My Emacs Configuration

January 22, 2022

Contents

1 Straight

```
(defvar bootstrap-version)
(let ((bootstrap-file
       (expand-file-name "straight/repos/straight.el/bootstrap.el" user-emacs-directory
      (bootstrap-version 5))
  (unless (file-exists-p bootstrap-file)
    (with-current-buffer
(url-retrieve-synchronously
 "https://raw.githubusercontent.com/raxod502/straight.el/develop/install.el"
 'silent 'inhibit-cookies)
      (goto-char (point-max))
      (eval-print-last-sexp)))
  (load bootstrap-file nil 'nomessage))
```

```
(straight-use-package 'use-package)
;; (setq use-package-verbose t)
```

2 Custom Settings

Set seperate custom file so that custom settings don't get overwritten by org-babel-tangle.

```
(setq custom-file "~/.emacs.d/custom.el")
(load custom-file)
```

3 Emacs

This section is for miscellaneous settings which do not belong elsewhere, everything should be commented explaining why it was added.

```
;; clean scratch buffer
(setq initial-scratch-message nil)

;; fix sizing errors with tiling window managers
(setq frame-resize-pixelwise t)
```

4 Backup

```
(setq backup-directory-alist `(("." . "~/.saves")))
(setq backup-by-copying t)
(setq delete-old-versions t
      kept-new-versions 6
      kept-old-versions 2
      version-control t)
```

5 General

This section sets up general.el. This section should not be used to set keybindings (except for the most general ones). Keybindings should go with their respective packages.

```
(use-package general
  :straight t
  :config
  (general-auto-unbind-keys)
  (general-evil-setup)

  (general-create-definer leader-key-def
    :keymaps '(normal insert visual emacs)
    :prefix "SPC"
    :global-prefix "C-SPC")
  )
```

6 Dired

```
(use-package dired
  :init
  (global-auto-revert-mode)
  (use-package all-the-icons-dired
    :straight t
    :hook (dired-mode . all-the-icons-dired-mode))


  (use-package dired-x
    :after general
    :general
    (leader-key-def
      "<SPC>" 'dired-jump
      "c" '(lambda () (interactive) (find-file "~/.local/share/chezmoi/dot_emacs.d/ema
      ))
  :custom
  (dired-listing-switches "-ABDgho --group-directories-first")
  (global-auto-revert-non-file-buffers t)
  :general
  (
   :keymaps 'dired-mode-map
   "h" 'dired-up-directory
   "l" 'dired-find-file)
  )
```

7 Minibuffer

```
(use-package selectrum
  :straight t
  :init
  (selectrum-mode +1))

(use-package selectrum-prescient
  :straight t
  :init
  (selectrum-prescient-mode +1)
  (prescient-persist-mode +1)
  )

(use-package consult
  :straight t
  :after selectrum
  :custom
  (consult-project-root-function #'projectile-project-root)
  :config
  (autoload 'projectile-project-root "projectile")

  :general
  (leader-key-def
    "/" 'consult-line
    "b" 'consult-buffer
    "r" 'consult-recent-file
    "i" 'consult-imenu
    "s" 'consult-ripgrep
    )
  )

(use-package marginalia
  :straight t
  :after selectrum
  :custom
  (marginalia-annotators '(marginalia-annotators-heavy))
  :init
  (marginalia-mode)
  )
```

8 Embark

```
(use-package embark
  :straight t
  :defer t

  :general
  ("M-m"  'embark-act)         ;; pick some comfortable binding
  ("C-;" 'embark-dwim)        ;; good alternative: M-.
  ("C-h B" 'embark-bindings) ;; alternative for `describe-bindings'

  :init

  ;; Optionally replace the key help with a completing-read interface
  (setq prefix-help-command #'embark-prefix-help-command)

  :config

  ;; Hide the mode line of the Embark live/completions buffers
  (add-to-list 'display-buffer-alist
       '("\\`\\*Embark Collect \\(Live\\|Completions\\)\\*"
 nil
 (window-parameters (mode-line-format . none)))))

(use-package embark-consult
  :straight t
  :after (embark consult)
  :demand t ; only necessary if you have the hook below
  ;; if you want to have consult previews as you move around an
  ;; auto-updating embark collect buffer
  :hook
  (embark-collect-mode . consult-preview-at-point-mode))
```

