Emacs Config

Jacob Stokes

Contents

1 Setup

1.1 Configure package sources

1.1.1 Require Package

```
(require 'package)
(setq package-archives
      '(("melpa" . "https://melpa.org/packages/")
("gnu" . "https://elpa.gnu.org/packages/")
("org" . "http://orgmode.org/elpa/")))
(package-initialize)
```

1.1.2 Use Package

1.2 Appearence

1.2.1 Turn off menu bar and tool bar.

```
(menu-bar-mode -1)
(tool-bar-mode -1)
(scroll-bar-mode -1)
```

1.2.2 Add in dashboard

```
(use-package dashboard
  :ensure t
  :config
  (setq dashboard-items '((recents . 10)
  (bookmarks . 10)
  (agenda . 10))
dashboard-center-content t
dashboard-startup-banner 4
dashboard-show-shortcuts nil
dashboard-set-footer nil
dashboard-init-info "")
  (dashboard-setup-startup-hook)
  )
```

1.2.3 Doom Themes

Doom themes. Various options disabled for the time being.

```
(use-package doom-themes
  :ensure t
  :config
  ;; Global settings (defaults)
  ;; (setq doom-themes-enable-bold t    ; if nil, bold is universally disabled
  ;;     doom-themes-enable-italic t) ; if nil, italics is universally disabled
  (load-theme 'doom-vibrant t)
  (doom-themes-org-config))

;; Enable flashing mode-line on errors
;; (doom-themes-visual-bell-config)

;; Enable custom neotree theme (all-the-icons must be installed!)
;;(doom-themes-neotree-config)
;; or for treemacs users
;; (setq doom-themes-treemacs-theme "doom-colors") ; use the colorful treemacs theme
 (doom-themes-treemacs-config)
```

1.2.4 Disable splash screen.

Does what is says on the tin. Allows the Dashboard to be loaded.

```
(setq inhibit-splash-screen t)
```

1.2.5 All the Icons

```
(use-package all-the-icons)
```

1.3 Key Bindings

1.3.1 General bindings - to sort

```
(global-set-key (kbd "<f5>") 'restart-emacs)
(global-set-key (kbd "<f6>") 'olivetti-mode)
(global-set-key (kbd "<f12>") 'open-file-fast)
(global-set-key (kbd "<M-f12>") 'package-install)
```

1.3.2 Kill other buffer.

```
(defun other-window-kill-buffer ()
  "Kill the buffer in the other window"
  (interactive)
  ;; Window selection is used because point goes to a different window
  ;; if more than 2 windows are present
  (let ((win-curr (selected-window))
(win-other (next-window)))
    (select-window win-other)
    (kill-this-buffer)
    (select-window win-curr)))

(global-set-key (kbd "C-x K") 'other-window-kill-buffer)
```

1.3.3 Kill all buffers matching string, no prompt

```
(defun is-help-buffer (buffer)
  (let ((name (buffer-name buffer)))
    (and (= ?* (aref name 0))
 (string-match "Help" name))))

(defun kill-help-buffers ()
  (interactive)
  (cl-loop for buffer being the buffers
do (and (is-help-buffer buffer) (kill-buffer buffer))))

(global-set-key (kbd "C-x C-k h") 'kill-help-buffers)
```

1.3.4 Delete shortcuts

```
(global-set-key (kbd "C-;") 'delete-backward-char)
```

1.3.5 Comment out

1. Elisp

   ```
   (global-set-key (kbd "C-c C-;") 'elisp-comment-out)
   ```

1.3.6 Rebind org-agenda-files

```
;;(global-set-key (kbd "C-x ,") 'org-cycle-agenda-files)
```

1.3.7 Windmove

```
;; (global-set-key (kbd "C-,")  'windmove-left)
;; (global-set-key (kbd "C-. <right>") 'windmove-right)
```

