Emacs configurations

Jefter Santiago

July 8, 2021

Contents

1 General config

1.1 Personal configs

```
(setq-default user-full-name "Jefter␣Santiago")
(setq-default user-mail-address "
   jeftersantiago@protonmail.com")
(load "~/.local/bin/private.el")
```

Separated file to store generated config from emacs.

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

1.2 Backup disabled

```
(setq-default backup-inhibited t)
(setq-default create-lockfiles nil)
(setq-default make-backup-files nil)
(use-package real-auto-save
  :ensure t
  :demand t
  :config (setq real-auto-save-interval 10)
  :hook (prog-mode . real-auto-save-mode))
```

2 UI Configuration

2.1 Basics

```
(scroll-bar-mode 0)
(tool-bar-mode 0)
(menu-bar-mode 0)
(setq inhibit-startup-message t)
```

Makes the parenthesis shine

```
(use-package rainbow-delimiters :ensure t)
```

Doesn't ask to confirm everytime

```
  (setq confirm-kill-processes nil)
  (setq-default truncate-lines t)
  (setq-default fill-column 80)

;(setq-default cursor-type 'square)
(defalias 'yes-or-no-p 'y-or-n-p)
```

2.2 Theme and transparency

Loading theme and setting modeline background color.

```
;  (use-package cherry-blossom-theme
;    :config
;    (load-theme 'cherry-blossom t)
;    :ensure t)

  (use-package dracula-theme ;almost-mono-themes
   :config
   (load-theme 'dracula t) ; 'almost-mono-black t)
   (let ((line (face-attribute 'mode-line :underline)))
     (set-face-attribute 'mode-line          nil :
        overline   line)
```

```
    (set-face-attribute 'mode-line-inactive nil :
       overline  line)
    (set-face-attribute 'mode-line-inactive nil :
       underline  line)
    (set-face-attribute 'mode-line          nil :box
             nil)
    (set-face-attribute 'mode-line-inactive nil :box
             nil)
    (set-face-attribute 'mode-line-inactive nil :
       background "#212121"))
                                     :ensure t)
```

Enabling transparency

```
(set-frame-parameter (selected-frame) 'alpha '(95 95))
(add-to-list 'default-frame-alist '(alpha 95 95))
```

2.3 Font

Took from here: `https://emacs.stackexchange.com/q/45895`

```
(set-frame-font "Liberation␣Mono-12:antialias=none")
```

```
(use-package default-text-scale
 :ensure t
 :hook (after-init . default-text-scale-mode))
```

```
(set-language-environment "UTF-8")
(global-prettify-symbols-mode t)
```

```
(prefer-coding-system 'utf-8)
```

2.4 Modeline

Nice and simple.

```
(use-package moody
  :config
  (setq x-underline-at-descent-line t)
  (moody-replace-mode-line-buffer-identification)
  (moody-replace-vc-mode)
  :ensure t)
```

2.5 Line Number

```
;       (global-display-line-numbers-mode)
;       (setq display-line-numbers-type 'relative)
```

2.6 Font lock

Disables syntax hilight from startup

```
(global-set-key (kbd "C-x C-l") 'font-lock-mode)
```

3 Navigation

3.1 Evil Mode

```
(add-to-list 'load-path "~/.emacs.d/evil")
(require 'evil)
(evil-mode 1)
```

3.2 Smart Parents

Creates pairs of parenthesis in a smart way

```
(use-package smartparens
  :ensure t
  :config
  (sp-use-paredit-bindings)
  (add-hook 'prog-mode-hook #'smartparens-mode)
  (sp-pair "{" nil :post-handlers '(("||\n[i]" "RET")))
     )

(setq-default indent-tabs-mode nil)
(setq-default tab-width 4)
```

3.3 Scrolling

```
(setq kill-buffer-query-functions
  (remq 'process-kill-buffer-query-function
   kill-buffer-query-functions))
;; mouse scrolls very slowly
```

```
(setq confirm-kill-processes nil)
(setq scroll-step              1
scroll-conservatively  10000
mouse-wheel-scroll-amount '(1 ((shift) . 1))
mouse-wheel-progressive-speed nil
mouse-wheel-follow-mouse 't)
```

3.4 Inserting new line

Add a new line below the current line

```
(defun insert-new-line-below ()
  (interactive)
  (let ((oldpos (point)))
    (end-of-line)
    (newline-and-indent)))
(global-set-key (kbd "C-o") 'insert-new-line-below)
```