9 Eshell

Configuration and setup for eshell

```
(use-package eshell
  :init
  (defun eshell-other-window ()
```

```
    "Open a ‘eshell’ in a new window."
    (interactive)
    (let ((buf (eshell)))
      (switch-to-buffer (other-buffer buf))
      (switch-to-buffer-other-window buf)))
  :general
  (leader-key-def
    "t" 'eshell-other-window)
  )
```

10 Org Mode

```
(use-package org
  :straight
  :defer t
  :init
  (use-package evil-org
    :straight t
    :after (org evil)
    :hook (org-mode . (lambda () evil-org-mode))
    :config
    (require 'evil-org-agenda)
    (evil-org-agenda-set-keys))

  (use-package org-superstar
    :straight t
    :after org
    :hook (org-mode . (lambda () (org-superstar-mode 1))))

  (use-package ox-pandoc
    :straight t
    :after org
    )

  (use-package org-download
    :straight t)

  :custom
  (org-confirm-babel-evaluate nil)
```

```
  (org-src-window-setup 'current-window)
  (org-M-RET-may-split-line nil)
  (org-image-actual-width 400)
  (org-export-with-author "Ethan Coe-Renner")

  (org-default-notes-file (concat org-directory "/notes.org"))

  (org-capture-bookmark nil)
  (org-capture-templates
   '(("t" "add a [t]ask" entry (file (concat org-directory "/tasks.org"))
      "* TODO %?" :kill-buffer t)))

  (require 'org-tempo)

  :general
  ("C-c c" 'org-capture)
  ("C-c a" 'org-agenda)

  :config
  (org-babel-do-load-languages
   'org-babel-load-languages
   '((emacs-lisp . t)
     ))
  (add-to-list 'org-structure-template-alist '("el" . "src emacs-lisp"))
  (add-to-list 'org-agenda-files (concat org-directory "/tasks.org"))


  :hook
  (org-mode . org-indent-mode)
  (org-capture-mode . evil-insert-state)
  )
```

11 Literate Calc

```
(use-package literate-calc-mode
  :straight t
  )
```

12 Yequake

Create temporary emacs frames for simple tasks, i.e. org capture. Currently will not work unless an emacs frame is already open as per this issue, which sort of makes this useless for my purposes (which are quickly adding tasks from outside of emacs). Despite that issue, I am leaving this here in case the issue is fixed or I can find a non-hacky workaround, because the concept is really cool. Also, doesn't seem to load properly.

```
;; (use-package yequake
;;   :custom
;;   (yequake-frames
;;    '(("org-capture"
;;       (buffer-fns . (yequake-org-capture))
;;       (width . 0.75)
;;       (height . 0.5)
;;       (alpha . 0.95)
;;       (frame-parameters . ((undecorated . t)
;;                            (skip-taskbar . t)
;;                            (sticky . t))))))
;;   )
```

13 EVIL

setup evil and related packages

```
(use-package evil
  :straight t
  :init
  (global-visual-line-mode 1)

  (use-package undo-fu :straight t)
  (use-package evil-collection
    :straight t
    :init
    :after evil
    :config
    (evil-collection-init))

  (use-package evil-goggles
```

```
  :straight t
  :config
  (evil-goggles-mode 1))

(use-package evil-commentary
  :straight t
  :config
  (evil-commentary-mode 1))

(use-package evil-snipe
  :straight t
  :init
  (evil-snipe-mode 1)
  (evil-snipe-override-mode 1)
  :custom
  (evil-snipe-scope 'visible)
  (evil-snipe-repeat-scope 'visible)
  :hook (magit-mode . turn-off-evil-snipe-override-mode)
  )

(use-package evil-multiedit
  :straight t
  :general
  (:states '(normal visual)
   "R" 'evil-multiedit-match-all
   "M-d" 'evil-multiedit-match-and-next
   "M-D" 'evil-multiedit-match-and-prev
   )
  )
(use-package evil-surround
  :straight t
  :config
  (global-evil-surround-mode 1))

:custom
(evil-undo-system 'undo-fu)
(evil-want-C-u-scroll t)
(evil-respect-visual-line-mode t)
(evil-want-keybinding nil)
```

```
  :config
  (evil-mode 1)
  (general-def
    "C-M-u" 'universal-argument ;; doesn't work with :general for some reason
    )
  (general-def
    :states 'motion
    "j" 'evil-next-visual-line
    "k" 'evil-previous-visual-line)
  )
```