1.4 Navigation

1.4.1 Open-file-fast

```
(defun open-file-fast ()
  "Prompt to open a file from bookmark ‘bookmark-bmenu-list’.
This command is similar to ‘bookmark-jump’, but use ‘ido-mode’ interface, and ignore c

URL ‘http://ergoemacs.org/emacs/emacs_hotkey_open_file_fast.html’
Version 2019-02-26"
```

```
  (interactive)
  (require 'bookmark)
  (bookmark-maybe-load-default-file)
  (let (($this-bookmark
 (ido-completing-read "Open bookmark:" (mapcar (lambda ($x) (car $x)) bookmark-alist))
    (find-file (bookmark-get-filename $this-bookmark))
    ;; (bookmark-jump $this-bookmark)
    ))
```

1.4.2 Treemacs

```
(use-package treemacs
  :ensure t
  :defer t
  :init
  (with-eval-after-load 'winum
    (define-key winum-keymap (kbd "M-0") #'treemacs-select-window))
  :config
  (progn
    (setq treemacs-collapse-dirs                 (if treemacs-python-executable 3 0)
  treemacs-deferred-git-apply-delay      0.5
  treemacs-directory-name-transformer    #'identity
  treemacs-display-in-side-window        t
  treemacs-eldoc-display                 t
  treemacs-file-event-delay              5000
  treemacs-file-extension-regex          treemacs-last-period-regex-value
  treemacs-file-follow-delay             0.2
  treemacs-file-name-transformer         #'identity
  treemacs-follow-after-init             t
  treemacs-git-command-pipe              ""
  treemacs-goto-tag-strategy             'refetch-index
  treemacs-indentation                   2
  treemacs-indentation-string            " "
  treemacs-is-never-other-window         t
  treemacs-max-git-entries               5000
  treemacs-missing-project-action        'ask
  treemacs-move-forward-on-expand        nil
  treemacs-no-png-images                 nil
```

```
  treemacs-no-delete-other-windows       t
  treemacs-project-follow-cleanup        nil
  treemacs-persist-file                  (expand-file-name ".cache/treemacs-persist" us
  treemacs-position                      'left
  treemacs-recenter-distance             0.1
  treemacs-recenter-after-file-follow    nil
  treemacs-recenter-after-tag-follow     nil
  treemacs-recenter-after-project-jump   'always
  treemacs-recenter-after-project-expand 'on-distance
  treemacs-show-cursor                   nil
  treemacs-show-hidden-files             t
  treemacs-silent-filewatch              nil
  treemacs-silent-refresh                nil
  treemacs-sorting                       'alphabetic-asc
  treemacs-space-between-root-nodes      t
  treemacs-tag-follow-cleanup            t
  treemacs-tag-follow-delay              1.5
  treemacs-user-mode-line-format         nil
  treemacs-user-header-line-format       nil
  treemacs-width                         35)

    ;; The default width and height of the icons is 22 pixels. If you are
    ;; using a Hi-DPI display, uncomment this to double the icon size.
    ;;(treemacs-resize-icons 44)

    (treemacs-follow-mode t)
    (treemacs-filewatch-mode t)
    (treemacs-fringe-indicator-mode t)
    (pcase (cons (not (null (executable-find "git")))
 (not (null treemacs-python-executable)))
      (`(t . t)
       (treemacs-git-mode 'deferred))
      (`(t . _)
       (treemacs-git-mode 'simple))))
  :bind
  (:map global-map
("M-0"       . treemacs-select-window)
("C-x t 1"   . treemacs-delete-other-windows)
("C-x t t"   . treemacs)
("C-x t B"   . treemacs-bookmark)
```

```
("C-x t C-t" . treemacs-find-file)
("C-x t M-t" . treemacs-find-tag)))

(use-package treemacs-evil
  :after treemacs evil
  :ensure t)

(use-package treemacs-projectile
  :after treemacs projectile
  :ensure t)

(use-package treemacs-icons-dired
  :after treemacs dired
  :ensure t
  :config (treemacs-icons-dired-mode))

(use-package treemacs-magit
  :after treemacs magit
  :ensure t)

(use-package treemacs-persp ;;treemacs-persective if you use perspective.el vs. persp-m
  :after treemacs persp-mode ;;or perspective vs. persp-mode
  :ensure t
  :config (treemacs-set-scope-type 'Perspectives))
```

