YSB's Doom Emacs Config

Derek Taylor (DT) and Yuvraj Birdi (YSB)

March 25, 2023

Contents

1 TABLE OF CONTENTS TOC

-
-
- –
  –
  –
  –
-
-
-
- –

–

- –
  –
  –
  –
-
-
-
-
-
-
-
-
-
-
- –
  –
-
-
-
-
-
-
-
-
- –

- –
- –
- –
- –

2 ABOUT THIS CONFIG

This is my personal Doom Emacs config. Doom Emacs is a distribution of Emacs that uses the "evil" keybindings (Vim keybindings) and includes a number of nice extensions and a bit of configuration out of the box. I am maintaining this config not just for myself, but also for those that want to explore some of what is possible with Emacs. I will add a lot of examples of plugins and settings, some of them I may not even use personally. I do this because many people following me on YouTube look at my configs as "documentation".

3 BEACON

Never lose your cursor. When you scroll, your cursor will shine! This is a global minor-mode. Turn it on everywhere with:

```
(beacon-mode 1)
```

4 BOOKMARKS AND BUFFERS

Doom Emacs uses 'SPC b' for keybindings related to bookmarks and buffers.

4.1 Bookmarks

Bookmarks are somewhat like registers in that they record positions you can jump to. Unlike registers, they have long names, and they persist automatically from one Emacs session to the next. The prototypical use of bookmarks is to record where you were reading in various files.

```
(map! :leader
      (:prefix ("b". "buffer")
       :desc "List bookmarks" "L" #'list-bookmarks
       :desc "Save current bookmarks to bookmark file" "w" #'bookmark-save))
```

4.2 Buffers

Regarding *buffers*, the text you are editing in Emacs resides in an object called a *buffer*. Each time you visit a file, a buffer is used to hold the file's text. Each time you invoke Dired, a buffer is used to hold the directory listing. *Ibuffer* is a program that lists all of your Emacs *buffers*, allowing you to navigate between them and filter them.

COMMAND	DESCRIPTION	KEYBINDING
ibuffer	Launch ibuffer	SPC b i
kill-buffer	Kill current buffer	SPC b k
next-buffer	Goto next buffer	SPC b n
previous-buffer	Goto previous buffer	SPC b p
save-buffer	Save current buffer	SPC b s

4.3 Global Auto Revert

A buffer can get out of sync with respect to its visited file on disk if that file is changed by another program. To keep it up to date, you can enable Auto Revert mode by typing M-x auto-revert-mode, or you can set it to be turned on globally with 'global-auto-revert-mode'. I have also turned on Global Auto Revert on non-file buffers, which is especially useful for 'dired' buffers.

```
(global-auto-revert-mode 1)
(setq global-auto-revert-non-file-buffers t)
```

4.4 Keybindings within ibuffer mode

COMMAND	DESCRIPTION	KEYBINDING
ibuffer-mark-forward	Mark the buffer	m
ibuffer-unmark-forward	Unmark the buffer	u
ibuffer-do-kill-on-deletion-marks	Kill the marked buffers	x
ibuffer-filter-by-content	Ibuffer filter by content	f c
ibuffer-filter-by-directory	Ibuffer filter by directory	f d
ibuffer-filter-by-filename	Ibuffer filter by filename (full path)	f f
ibuffer-filter-by-mode	Ibuffer filter by mode	f m
ibuffer-filter-by-name	Ibuffer filter by name	f n
ibuffer-filter-disable	Disable ibuffer filter	f x
ibuffer-do-kill-lines	Hide marked buffers	g h
ibuffer-update	Restore hidden buffers	g H

```
(evil-define-key 'normal ibuffer-mode-map
  (kbd "f c") 'ibuffer-filter-by-content
  (kbd "f d") 'ibuffer-filter-by-directory
  (kbd "f f") 'ibuffer-filter-by-filename
  (kbd "f m") 'ibuffer-filter-by-mode
  (kbd "f n") 'ibuffer-filter-by-name
  (kbd "f x") 'ibuffer-filter-disable
  (kbd "g h") 'ibuffer-do-kill-lines
  (kbd "g H") 'ibuffer-update)
```

5 CALENDAR

Let's make a 12-month calendar available so we can have a calendar app that, when we click on time/date in xmobar, we get a nice 12-month calendar to view.

This is a modification of: `http://homepage3.nifty.com/oatu/emacs/calendar.html` See also: `https://stackoverflow.com/questions/9547912/emacs-calendar-show-more-than-3-months`

```
;; https://stackoverflow.com/questions/9547912/emacs-calendar-show-more-than-3-months
(defun dt/year-calendar (&optional year)
  (interactive)
  (require 'calendar)
  (let* (
      (current-year (number-to-string (nth 5 (decode-time (current-time)))))
```

```
      (month 0)
      (year (if year year (string-to-number (format-time-string "%Y" (current-time))))
    (switch-to-buffer (get-buffer-create calendar-buffer))
    (when (not (eq major-mode 'calendar-mode))
      (calendar-mode))
    (setq displayed-month month)
    (setq displayed-year year)
    (setq buffer-read-only nil)
    (erase-buffer)
    ;; horizontal rows
    (dotimes (j 4)
      ;; vertical columns
      (dotimes (i 3)
        (calendar-generate-month
          (setq month (+ month 1))
          year
          ;; indentation / spacing between months
          (+ 5 (* 25 i))))
      (goto-char (point-max))
      (insert (make-string (- 10 (count-lines (point-min) (point-max))) ?\n))
      (widen)
      (goto-char (point-max))
      (narrow-to-region (point-max) (point-max)))
    (widen)
    (goto-char (point-min))
    (setq buffer-read-only t)))

(defun dt/scroll-year-calendar-forward (&optional arg event)
  "Scroll the yearly calendar by year in a forward direction."
  (interactive (list (prefix-numeric-value current-prefix-arg)
                     last-nonmenu-event))
  (unless arg (setq arg 0))
  (save-selected-window
    (if (setq event (event-start event)) (select-window (posn-window event)))
    (unless (zerop arg)
      (let* (
             (year (+ displayed-year arg)))
        (dt/year-calendar year)))
    (goto-char (point-min))
    (run-hooks 'calendar-move-hook)))
```

```
(defun dt/scroll-year-calendar-backward (&optional arg event)
  "Scroll the yearly calendar by year in a backward direction."
  (interactive (list (prefix-numeric-value current-prefix-arg)
                     last-nonmenu-event))
  (dt/scroll-year-calendar-forward (- (or arg 1)) event))

(map! :leader
      :desc "Scroll year calendar backward" "<left>" #'dt/scroll-year-calendar-backward
      :desc "Scroll year calendar forward" "<right>" #'dt/scroll-year-calendar-forward)

(defalias 'year-calendar 'dt/year-calendar)
```

Let's also play around with calfw.

```
(use-package! calfw)
(use-package! calfw-org)
```

6 CENTAUR-TABS

To use tabs in Doom Emacs, be sure to uncomment "tabs" in Doom's init.el. Displays tabs at the top of the window similar to tabbed web browsers such as Firefox. I don't actually use tabs in Emacs. I placed this in my config to help others who may want tabs. In the default configuration of Doom Emacs, 'SPC t' is used for "toggle" keybindings, so I choose 'SPC t c' to toggle centaur-tabs. The "g" prefix for keybindings is used for a bunch of evil keybindings in Doom, but "g" plus the arrow keys were not used, so I thought I would bind those for tab navigation. But I did leave the default "g t" and "g T" intact if you prefer to use those for centaur-tabs-forward/backward.