14 Help

```
(use-package which-key
  :defer t
  :straight t
  :init (which-key-mode)
  :custom
  (which-key-idle-delay 0.3))

(use-package helpful
  :straight t
  :general
  (
   "C-h f" 'helpful-callable
   "C-h v" 'helpful-variable
   "C-h k" 'helpful-key
   "C-c C-h" 'helpful-at-point
   )
  )

(use-package define-word
  :straight t
  :general
  ("C-h C-w" 'define-word-at-point)
  )
```

15 Elfeed

Setup for Elfeed, an RSS reader

```
(use-package elfeed
  :straight t
  :general
  (leader-key-def
    "r" 'elfeed)
  :custom
  (elfeed-feeds
   '(
     ;; Blogs
     ("http://daviddfriedman.blogspot.com/atom.xml" blog)
     ("http://www.econlib.org/feed/indexCaplan_xml" blog)
     ("https://nullprogram.com/feed/" blog)
     ("https://feeds.feedburner.com/mrmoneymustache" blog)
     ("https://astralcodexten.substack.com/feed" blog)
     ("https://www.singlelunch.com/feed" blog)
     ("https://www.overcomingbias.com/feed" blog)
     ("https://protesilaos.com/codelog.xml" blog)
     ("https://protesilaos.com/politics.xml" blog)

     ;; Fora
     ("https://www.lesswrong.com/feed.xml?view=curated-rss" forum)

     ;; Comics
     ("https://xkcd.com/rss.xml" comic)
     ("https://www.monkeyuser.com/feed.xml" comic)
     ))
  :hook
  (elfeed-search-mode . elfeed-update) ;; auto update when opened
  :config
  )
```

16 Music

```
(use-package emms
  :straight t
  :custom
```

```
  (emms-source-file-default-directory "~/media/music/")
  :init
  (emms-all)
  (emms-default-players)
  :general
  (leader-key-def
    "m" 'emms
    ))
```

17 Internet

Set default browser to qutebrowser. Currently doesn't work and isn't tangled.

```
(defun browse-url-qutebrowser (url)
  "Open URL in qutebrowser."
  (interactive (browse-url-interactive-arg "URL: "))
  (setq url (browse-url-encode-url url))
  (start-process (concat "qutebrowser " url) nil
 "qutebrowser"
 url
 ))

(setq browse-url-browser-function 'browse-url-qutebrowser)
```

For the time being, use eww by default

```
(setq browse-url-browser-function 'eww-browse-url)
```

18 Editing

This section contains packages and settings for non-evil specific editing

```
;; Delimiters
(use-package rainbow-delimiters
  :straight t
  :hook (prog-mode . rainbow-delimiters-mode))

(show-paren-mode 1)
(electric-pair-mode 1)
(setq electric-pair-inhibit-predicate 'electric-pair-conservative-inhibit)
```

19 Navigation

This section contains packages/configuration for non-evil-specific navigation

```
(use-package avy
  :straight t
  :general
  ("C-s" 'avy-goto-char-timer)
  )

(use-package winum
  :straight t
  :general
  (
   "M-1" 'winum-select-window-1
   "M-2" 'winum-select-window-2
   "M-3" 'winum-select-window-3
   "M-4" 'winum-select-window-4
   "M-5" 'winum-select-window-5
   "M-6" 'winum-select-window-6
   "M-7" 'winum-select-window-7
   "M-8" 'winum-select-window-8
   )
  :config
  (winum-mode t))

(use-package smartscan
  :straight t
  :hook (prog-mode . smartscan-mode))

(use-package rg
  :defer t
  :straight t
  :config
  (rg-enable-default-bindings))
```

20 Formatting

Automatic formatting

```
(use-package aggressive-indent
```

```
  :straight t
  :hook (prog-mode . aggressive-indent-mode)
  )

(use-package format-all
  :straight t
  :hook
  (prog-mode . format-all-mode)
  )
```