1.4.3 Windmove

```
;;(windmove-default-keybindings 'meta)

(when (fboundp 'windmove-default-keybindings)
  (windmove-default-keybindings))
```

1.4.4 TODO Ace-window

```
  (use-package ace-window
    :ensure t)
```

```
(global-set-key (kbd "M-o") 'ace-window)
```

1.5 Registers

1.5.1 File Registers

```
(set-register ?s (cons 'file "~/.emacs.d/settings.org"))
```

2 Writing

2.1 Spelling and Grammar

2.1.1 Dubcaps mode.

```
(defun dcaps-to-scaps ()
  "Convert word in DOuble CApitals to Single Capitals."
  (interactive)
  (and (= ?w (char-syntax (char-before)))
       (save-excursion
 (and (if (called-interactively-p)
  (skip-syntax-backward "w")
(= -3 (skip-syntax-backward "w")))
      (let (case-fold-search)
(looking-at "\\b[[:upper:]]\\{2\\}[[:lower:]]"))
      (capitalize-word 1)))))

(add-hook 'post-self-insert-hook #'dcaps-to-scaps nil 'local)


;; Dubcaps mode

(define-minor-mode dubcaps-mode
  "Toggle `dubcaps-mode'.  Converts words in DOuble CApitals to
Single Capitals as you type."
  :init-value nil
  :lighter (" DC")
  (if dubcaps-mode
      (add-hook 'post-self-insert-hook #'dcaps-to-scaps nil 'local)
```

```
    (remove-hook 'post-self-insert-hook #'dcaps-to-scaps 'local)))

(add-hook 'org-mode-hook 'dubcaps-mode)
(add-hook 'markdown-mode-hook 'dubcaps-mode)
```

2.1.2 Flyspell

```
 (use-package flyspell-correct
   :ensure t
   :after flyspell
   :bind (:map flyspell-mode-map ("C-'" . flyspell-correct-wrapper)))

 (use-package flyspell-correct-ivy
  :ensure t
  :after flyspell-correct)

 ;; Hook to org mode
;; (add-hook 'org-mode-hook 'flyspell-mode)
```

2.2 Citations

2.2.1 Reftex

1. Enable

```
    (use-package reftex
      :ensure t
      :config
      (setq bibtex-completion-bibliography '("~/org-ref-test/bibs/Testing2.bib")
    bibtex-completion-format-citation-functions '((org-mode      . bibtex-completion-f
          (latex-mode    . bibtex-completion-format-citation-cite)
          (markdown-mode . bibtex-completion-format-citation-pandoc-citeproc)
          (default       . bibtex-completion-format-citation-default))
    ivy-bibtex-default-action 'ivy-bibtex-insert-citation
    bibtex-completion-pdf-field "File")
```

```
  :bind (("C-C i" . ivy-bibtex)))


;; (setq bibtex-completion-bibliography
;;       '("~/org-ref-test/bibs/Testing2.bib"
;;         ))
;;
;;
;; (setq bibtex-completion-format-citation-functions
;;   '((org-mode      . bibtex-completion-format-citation-pandoc-citeproc)
;;     (latex-mode    . bibtex-completion-format-citation-cite)
;;     (markdown-mode . bibtex-completion-format-citation-pandoc-citeproc)
;;     (default       . bibtex-completion-format-citation-default)))



;; (setq ivy-bibtex-default-action 'ivy-bibtex-insert-citation)
;;
;; (global-set-key (kbd "C-c i") 'ivy-bibtex)
;;
;; (setq bibtex-completion-pdf-field "File")
;;
```