COMMAND	DESCRIPTION	KEYBINDING
centaur-tabs-mode	*Toggle tabs globally*	SPC t c
centaur-tabs-local-mode	*Toggle tabs local display*	SPC t C
centaur-tabs-forward	*Next tab*	g <right> or g t
centaur-tabs-backward	*Previous tab*	g <left> or g T
centaur-tabs-forward-group	*Next tab group*	g <down>
centaur-tabs-backward-group	*Previous tab group*	g <up>

```
(setq centaur-tabs-set-bar 'over
      centaur-tabs-set-icons t
```

```
      centaur-tabs-gray-out-icons 'buffer
      centaur-tabs-height 24
      centaur-tabs-set-modified-marker t
      centaur-tabs-style "bar"
      centaur-tabs-modified-marker "•")
(map! :leader
      :desc "Toggle tabs globally" "t c" #'centaur-tabs-mode
      :desc "Toggle tabs local display" "t C" #'centaur-tabs-local-mode)
(evil-define-key 'normal centaur-tabs-mode-map (kbd "g <right>") 'centaur-tabs-forward
                                               (kbd "g <left>")  'centaur-tabs-backward
                                               (kbd "g <down>")  'centaur-tabs-forward-
                                               (kbd "g <up>")    'centaur-tabs-backward
```

7 CLIPPY

Gives us a popup box with "Clippy, the paper clip". You can make him say various things by calling 'clippy-say' function. But the more useful functions of clippy are the two describe functions provided: 'clippy-describe-function' and 'clippy-describe-variable'. Hit the appropriate keybinding while the point is over a function/variable to call it. A popup with helpful clippy will appear, telling you about the function/variable (using describe-function and describe-variable respectively).

COMMAND	DESCRIPTION	KEYBINDING
clippy-describe-function	*Clippy describes function under point*	SPC c h f
clippy-describe-variable	*Clippy describes variable under point*	SPC c h v

```
(map! :leader
      (:prefix ("c h" . "Help info from Clippy")
       :desc "Clippy describes function under point" "f" #'clippy-describe-function
       :desc "Clippy describes variable under point" "v" #'clippy-describe-variable))
```

8 DASHBOARD

Emacs Dashboard is an extensible startup screen showing you recent files, bookmarks, agenda items and an Emacs banner.

8.1 Configuring Dashboard

```
(use-package dashboard
  :init      ;; tweak dashboard config before loading it
  (setq dashboard-set-heading-icons t)
  (setq dashboard-set-file-icons t)
  (setq dashboard-banner-logo-title "\nKEYBINDINGS:\
\nFind file               (SPC .)     \
Open buffer list    (SPC b i)\
\nFind recent files       (SPC f r)   \
Open the eshell     (SPC e s)\
\nOpen dired file manager (SPC d d)   \
List of keybindings (SPC h b b)")
  ;;(setq dashboard-startup-banner 'logo) ;; use standard emacs logo as banner
  (setq dashboard-startup-banner "~/.config/doom/doom-emacs-dash.png")  ;; use custom i
  (setq dashboard-center-content nil) ;; set to 't' for centered content
  (setq dashboard-items '((recents . 5)
                          (agenda . 5 )
                          (bookmarks . 5)
                          (projects . 5)
                          (registers . 5)))
  :config
  (dashboard-setup-startup-hook)
  (dashboard-modify-heading-icons '((recents . "file-text")
                                    (bookmarks . "book"))))
```

8.2 Dashboard in Emacsclient

This setting ensures that emacsclient always opens on **dashboard** rather than **scratch**.

```
(setq doom-fallback-buffer-name "*dashboard*")
```

9 DIRED

Dired is the file manager within Emacs. Below, I setup keybindings for image previews (peep-dired). Doom Emacs does not use 'SPC d' for any of its keybindings, so I've chosen the format of 'SPC d' plus 'key'.

9.1 Keybindings To Open Dired

COMMAND	DESCRIPTION	KEYBINDING
dired	*Open dired file manager*	SPC d d
dired-jump	*Jump to current directory in dired*	SPC d j

9.2 Keybindings Within Dired

9.2.1 Basic dired commands

COMMAND	DESCRIPTION	KEYBINDING
dired-view-file	*View file in dired*	SPC d v
dired-up-directory	*Go up in directory tree*	h
dired-find-file	*Go down in directory tree (or open if file)*	l
dired-next-line	Move down to next line	j
dired-previous-line	Move up to previous line	k
dired-mark	Mark file at point	m
dired-unmark	Unmark file at point	u
dired-do-copy	Copy current file or marked files	C
dired-do-rename	Rename current file or marked files	R
dired-hide-details	Toggle detailed listings on/off	(
dired-git-info-mode	Toggle git information on/off	)
dired-create-directory	Create new empty directory	+
dired-diff	Compare file at point with another	=
dired-subtree-toggle	Toggle viewing subtree at point	TAB

9.2.2 Dired commands using regex

COMMAND	DESCRIPTION	KEYBINDING
dired-mark-files-regexp	Mark files using regex	% m
dired-do-copy-regexp	Copy files using regex	% C
dired-do-rename-regexp	Rename files using regex	% R
dired-mark-files-regexp	Mark all files using regex	* %