21 Git

Setup git integration

```
(use-package magit
  :commands magit-status
  :straight t
  :general
  (leader-key-def
    "g" 'magit-status)
  )
```

22 Discord

Setup git integration

```
(use-package elcord)
```

23 Project Management

```
(use-package projectile
  :straight t
  :custom
  (projectile-switch-project-action #'projectile-dired)
  :config (projectile-mode)
  :general
  (leader-key-def
    "p" 'projectile-command-map
    ))
```

24 Major Modes

Set up major modes for languages, etc

```
(use-package toml-mode :straight t
  :mode "\\.toml\\'")
(use-package yaml-mode
  :straight t
  :mode "\\.yml\\'"
  )
(use-package rustic :straight t)
(use-package nix-mode :straight t
  :mode "\\.nix\\'")
(use-package json-mode :straight t
  :mode "\\.json\\'")

(use-package mips-mode :straight t
  :mode "\\.asm\\'")

(use-package kbd-mode
  :straight (kbd-mode :type git :host github :repo "kmonad/kbd-mode")
  :mode "\\.kbd\\'")
```

25 Checkers

Set up checkers, i.e. syntax checking, spell checkers, etc

```
(use-package flycheck
  :straight t
  :defer t
  :init (global-flycheck-mode)
  )
```

26 Completion

```
(use-package company
  :straight t
  :custom
  (company-minimum-prefix-length 3)
  :hook
```

```
  (after-init . global-company-mode)
  )
```

27 LSP

```
(use-package lsp-mode
  :straight t
  :custom
  (gc-cons-threshold 100000000) ;; set per the lsp-doctor recommendation
  (read-process-output-max (* 1024 1024)) ;; same reason ^
  (lsp-keymap-prefix "C-c l")
  :hook (
 (rustic-mode . lsp)
 (c-mode . lsp)
 (lsp-mode . lsp-enable-which-key-integration))
  :commands lsp)

(use-package lsp-ui
  :straight t
  :hook (lsp-mode . lsp-ui-mode)
  :commands lsp-ui-mode)
(use-package lsp-treemacs
  :straight t
  :after lsp-mode
  :commands lsp-treemacs-errors-list)
```

28 GUI

Set gui settings, theme, fonts, etc

```
;; disabling useless ui elements
(scroll-bar-mode -1)
(menu-bar-mode -1)
(tool-bar-mode -1)
(setq inhibit-startup-screen t)

(global-hl-line-mode)

;; theme
```

```
(use-package doom-themes
  :straight t
  :init
  (load-theme 'doom-gruvbox t)
  (defun doom-one-themes-toggle ()
    "Toggle between `doom-one' and `doom-one-light' themes."
    (interactive)
    (pcase (car custom-enabled-themes)
      ('doom-gruvbox (load-theme 'doom-gruvbox-light))
      ('doom-gruvbox-light (load-theme 'doom-gruvbox))
      ))
  :general
  ("<f5>" 'doom-one-themes-toggle)
  )

;; dashboard
(use-package dashboard
  :straight t
  :custom
  (initial-buffer-choice (lambda () (get-buffer "*dashboard*")))
  (dashboard-startup-banner 'official)
  (dashboard-set-heading-icons t)
  (dashboard-set-file-icons t)
  (dashboard-set-init-info t)
  (dashboard-center-content t)
  (dashboard-items '((bookmarks . 5)
     (projects . 5)))

  :config
  (dashboard-setup-startup-hook))


;; font
(set-face-attribute 'default nil :font "Source Code Pro" :height 120)
(set-face-attribute 'fixed-pitch nil :font "Source Code Pro" :height 120)

;; line numbers
(setq display-line-numbers 'relative)
(dolist (mode '(text-mode-hook
prog-mode-hook
```

```
conf-mode-hook
rust-mode-hook))
  (add-hook mode (lambda () (display-line-numbers-mode 1))))

;; modeline
(use-package doom-modeline
  :straight t
  :custom
  (doom-modeline-icon t) ;; fix icons in server
  :init
  (doom-modeline-mode 1))
```

29 local variables

;; Local Variables: ;; eval: (add-hook 'after-save-hook (lambda ()(if (y-or-n-p "Tangle?")(org-babel-tangle))) nil t) ;; End: