\\chapter{%s}" . "\\chapter*{%s}")
   ("\\section{%s}" . "\\section*{%s}")
   ("\\subsection{%s}" . "\\subsection*{%s}")
   ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
   ))
(add-to-list
 'org-latex-classes
 '("rpg-module"
   "\\RequirePackage{pgfmath}
      \\documentclass[a4paper,acdesc]{rpg-module}."
   ("\\part{%s}" . "\\part{%s}")
   ("\\section{%s}" . "\\section*{%s}")
   ("\\subsection{%s}" . "\\subsection*{%s}")
   ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
   ))
(add-to-list
 'org-latex-classes
 '("koma-article"
   "\\documentclass{scrartcl}"
   ("\\section{%s}" . "\\section*{%s}")
   ("\\subsection{%s}" . "\\subsection*{%s}")
   ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
   ("\\paragraph{%s}" . "\\paragraph*{%s}")
   ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))
 ))
```

5 Custom

5.1 org include generator

```
  (add-to-list 'load-path (expand-file-name "~/.config/doom"))
  (require 'org-include-generator)

(map! :leader (:prefix-map ("d" . "D&D")
                            :desc  "included" "i" #'org-include-generate-from-current))
```