9.2.3 File permissions and ownership

COMMAND	DESCRIPTION	KEYBINDING
dired-do-chgrp	Change the group of marked files	g G
dired-do-chmod	Change the mode of marked files	M
dired-do-chown	Change the owner of marked files	O
dired-do-rename	Rename file or all marked files	R

```
(map! :leader
      (:prefix ("d" . "dired")
       :desc "Open dired" "d" #'dired
       :desc "Dired jump to current" "j" #'dired-jump)
      (:after dired
       (:map dired-mode-map
        :desc "Peep-dired image previews" "d p" #'peep-dired
        :desc "Dired view file" "d v" #'dired-view-file)))

(evil-define-key 'normal dired-mode-map
  (kbd "M-RET") 'dired-display-file
  (kbd "h") 'dired-up-directory
  (kbd "l") 'dired-open-file ; use dired-find-file instead of dired-open.
  (kbd "m") 'dired-mark
  (kbd "t") 'dired-toggle-marks
  (kbd "u") 'dired-unmark
  (kbd "C") 'dired-do-copy
  (kbd "D") 'dired-do-delete
  (kbd "J") 'dired-goto-file
  (kbd "M") 'dired-do-chmod
  (kbd "O") 'dired-do-chown
  (kbd "P") 'dired-do-print
  (kbd "R") 'dired-do-rename
  (kbd "T") 'dired-do-touch
  (kbd "Y") 'dired-copy-filenamecopy-filename-as-kill ; copies filename to kill ring.
  (kbd "Z") 'dired-do-compress
  (kbd "+") 'dired-create-directory
  (kbd "-") 'dired-do-kill-lines
  (kbd "% l") 'dired-downcase
  (kbd "% m") 'dired-mark-files-regexp
  (kbd "% u") 'dired-upcase
  (kbd "* %") 'dired-mark-files-regexp
  (kbd "* .") 'dired-mark-extension
  (kbd "* /") 'dired-mark-directories
  (kbd "; d") 'epa-dired-do-decrypt
  (kbd "; e") 'epa-dired-do-encrypt)
;; Get file icons in dired
(add-hook 'dired-mode-hook 'all-the-icons-dired-mode)
;; With dired-open plugin, you can launch external programs for certain extensions
;; For example, I set all .png files to open in 'sxiv' and all .mp4 files to open in 'm
```

```
(setq dired-open-extensions '(("gif" . "sxiv")
                              ("jpg" . "sxiv")
                              ("png" . "sxiv")
                              ("mkv" . "mpv")
                              ("mp4" . "mpv")))
```

9.3 Keybindings Within Dired With Peep-Dired-Mode Enabled

If peep-dired is enabled, you will get image previews as you go up/down with 'j' and 'k'

COMMAND	DESCRIPTION	KEYBINDING
peep-dired	*Toggle previews within dired*	SPC d p
peep-dired-next-file	*Move to next file in peep-dired-mode*	j
peep-dired-prev-file	*Move to previous file in peep-dired-mode*	k

```
(evil-define-key 'normal peep-dired-mode-map
  (kbd "j") 'peep-dired-next-file
  (kbd "k") 'peep-dired-prev-file)
(add-hook 'peep-dired-hook 'evil-normalize-keymaps)
```

9.4 Making deleted files go to trash can

```
(setq delete-by-moving-to-trash t
      trash-directory "~/.local/share/Trash/files/")
```

`NOTE`: For convenience, you may want to create a symlink to 'local/share/Trash' in your home directory:

```
cd ~/
ln -s ~/.local/share/Trash .
```

10 DOOM THEME

Setting the theme to doom-one. To try out new themes, I set a keybinding for counsel-load-theme with 'SPC h t'.

```
(setq doom-theme 'doom-ayu-dark)
(map! :leader
      :desc "Load new theme" "h t" #'counsel-load-theme)
```

11 ELFEED

An RSS newsfeed reader for Emacs.

```
(use-package! elfeed-goodies)
(elfeed-goodies/setup)
(setq elfeed-goodies/entry-pane-size 0.5)
(add-hook 'elfeed-show-mode-hook 'visual-line-mode)
(evil-define-key 'normal elfeed-show-mode-map
  (kbd "J") 'elfeed-goodies/split-show-next
  (kbd "K") 'elfeed-goodies/split-show-prev)
(evil-define-key 'normal elfeed-search-mode-map
  (kbd "J") 'elfeed-goodies/split-show-next
  (kbd "K") 'elfeed-goodies/split-show-prev)
(setq elfeed-feeds (quote
                    (("https://www.reddit.com/r/linux.rss" reddit linux)
                     ("https://www.reddit.com/r/commandline.rss" reddit commandline)
                     ("https://www.reddit.com/r/distrotube.rss" reddit distrotube)
                     ("https://www.reddit.com/r/emacs.rss" reddit emacs)
                     ("https://www.gamingonlinux.com/article_rss.php" gaming linux)
                     ("https://hackaday.com/blog/feed/" hackaday linux)
                     ("https://opensource.com/feed" opensource linux)
                     ("https://linux.softpedia.com/backend.xml" softpedia linux)
                     ("https://itsfoss.com/feed/" itsfoss linux)
                     ("https://www.zdnet.com/topic/linux/rss.xml" zdnet linux)
                     ("https://www.phoronix.com/rss.php" phoronix linux)
                     ("http://feeds.feedburner.com/d0od" omgubuntu linux)
                     ("https://www.computerworld.com/index.rss" computerworld linux)
                     ("https://www.networkworld.com/category/linux/index.rss" networkwo
                     ("https://www.techrepublic.com/rssfeeds/topic/open-source/" techre
                     ("https://betanews.com/feed" betanews linux)
                     ("http://lxer.com/module/newswire/headlines.rss" lxer linux)
                     ("https://distrowatch.com/news/dwd.xml" distrowatch linux))))
```

12 EMMS

One of the media players available for Emacs is emms, which stands for Emacs Multimedia System. By default, Doom Emacs does not use 'SPC a',' so the format I use for these bindings is 'SPC a' plus 'key'.

COMMAND	DESCRIPTION	KEYBINDING
emms-playlist-mode-go	*Switch to the playlist buffer*	SPC a a
emms-pause	*Pause the track*	SPC a x
emms-stop	*Stop the track*	SPC a s
emms-previous	*Play previous track in playlist*	SPC a p
emms-next	*Play next track in playlist*	SPC a n

```
(emms-all)
(emms-default-players)
(emms-mode-line 1)
(emms-playing-time 1)
(setq emms-source-file-default-directory "~/Music/"
      emms-playlist-buffer-name "*Music*"
      emms-info-asynchronously t
      emms-source-file-directory-tree-function 'emms-source-file-directory-tree-find)
(map! :leader
      (:prefix ("a" . "EMMS audio player")
       :desc "Go to emms playlist" "a" #'emms-playlist-mode-go
       :desc "Emms pause track" "x" #'emms-pause
       :desc "Emms stop track" "s" #'emms-stop
       :desc "Emms play previous track" "p" #'emms-previous
       :desc "Emms play next track" "n" #'emms-next))
```

13 EMOJIS

Emojify is an Emacs extension to display emojis. It can display github style emojis like :smile: or plain ascii ones like :).

```
(use-package emojify
  :hook (after-init . global-emojify-mode))
```

14 ERC

ERC is a built-in Emacs IRC client.