4 Code

4.1 Ivy

```
(use-package ivy
 :ensure t
 :config(ivy-mode 1))
```

4.2 Swiper

```
(use-package swiper
  :ensure t
  :config
  (progn
    (ivy-mode 1)
    (setq ivy-use-virtual-buffers t)
    (global-set-key "\C-s" 'swiper)))
```

4.3 Try

```
(use-package try
  :ensure t
  :config
  (progn  (global-set-key (kbd "C-x b") '
     ivy-switch-buffer)))
(setq ivy-use-virtual-buffers t)
(setq ivy-display-style 'fancy)
```

4.4 Which-key

```
(use-package which-key
  :ensure t
  :config (which-key-mode))
```

4.5 Language specifics

4.5.1 Julia

```
(use-package julia-mode
  :ensure t
  :hook ((julia-mode) . jl))
```

5 Org-mode

5.1 UI

```
; this allows to use some shortcuts .. begins_src..
(require 'org-tempo)
; enabling syntax hilight
(add-hook 'org-mode-hook 'font-lock-mode)

(add-to-list 'org-modules 'org-tempo t)

(use-package org-bullets
  :hook (org-mode . org-bullets-mode)
  :custom
  (org-bullets-bullet-list '("   " "   " "   " "   " "
        " "   " "   ")))
  (setq org-ellipsis "   ")

(font-lock-add-keywords
 'org-mode
 '(("^[[:space:]]*\\(-\\) "
    (0 (prog1 () (compose-region (match-beginning 1) (
        match-end 1) "   "))))))

; (setq org-src-tab-acts-natively t)
(setq org-src-window-setup 'current-window)
(add-to-list 'org-structure-template-alist
             '("el" . "src emacs-lisp"))
```

5.2 Tasks management

```
(add-hook 'org-mode-hook 'auto-fill-mode)
(setq-default fill-column 79)
(setq org-todo-keywords '((sequence "TODO(t)" "NEXT(n)"
    "|" "DONE(d!)" "DROP(x!)"))
      org-log-into-drawer t)

(defun org-file-path (filename)
  " Return the absolute address of an org file, give
     its relative name"
  (concat (file-name-as-directory org-directory)
     filename))

(setq org-index-file (org-file-path "todo.org"))
(setq org-archive-location
      (concat (org-file-path "done.org") "::* From %s")
         )

;; copy the content out of the archive.org file and
   yank in the inbox.org
(setq org-agenda-files (list org-index-file))
                                              ; mark a todo
                                                as done and
```

```
                                    move it to
                                    an
                                    appropriate
                                    place in the
                                     archive.
(defun hrs/mark-done-and-archive ()
  "Mark the state of an org-mode item as DONE and
     archive it."
  (interactive)
  (org-todo 'done)
  (org-archive-subtree))
(global-set-key (kbd "C-c C-x C-s") 'hrs/
   mark-done-and-archive)
(setq org-log-done 'time)
```

5.3 Capturing Tasks

```
(setq org-capture-templates
      '(("t" "Todo"
         entry
         (file+headline org-index-file "Inbox")
         "* TODO %?\n")))
(setq org-refile-use-outline-path t)
(setq org-outline-path-complete-in-steps nil)
(define-key global-map "\C-cc" 'org-capture)
```

5.4 Displaying inline images

The joy of programming = `https://joy.pm/post/2017-09-17-a_graphviz_primer/nn`

```
(defun my/fix-inline-images ()
(when org-inline-image-overlays
(org-redisplay-inline-images)))
(add-hook 'org-babel-after-execute-hook 'my/
   fix-inline-images)
(setq-default org-image-actual-width 620)
(global-set-key (kbd "C-c i") 'org-toggle-inline-images
   )
```

5.5 Exporting with org-mode

Makes UTF-8 symbols appears in buffer I use it for editing Latex

```
(add-hook 'org-mode-hook
(lambda () (org-toggle-pretty-entities)))
;; Opening pdfs
(add-to-list 'org-file-apps '("\\.pdf" . "xreader␣%s"))
(global-set-key (kbd "C-x␣p") 'org-latex-export-to-pdf)
```