COMMAND	DESCRIPTION	KEYBINDING
erc-tls	*Launch ERC using more secure TLS connection*	SPC e E

```
(map! :leader
      (:prefix ("e". "evaluate/ERC/EWW")
       :desc "Launch ERC with TLS connection" "E" #'erc-tls))

(setq erc-prompt (lambda () (concat "[" (buffer-name) "]"))
      erc-server "irc.libera.chat"
      erc-nick "ysb"
      erc-user-full-name "Yuvraj Birdi"
      erc-track-shorten-start 24
      erc-autojoin-channels-alist '(("irc.libera.chat" "#archlinux" "#linux" "#emacs"))
      erc-kill-buffer-on-part t
      erc-fill-column 100
      erc-fill-function 'erc-fill-static
      erc-fill-static-center 20
      ;; erc-auto-query 'bury
      )
```

15 EVALUATE ELISP EXPRESSIONS

Changing some keybindings from their defaults to better fit with Doom Emacs, and to avoid conflicts with my window managers which sometimes use the control key in their keybindings. By default, Doom Emacs does not use 'SPC e' for anything, so I choose to use the format 'SPC e' plus 'key' for these (I also use 'SPC e' for 'eww' keybindings).

COMMAND	DESCRIPTION	KEYBINDING
eval-buffer	*Evaluate elisp in buffer*	SPC e b
eval-defun	*Evaluate the defun containing or after point*	SPC e d
eval-expression	*Evaluate an elisp expression*	SPC e e
eval-last-sexp	*Evaluate elisp expression before point*	SPC e l
eval-region	*Evaluate elisp in region*	SPC e r

```
(map! :leader
      (:prefix ("e". "evaluate/ERC/EWW")
       :desc "Evaluate elisp in buffer" "b" #'eval-buffer
       :desc "Evaluate defun" "d" #'eval-defun
       :desc "Evaluate elisp expression" "e" #'eval-expression
       :desc "Evaluate last sexpression" "l" #'eval-last-sexp
       :desc "Evaluate elisp in region" "r" #'eval-region))
```

16 EWW

EWW is the Emacs Web Wowser, the builtin browser in Emacs. Below I set urls to open in a specific browser (eww) with browse-url-browser-function. By default, Doom Emacs does not use 'SPC e' for anything, so I choose to use the format 'SPC e' plus 'key' for these (I also use 'SPC e' for 'eval' keybindings). I chose to use 'SPC s w' for eww-search-words because Doom Emacs uses 'SPC s' for 'search' commands.

```
(setq browse-url-browser-function 'eww-browse-url)
(map! :leader
      :desc "Search web for text between BEG/END"
      "s w" #'eww-search-words
      (:prefix ("e" . "evaluate/ERC/EWW")
       :desc "Eww web browser" "w" #'eww
       :desc "Eww reload page" "R" #'eww-reload))
```

17 EXWM

```
(autoload 'exwm-enable "exwm-config.el")
```

18 FONTS

Settings related to fonts within Doom Emacs:

- 'doom-font' – standard monospace font that is used for most things in Emacs.
- 'doom-variable-pitch-font' – variable font which is useful in some Emacs plugins.
- 'doom-big-font' – used in doom-big-font-mode; useful for presentations.
- 'font-lock-comment-face' – for comments.
- 'font-lock-keyword-face' – for keywords with special significance like 'setq' in elisp.

```
(setq doom-font (font-spec :family "JetBrains Mono" :size 15)
      doom-variable-pitch-font (font-spec :family "Ubuntu" :size 15)
      doom-big-font (font-spec :family "JetBrains Mono" :size 24))
```

```
(after! doom-themes
  (setq doom-themes-enable-bold t
        doom-themes-enable-italic t))
(custom-set-faces!
  '(font-lock-comment-face :slant italic)
  '(font-lock-keyword-face :slant italic))
```

19 INSERT DATE

Some custom functions to insert the date. The function 'insert-todays-date' can be used one of three different ways: (1) just the keybinding without the universal argument prefix, (2) with one universal argument prefix, or (3) with two universal argument prefixes. The universal argument prefix is 'SPC-u' in Doom Emacs (C-u in standard GNU Emacs). The function 'insert-any-date' only outputs to one format, which is the same format as 'insert-todays-date' without a prefix.

COMMAND	EXAMPLE OUTPUT	KEYBINDING
dt/insert-todays-date	Friday, November 19, 2021	SPC i d t
dt/insert-todays-date	11-19-2021	SPC u SPC i d t
dt/insert-todays-date	2021-11-19	SPC u SPC u SPC i d t
dt/insert-any-date	Friday, November 19, 2021	SPC i d a

```
(defun dt/insert-todays-date (prefix)
  (interactive "P")
  (let ((format (cond
                 ((not prefix) "%A, %B %d, %Y")
                 ((equal prefix '(4)) "%m-%d-%Y")
                 ((equal prefix '(16)) "%Y-%m-%d"))))
    (insert (format-time-string format))))

(require 'calendar)
(defun dt/insert-any-date (date)
  "Insert DATE using the current locale."
  (interactive (list (calendar-read-date)))
  (insert (calendar-date-string date)))

(map! :leader
      (:prefix ("i d" . "Insert date")
```

```
        :desc "Insert any date" "a" #'dt/insert-any-date
        :desc "Insert todays date" "t" #'dt/insert-todays-date))
```

20 IVY

Ivy is a generic completion mechanism for Emacs.

20.1 IVY-POSFRAME

Ivy-posframe is an ivy extension, which lets ivy use posframe to show its candidate menu. Some of the settings below involve:

- ivy-posframe-display-functions-alist – sets the display position for specific programs
- ivy-posframe-height-alist – sets the height of the list displayed for specific programs

Available functions (positions) for 'ivy-posframe-display-functions-alist'

- ivy-posframe-display-at-frame-center
- ivy-posframe-display-at-window-center
- ivy-posframe-display-at-frame-bottom-left
- ivy-posframe-display-at-window-bottom-left
- ivy-posframe-display-at-frame-bottom-window-center
- ivy-posframe-display-at-point
- ivy-posframe-display-at-frame-top-center

`NOTE:` If the setting for 'ivy-posframe-display' is set to 'nil' (false), anything that is set to 'ivy-display-function-fallback' will just default to their normal position in Doom Emacs (usually a bottom split). However, if this is set to 't' (true), then the fallback position will be centered in the window.

```
(setq ivy-posframe-display-functions-alist
      '((swiper                     . ivy-posframe-display-at-point)
        (complete-symbol            . ivy-posframe-display-at-point)
        (counsel-M-x                . ivy-display-function-fallback)
        (counsel-esh-history        . ivy-posframe-display-at-window-center)
```

```
        (counsel-describe-function  . ivy-display-function-fallback)
        (counsel-describe-variable  . ivy-display-function-fallback)
        (counsel-find-file          . ivy-display-function-fallback)
        (counsel-recentf            . ivy-display-function-fallback)
        (counsel-register           . ivy-posframe-display-at-frame-bottom-window-cente
        (dmenu                      . ivy-posframe-display-at-frame-top-center)
        (nil                        . ivy-posframe-display))
      ivy-posframe-height-alist
      '((swiper . 20)
        (dmenu . 20)
        (t . 10)))
(ivy-posframe-mode 1) ; 1 enables posframe-mode, 0 disables it.
```

20.2 IVY KEYBINDINGS

By default, Doom Emacs does not use 'SPC v', so the format I use for these bindings is 'SPC v' plus 'key'.

```
(map! :leader
      (:prefix ("v" . "Ivy")
       :desc "Ivy push view" "v p" #'ivy-push-view
       :desc "Ivy switch view" "v s" #'ivy-switch-view))
```

21 LINE SETTINGS

I set comment-line to 'SPC TAB TAB' which is a rather comfortable keybinding for me on my ZSA Moonlander keyboard. The standard Emacs keybinding for comment-line is 'C-x C-;'. The other keybindings are for commands that toggle on/off various line-related settings. Doom Emacs uses 'SPC t' for "toggle" commands, so I choose 'SPC t' plus 'key' for those bindings.

COMMAND	DESCRIPTION	KEYBINDING
comment-line	*Comment or uncomment lines*	SPC TAB TAB
hl-line-mode	*Toggle line highlighting in current frame*	SPC t h
global-hl-line-mode	*Toggle line highlighting globally*	SPC t H
doom/toggle-line-numbers	*Toggle line numbers*	SPC t l
toggle-truncate-lines	*Toggle truncate lines*	SPC t t

```
(setq display-line-numbers-type 't)
(setq display-line-numbers-type 'visual)
(map! :leader
      :desc "Comment or uncomment lines" "TAB TAB" #'comment-line
      (:prefix ("t" . "toggle")
       :desc "Toggle line numbers" "l" #'doom/toggle-line-numbers
       :desc "Toggle line highlight in frame" "h" #'hl-line-mode
       :desc "Toggle line highlight globally" "H" #'global-hl-line-mode
       :desc "Toggle truncate lines" "t" #'toggle-truncate-lines))
```

22 MARKDOWN

```
(custom-set-faces
 '(markdown-header-face ((t (:inherit font-lock-function-name-face :weight bold :family
 '(markdown-header-face-1 ((t (:inherit markdown-header-face :height 1.7))))
 '(markdown-header-face-2 ((t (:inherit markdown-header-face :height 1.6))))
 '(markdown-header-face-3 ((t (:inherit markdown-header-face :height 1.5))))
 '(markdown-header-face-4 ((t (:inherit markdown-header-face :height 1.4))))
 '(markdown-header-face-5 ((t (:inherit markdown-header-face :height 1.3))))
 '(markdown-header-face-6 ((t (:inherit markdown-header-face :height 1.2)))))
```

23 MINIMAP

A minimap sidebar displaying a smaller version of the current buffer on either the left or right side. It highlights the currently shown region and updates its position automatically. Be aware that this minimap program does not work in Org documents. This is not unusual though because I have tried several minimap programs and none of them can handle Org.

COMMAND	DESCRIPTION	KEYBINDING
minimap-mode	*Toggle minimap-mode*	SPC t m

```
(setq minimap-window-location 'right)
(map! :leader
      (:prefix ("t" . "toggle")
       :desc "Toggle minimap-mode" "m" #'minimap-mode))
```

24 MODELINE

The modeline is the bottom status bar that appears in Emacs windows. For more information on what is available to configure in the Doom modeline, check out: `https://github.com/seagle0128/doom-modeline`

```
(set-face-attribute 'mode-line nil :font "JetBrains Mono-13")
(setq doom-modeline-height 30     ;; sets modeline height
      doom-modeline-bar-width 5   ;; sets right bar width
      doom-modeline-persp-name t  ;; adds perspective name to modeline
      doom-modeline-persp-icon t) ;; adds folder icon next to persp name
```

25 MOUSE SUPPORT

Adding mouse support in the terminal version of Emacs.

```
(xterm-mouse-mode 1)
```

26 NEOTREE

Neotree is a file tree viewer. When you open neotree, it jumps to the current file thanks to neo-smart-open. The neo-window-fixed-size setting makes the neotree width be adjustable. Doom Emacs had no keybindings set for neotree. Since Doom Emacs uses 'SPC t' for 'toggle' keybindings, I used 'SPC t n' for toggle-neotree.

COMMAND	DESCRIPTION	KEYBINDING
neotree-toggle	*Toggle neotree*	SPC t n
neotree- dir	*Open directory in neotree*	SPC d n

```
(after! neotree
  (setq neo-smart-open t
        neo-window-fixed-size nil))
(after! doom-themes
  (setq doom-neotree-enable-variable-pitch t))
(map! :leader
      :desc "Toggle neotree file viewer" "t n" #'neotree-toggle
      :desc "Open directory in neotree" "d n" #'neotree-dir)
```

27 OPEN SPECIFIC FILES

Keybindings to open files that I work with all the time using the find-file command, which is the interactive file search that opens with 'C-x C-f' in GNU Emacs or 'SPC f f' in Doom Emacs. These keybindings use find-file non-interactively since we specify exactly what file to open. The format I use for these bindings is 'SPC =' plus 'key' since Doom Emacs does not use 'SPC ='.

PATH TO FILE	DESCRIPTION	KEYBINDING
~/Org/agenda.org	*Edit agenda file*	SPC = a
~/.config/doom/config.org	*Edit doom config.org*	SPC = c
~/.config/doom/init.el	*Edit doom init.el*	SPC = i
~/.config/doom/packages.el	*Edit doom packages.el*	SPC = p
~/.config/doom/eshell/aliases	*Edit eshell aliases*	SPC = e a
~/.config/doom/eshell/profile	*Edit eshell profile*	SPC = e p

```
(map! :leader
      (:prefix ("=" . "open file")
       :desc "Edit agenda file" "a" #'(lambda () (interactive) (find-file "~/nc/Org/age
       :desc "Edit doom config.org" "c" #'(lambda () (interactive) (find-file "~/.confi
       :desc "Edit doom init.el" "i" #'(lambda () (interactive) (find-file "~/.config/c
       :desc "Edit doom packages.el" "p" #'(lambda () (interactive) (find-file "~/.conf
(map! :leader
      (:prefix ("= e" . "open eshell files")
       :desc "Edit eshell aliases" "a" #'(lambda () (interactive) (find-file "~/.config
       :desc "Edit eshell profile" "p" #'(lambda () (interactive) (find-file "~/.config
```

28 ORG MODE

I wrapped most of this block in (after! org). Without this, my settings might be evaluated too early, which will result in my settings being overwritten by Doom's defaults. I have also enabled org-journal, org-superstar and org-roam by adding (+journal +pretty +roam2) to the org section of my Doom Emacs init.el.