6 Latex

```
(use-package auctex
  :ensure t
  :hook ((latex-mode LaTeX-mode) . tex)
  :config
  (font-lock-mode)
  (add-to-list 'font-latex-math-environments "dmath"))

(add-hook 'LaTeX-mode-hook 'TeX-mode)
(add-hook 'LaTeX-mode-hook 'font-lock-mode)


(add-hook 'LaTeX-mode-hook 'visual-line-mode)
(add-hook 'LaTeX-mode-hook 'flyspell-mode)
(add-hook 'LaTeX-mode-hook 'LaTeX-math-mode)

(add-hook 'LaTeX-mode-hook 'turn-on-reftex)

(setq reftex-plug-into-AUCTeX t)

(setq TeX-auto-save t)
(setq TeX-parse-self t)
(setq TeX-save-query t)
(setq-default TeX-master nil)
(setq TeX-PDF-mode t)
                                                    ; (add-hook '
                                                       LateX-mode-hook
                                                        (lambda ()
                                                       (
                                                       latex-preview-pane-mode
                                                       )))
                                                    ; (
                                                       global-set-key
                                                        (kbd "C-x l
                                                        ") '
                                                       latex-preview-pane-mode
                                                       )
(global-set-key (kbd "C-x l ") 'pdflatex)
(add-to-list 'org-latex-packages-alist '("" "listings"
   nil))
(setq org-latex-listings t)
(setq org-latex-listings-options '(("breaklines" "true"
   )))
```

7 Dired

```
(use-package dired-sidebar
  :ensure t
  :config
  (global-set-key (kbd "C-x C-n") '
     dired-sidebar-toggle-sidebar)
  (add-hook 'dired-mode-hook 'font-lock-mode))
```

7.1 Definying default applications open certain types of file.

```
(use-package dired-open
  :ensure t
  :config
  (setq dired-open-extensions
        '(("doc" . "openoffice4")
          ("docx" . "openoffice4")
          ("xopp" . "xournalpp")
          ("gif" . "mirage")
          ("jpeg" ."mirage")
          ("jpg" . "mirage")
          ("png" . "mirage")
          ("mkv" . "mpv")
```

```
            ("avi" . "mpv")
            ("mov" . "mpv")
            ("mp3" . "mpv")
            ("mp4" . "mpv")
            ("pdf" . "xreader")
            ("webm" . "mpv"))))
```

7.2 Hide dotfiles and extra information (aka ownership and such)

```
(use-package dired-hide-dotfiles
  :ensure t
  :config
  (dired-hide-dotfiles-mode)
  (define-key dired-mode-map "." '
     dired-hide-dotfiles-mode))

(setq-default dired-listing-switches "-lhvA")
(add-hook 'dired-mode-hook (lambda () (
   dired-hide-details-mode 1)))
;; Taken from here: https://emacs.stackexchange.com/
   questions/13080/reloading-directory-local-variables
   /13096#13096
(defun my-reload-dir-locals-for-current-buffer ()
  "reload␣dir␣locals␣for␣the␣current␣buffer"
  (interactivye)
  (let ((enable-local-variables :all))
    (hack-dir-local-variables-non-file-buffer)))
(defun
   my-reload-dir-locals-for-all-buffer-in-this-directory
    ()
  "For␣every␣buffer␣with␣the␣same␣`default-directory`␣
     as␣the
current␣buffer's,␣reload␣dir-locals."
  (interactive)
  (let ((dir default-directory))
    (dolist (buffer (buffer-list))
      (with-current-buffer buffer
        (when (equal default-directory dir))
        (my-reload-dir-locals-for-current-buffer)))))
```

8 Auto-completation

```
(use-package auto-complete
 :ensure t
 :init
 (global-auto-complete-mode))
```

global

9 Windows

9.1 Ace Window

```
(use-package ace-window
  :ensure t
  :init
  (progn
    (global-set-key [remap other-window] 'ace-window)
    (custom-set-faces
      '(aw-leading-char-face
        ((t (:inherit ace-jump-face-foreground :height
            2.0)))))))
```

9.2 Terminal

```
(use-package multi-term
 :ensure t
 :config  (setq multi-term-program "/bin/bash")
 (progn
   (global-set-key (kbd "C-x t") 'multi-term)))
```

10 External

Elcord

```
(use-package elcord
  :ensure t
  :config
  (setq elcord-refresh-rate 5))
  ;(elcord-mode))
(global-set-key (kbd "C-c d") 'elcord-mode)
```