`NOTE:` I have the location of my Org directory and Roam directory in $HOME/nc/ which is a Nextcloud folder that allows me to instantly sync all of my Org work between my home computer and my office computer.

```
(map! :leader
```

```
      :desc "Org babel tangle" "m B" #'org-babel-tangle)
(after! org
  (setq org-directory "~/nc/Org/"
        org-agenda-files '("~/nc/Org/agenda.org")
        org-default-notes-file (expand-file-name "notes.org" org-directory)
        org-ellipsis "  "
        org-superstar-headline-bullets-list '("" "" "" "" "" "" "")
        org-superstar-itembullet-alist '((?+ . ?) (?- . ?)) ; changes +/- symbols in it
        org-log-done 'time
        org-hide-emphasis-markers t
        ;; ex. of org-link-abbrev-alist in action
        ;; [[arch-wiki:Name_of_Page][Description]]
        org-link-abbrev-alist    ; This overwrites the default Doom org-link-abbrev-lis
          '(("google" . "http://www.google.com/search?q=")
            ("arch-wiki" . "https://wiki.archlinux.org/index.php/")
            ("ddg" . "https://duckduckgo.com/?q=")
            ("wiki" . "https://en.wikipedia.org/wiki/"))
        org-table-convert-region-max-lines 20000
        org-todo-keywords        ; This overwrites the default Doom org-todo-keywords
          '((sequence
             "TODO(t)"           ; A task that is ready to be tackled
             "BLOG(b)"           ; Blog writing assignments
             "GYM(g)"            ; Things to accomplish at the gym
             "PROJ(p)"           ; A project that contains other tasks
             "VIDEO(v)"          ; Video assignments
             "WAIT(w)"           ; Something is holding up this task
             "|"                 ; The pipe necessary to separate "active" states and "
             "DONE(d)"           ; Task has been completed
             "CANCELLED(c)" ))))) ; Task has been cancelled
```

Setting org-mode load languages

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

28.1 Org fonts

I have created an interactive function for each color scheme (M-x dt/org-colors-*). These functions will set appropriate colors and font attributes for org-level fonts and the org-table font.

```
(defun dt/org-colors-doom-one ()
  "Enable Doom One colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#51afef" ultra-bold)
         (org-level-2 1.6 "#c678dd" extra-bold)
         (org-level-3 1.5 "#98be65" bold)
         (org-level-4 1.4 "#da8548" semi-bold)
         (org-level-5 1.3 "#5699af" normal)
         (org-level-6 1.2 "#a9a1e1" normal)
         (org-level-7 1.1 "#46d9ff" normal)
         (org-level-8 1.0 "#ff6c6b" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-dracula ()
  "Enable Dracula colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#8be9fd" ultra-bold)
         (org-level-2 1.6 "#bd93f9" extra-bold)
         (org-level-3 1.5 "#50fa7b" bold)
         (org-level-4 1.4 "#ff79c6" semi-bold)
         (org-level-5 1.3 "#9aedfe" normal)
         (org-level-6 1.2 "#caa9fa" normal)
         (org-level-7 1.1 "#5af78e" normal)
         (org-level-8 1.0 "#ff92d0" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-gruvbox-dark ()
  "Enable Gruvbox Dark colors for Org headers."
```

```
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#458588" ultra-bold)
         (org-level-2 1.6 "#b16286" extra-bold)
         (org-level-3 1.5 "#98971a" bold)
         (org-level-4 1.4 "#fb4934" semi-bold)
         (org-level-5 1.3 "#83a598" normal)
         (org-level-6 1.2 "#d3869b" normal)
         (org-level-7 1.1 "#d79921" normal)
         (org-level-8 1.0 "#8ec07c" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-monokai-pro ()
  "Enable Monokai Pro colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#78dce8" ultra-bold)
         (org-level-2 1.6 "#ab9df2" extra-bold)
         (org-level-3 1.5 "#a9dc76" bold)
         (org-level-4 1.4 "#fc9867" semi-bold)
         (org-level-5 1.3 "#ff6188" normal)
         (org-level-6 1.2 "#ffd866" normal)
         (org-level-7 1.1 "#78dce8" normal)
         (org-level-8 1.0 "#ab9df2" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-nord ()
  "Enable Nord colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#81a1c1" ultra-bold)
         (org-level-2 1.6 "#b48ead" extra-bold)
         (org-level-3 1.5 "#a3be8c" bold)
         (org-level-4 1.4 "#ebcb8b" semi-bold)
         (org-level-5 1.3 "#bf616a" normal)
```

```
          (org-level-6 1.2 "#88c0d0" normal)
          (org-level-7 1.1 "#81a1c1" normal)
          (org-level-8 1.0 "#b48ead" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-oceanic-next ()
  "Enable Oceanic Next colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#6699cc" ultra-bold)
         (org-level-2 1.6 "#c594c5" extra-bold)
         (org-level-3 1.5 "#99c794" bold)
         (org-level-4 1.4 "#fac863" semi-bold)
         (org-level-5 1.3 "#5fb3b3" normal)
         (org-level-6 1.2 "#ec5f67" normal)
         (org-level-7 1.1 "#6699cc" normal)
         (org-level-8 1.0 "#c594c5" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-palenight ()
  "Enable Palenight colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#82aaff" ultra-bold)
         (org-level-2 1.6 "#c792ea" extra-bold)
         (org-level-3 1.5 "#c3e88d" bold)
         (org-level-4 1.4 "#ffcb6b" semi-bold)
         (org-level-5 1.3 "#a3f7ff" normal)
         (org-level-6 1.2 "#e1acff" normal)
         (org-level-7 1.1 "#f07178" normal)
         (org-level-8 1.0 "#ddffa7" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-solarized-dark ()
  "Enable Solarized Dark colors for Org headers."
```

```
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#268bd2" ultra-bold)
         (org-level-2 1.6 "#d33682" extra-bold)
         (org-level-3 1.5 "#859900" bold)
         (org-level-4 1.4 "#b58900" semi-bold)
         (org-level-5 1.3 "#cb4b16" normal)
         (org-level-6 1.2 "#6c71c4" normal)
         (org-level-7 1.1 "#2aa198" normal)
         (org-level-8 1.0 "#657b83" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-solarized-light ()
  "Enable Solarized Light colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#268bd2" ultra-bold)
         (org-level-2 1.6 "#d33682" extra-bold)
         (org-level-3 1.5 "#859900" bold)
         (org-level-4 1.4 "#b58900" semi-bold)
         (org-level-5 1.3 "#cb4b16" normal)
         (org-level-6 1.2 "#6c71c4" normal)
         (org-level-7 1.1 "#2aa198" normal)
         (org-level-8 1.0 "#657b83" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

(defun dt/org-colors-tomorrow-night ()
  "Enable Tomorrow Night colors for Org headers."
  (interactive)
  (dolist
      (face
       '((org-level-1 1.7 "#81a2be" ultra-bold)
         (org-level-2 1.6 "#b294bb" extra-bold)
         (org-level-3 1.5 "#b5bd68" bold)
         (org-level-4 1.4 "#e6c547" semi-bold)
         (org-level-5 1.3 "#cc6666" normal)
```

```
        (org-level-6 1.2 "#70c0ba" normal)
        (org-level-7 1.1 "#b77ee0" normal)
        (org-level-8 1.0 "#9ec400" normal)))
    (set-face-attribute (nth 0 face) nil :font doom-variable-pitch-font :weight (nth 3
    (set-face-attribute 'org-table nil :font doom-font :weight 'normal :height 1.0 :fo

;; Load our desired dt/org-colors-* theme on startup
(dt/org-colors-doom-one)
```

28.2 Org-export

We need ox-man for "Org eXporting" to manpage format and ox-gemini for exporting to gemtext (for the gemini protocol).

NOTE: I also enable ox-publish for converting an Org site into an HTML site, but that is done in init.el (org +publish).

```
(use-package ox-man)
(use-package ox-gemini)
```

28.3 Org-journal

```
(setq org-journal-dir "~/nc/Org/journal/"
      org-journal-date-prefix "* "
      org-journal-time-prefix "** "
      org-journal-date-format "%B %d, %Y (%A) "
      org-journal-file-format "%Y-%m-%d.org")
```

28.4 Org-publish

```
(setq org-publish-use-timestamps-flag nil)
(setq org-export-with-broken-links t)
(setq org-publish-project-alist
      '(("distro.tube without manpages"
         :base-directory "~/nc/gitlab-repos/distro.tube/"
         :base-extension "org"
         :publishing-directory "~/nc/gitlab-repos/distro.tube/html/"
         :recursive t
         :exclude "org-html-themes/.*\\|man-org/man*"
         :publishing-function org-html-publish-to-html
         :headline-levels 4             ; Just the default for this project.
```

```
:auto-preamble t)
("man0p"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man0p/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man0p/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man1"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man1/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man1/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man1p"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man1p/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man1p/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man2"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man2/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man2/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man3"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man3/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man3/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
```

```
:auto-preamble t)
("man3p"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man3p/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man3p/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man4"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man4/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man4/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man5"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man5/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man5/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man6"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man6/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man6/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
:auto-preamble t)
("man7"
:base-directory "~/nc/gitlab-repos/distro.tube/man-org/man7/"
:base-extension "org"
:publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man7/"
:recursive t
:publishing-function org-html-publish-to-html
:headline-levels 4             ; Just the default for this project.
```

```
        :auto-preamble t)
        ("man8"
        :base-directory "~/nc/gitlab-repos/distro.tube/man-org/man8/"
        :base-extension "org"
        :publishing-directory "~/nc/gitlab-repos/distro.tube/html/man-org/man8/"
        :recursive t
        :publishing-function org-html-publish-to-html
        :headline-levels 4             ; Just the default for this project.
        :auto-preamble t)
        ("org-static"
        :base-directory "~/Org/website"
        :base-extension "css\\|js\\|png\\|jpg\\|gif\\|pdf\\|mp3\\|ogg\\|swf"
        :publishing-directory "~/public_html/"
        :recursive t
        :exclude ".*/org-html-themes/.*"
        :publishing-function org-publish-attachment)
        ("dtos.dev"
        :base-directory "~/nc/gitlab-repos/dtos.dev/"
        :base-extension "org"
        :publishing-directory "~/nc/gitlab-repos/dtos.dev/html/"
        :recursive t
        :publishing-function org-html-publish-to-html
        :headline-levels 4             ; Just the default for this project.
        :auto-preamble t)

      ))
```

28.5 Org-auto-tangle

`org-auto-tangle` allows you to add the option `#+auto_tangle: t` in your Org file so that it automatically tangles when you save the document.

```
(use-package! org-auto-tangle
  :defer t
  :hook (org-mode . org-auto-tangle-mode)
  :config
  (setq org-auto-tangle-default t))
```

29 PASSWORD STORE

Uses the standard Unix password store "pass".

```
(use-package! password-store)
```

30 PERSPECTIVE

Perspective provides multiple named workspaces (or "perspectives") in Emacs, similar to having multiple desktops in window managers like Awesome and XMonad. Each perspective has its own buffer list and its own window layout, making it easy to work on many separate projects without getting lost in all the buffers. Switching to a perspective activates its window configuration, and when in a perspective, only its buffers are available (by default). Doom Emacs uses 'SPC some$_{\text{key}}$' for binding some of the perspective commands, so I used this binging format for the perspective bindings that I created..

COMMAND	DESCRIPTION	KEYBINDING
persp-switch	Switch to perspective NAME	SPC DEL
persp-switch-to-buffer	Switch to buffer in perspective	SPC ,
persp-next	Switch to next perspective	SPC]
persp-prev	Switch to previous perspective	SPC [
persp-add-buffer	Add a buffer to current perspective	SPC +
persp-remove-by-name	Remove perspective by name	SPC -
+workspace/switch-to-{0-9}	Switch to workspace n	SPC 0-9

```
(map! :leader
      :desc "Switch to perspective NAME" "DEL" #'persp-switch
      :desc "Switch to buffer in perspective" "," #'persp-switch-to-buffer
      :desc "Switch to next perspective" "]" #'persp-next
      :desc "Switch to previous perspective" "[" #'persp-prev
      :desc "Add a buffer current perspective" "+" #'persp-add-buffer
      :desc "Remove perspective by name" "-" #'persp-remove-by-name)
```

31 RAINBOW MODE

Rainbox mode displays the actual color for any hex value color. It's such a nice feature that I wanted it turned on all the time, regardless of what mode I am in. The following creates a global minor mode for rainbow-mode

and enables it (exception: org-agenda-mode since rainbow-mode destroys all highlighting in org-agenda).

```
(define-globalized-minor-mode global-rainbow-mode rainbow-mode
  (lambda ()
    (when (not (memq major-mode
                (list 'org-agenda-mode)))
     (rainbow-mode 1))))
(global-rainbow-mode 1 )
```

32 REGISTERS

Emacs registers are compartments where you can save text, rectangles and positions for later use. Once you save text or a rectangle in a register, you can copy it into the buffer once or many times; once you save a position in a register, you can jump back to that position once or many times. The default GNU Emacs keybindings for these commands (with the exception of counsel-register) involves 'C-x r' followed by one or more other keys. I wanted to make this a little more user friendly, and since I am using Doom Emacs, I choose to replace the 'C-x r' part of the key chords with 'SPC r'.

COMMAND	DESCRIPTION	KEYBINDING
copy-to-register	*Copy to register*	SPC r c
frameset-to-register	*Frameset to register*	SPC r f
insert-register	*Insert contents of register*	SPC r i
jump-to-register	*Jump to register*	SPC r j
list-registers	*List registers*	SPC r l
number-to-register	*Number to register*	SPC r n
counsel-register	*Interactively choose a register*	SPC r r
view-register	*View a register*	SPC r v
window-configuration-to-register	*Window configuration to register*	SPC r w
increment-register	*Increment register*	SPC r +
point-to-register	*Point to register*	SPC r SPC

```
(map! :leader
      (:prefix ("r" . "registers")
       :desc "Copy to register" "c" #'copy-to-register
       :desc "Frameset to register" "f" #'frameset-to-register
       :desc "Insert contents of register" "i" #'insert-register
```

```
      :desc "Jump to register" "j" #'jump-to-register
      :desc "List registers" "l" #'list-registers
      :desc "Number to register" "n" #'number-to-register
      :desc "Interactively choose a register" "r" #'counsel-register
      :desc "View a register" "v" #'view-register
      :desc "Window configuration to register" "w" #'window-configuration-to-register
      :desc "Increment register" "+" #'increment-register
      :desc "Point to register" "SPC" #'point-to-register))
```

33 SHELLS

Settings for the various shells and terminal emulators within Emacs.

- 'shell-file-name' – sets the shell to be used in M-x shell, M-x term, M-x ansi-term and M-x vterm.
- 'eshell-aliases-file' – sets an aliases file for the eshell.

```
(setq shell-file-name "/bin/fish"
      vterm-max-scrollback 5000)
(setq eshell-rc-script "~/.config/doom/eshell/profile"
      eshell-aliases-file "~/.config/doom/eshell/aliases"
      eshell-history-size 5000
      eshell-buffer-maximum-lines 5000
      eshell-hist-ignoredups t
      eshell-scroll-to-bottom-on-input t
      eshell-destroy-buffer-when-process-dies t
      eshell-visual-commands'("bash" "fish" "htop" "ssh" "top" "zsh"))
(map! :leader
      :desc "Eshell" "e s" #'eshell
      :desc "Eshell popup toggle" "e t" #'+eshell/toggle
      :desc "Counsel eshell history" "e h" #'counsel-esh-history
      :desc "Vterm popup toggle" "v t" #'+vterm/toggle)
```

34 SPLITS

I set splits to default to opening on the right using 'prefer-horizontal-split'. I set a keybinding for 'clone-indirect-buffer-other-window' for when I want to have the same document in two splits. The text of the indirect buffer is always identical to the text of its base buffer; changes made by editing

either one are visible immediately in the other. But in all other respects, the indirect buffer and its base buffer are completely separate. For example, I can fold one split but other will be unfolded.

```
(defun prefer-horizontal-split ()
  (set-variable 'split-height-threshold nil t)
  (set-variable 'split-width-threshold 40 t)) ; make this as low as needed
(add-hook 'markdown-mode-hook 'prefer-horizontal-split)
(map! :leader
      :desc "Clone indirect buffer other window" "b c" #'clone-indirect-buffer-other-w
```

35 WINNER MODE

Winner mode has been included with GNU Emacs since version 20. This is a global minor mode and, when activated, it allows you to "undo" (and "redo") changes in the window configuration with the key commands 'SCP w <left>' and 'SPC w <right>'.

```
(map! :leader
      (:prefix ("w" . "window")
       :desc "Winner redo" "<right>" #'winner-redo
       :desc "Winner undo" "<left>" #'winner-undo))
```

36 ZAP TO CHAR

Emacs provides a 'zap-to-char' command that kills from the current point to a character. It is bound to 'M-z' in standard GNU Emacs but since Doom Emacs uses 'SPC' as its leader key and does not have 'SPC z' binded to anything, it just makes since to use it for 'zap-to-char'. Note that 'zap-to-char' can be used with the universal argument 'SPC u' to modify its behavior. Examples of 'zap-to-char' usage are listed in the table below:

KEYBINDING	WHAT IS DOES
SPC z e	deletes all chars to the next occurrence of 'e'
SPC u 2 SPC z e	deletes all chars to the second occurrence of 'e'
SPC u - SPC z e	deletes all chars to the previous occurrence of 'e'
SPC u - 2 SPC z e	deletes all chars to the second previous occurrence of 'e'
SPC u 1 0 0 SPC u SPC z e	deletes all chars to the 100th occurrence of 'e'

TIP: The universal argument (SPC u) can only take a single integer by default. If you need to use a multi-digit number (like 100 in the last example in the table above), then you must terminate the universal argument with another 'SPC u' after typing the number.

'zap-up-to-char' is an alternative command that does not zap the char specified. It is binded to 'SPC Z'. It can also be used in conjunction with the universal argument 'SPC u' in similar fashion to the the 'zap-to-char' examples above.

NOTE: Vim (evil mode) has similar functionality builtin. You can delete to the next occurrence of 'e' by using 'dte' in normal. To delete to the next occurrence of 'e' including the 'e', then you would use 'dfe'. And you can modify 'dt' and 'df' by prefixing them with numbers, so '2dte' would delete to the second occurrence of 'e'.

```
(map! :leader
      :desc "Zap to char" "z" #'zap-to-char
      :desc "Zap up to char" "Z" #'zap-up-to-char)
```

37 FLYSPELL

```
;; (add-hook 'text-mode-hook 'flyspell-mode)
;; (add-hook 'prog-mode-hook 'flyspell-prog-mode)
;; (setq-default spell-checking-enable-by-default nil)
;; (setq flyspell-issue-message-flag nil
;;       ispell-program-name "aspell"
;;       ispell-extra-args '("--sug-mode=ultra"))
```

38 JUPYTER

```
;; (setq org-latex-pdf-process (list
;;    "latexmk -pdflatex='lualatex -shell-escape -interaction nonstopmode' -pdf -f  %f
```