2. Functions

   (a) **TODO** Misc

```
(defun bibtex-completion-open-pdf-of-entry-at-point ()
  (interactive)
  (save-excursion
    (bibtex-beginning-of-entry)
    (when (looking-at bibtex-entry-maybe-empty-head)
      (bibtex-completion-open-pdf (bibtex-key-in-head)))))


(defun my/print-reference-title ()
  "Print the title to the reference at point in the minibuffer."
  (interactive)
```

```
(message
 (assoc-default "title"
(bibtex-completion-get-entry
 (org-ref-get-bibtex-key-under-cursor)))))
```

2.3 Templating

2.3.1 Yasnippets

1. Setup

```
(use-package yasnippet
  :ensure t
  :bind ("C-c 8" . yas-insert-snippet)
  :config
  (yas-global-mode 1))

;; (yas-global-mode 1)

;; (global-set-key (kbd "C-c 8") 'yas-insert-snippet)
```

3 Major modes

3.1 Org-mode

3.1.1 Required

1. Enable Export as markdown

```
(eval-after-load "org"
  '(require 'ox-md nil t))
```

2. Pandoc-mode Org-mode

```
(add-hook 'org-mode-hook 'pandoc-mode)
(setq org-pandoc-options-for-latex-pdf '((pdf-engine . "pdflatex")))
```

3.1.2 Org-babel

1. Load Languages

   ```
   (org-babel-do-load-languages
    'org-babel-load-languages
    '((python . t)))
   ```

2. Disable prompt

   ```
   (setq org-confirm-babel-evaluate nil)
   ```

3. Language Settings

   (a) Python Set org-babel python command to python3

   ```
   (setq org-babel-python-command "python3")
   ```

3.1.3 Tweaks

1. Org-Identation

   ```
   (setq org-startup-indented t)
   ```

2. Better org-refile

   ```
   (setq org-refile-targets '((nil :maxlevel . 9)
   (org-agenda-files :maxlevel . 9)))
   (setq org-outline-path-complete-in-steps nil)         ; Refile in a single go
   (setq org-refile-use-outline-path t)                  ; Show full paths for refili
   ```

3. SRC-blocks behaviour

```
(defvar org-blocks-hidden nil)

(defun org-toggle-blocks
()
  (interactive)
  (if org-blocks-hidden
      (org-show-block-all)
    (org-hide-block-all))
  (setq-local org-blocks-hidden (not org-blocks-hidden)))

(add-hook 'org-mode-hook 'org-toggle-blocks)

(define-key org-mode-map (kbd "C-c t") 'org-toggle-blocks)


(setq org-src-tab-acts-natively t)
```

3.1.4 Org-Journal

```
(use-package org-journal
  :ensure t
  :config
  (setq org-journal-dir "~/work/journal/"))
```

3.1.5 Org-Super-Agenda

```
(use-package org-super-agenda
  :ensure t
  :config
 (setq org-super-agenda-groups '(
   (:name "Waiting"
       :tag "shop"))))
```

3.1.6 Org-agenda

Commented out last part is for recusrively working through a directory. This clashed with dashboard and recentf, as these files were being opened at startup. One will do.

```
(setq org-agenda-files (apply 'append
     (mapcar
      (lambda (directory)
(directory-files-recursively
 directory org-agenda-file-regexp))
      '("~/work/agenda/"))))
 (define-key global-map "\C-ca" 'org-agenda)
(setq org-log-done t)
```

3.2 Environments

3.2.1 Python

1. Elpy

   ```
   (use-package elpy
     :ensure t
     :init (elpy-enable)
     :config (setq elpy-rpc-python-command "python3")
     )
   ```

3.3 Shell

3.3.1 Open Shell

```
(global-set-key (kbd "<f7>") 'shell)
```

4 Minor Modes/Tools

4.1 Ivy, Counsel, Swiper

4.1.1 Counsel

```
(use-package counsel :ensure t
  :after ivy
  :bind (("M-x" . 'counsel-M-x)
 ("C-x C-f" . 'counsel-find-file)
 ("<f1> f" . 'counsel-describe-function)
 ("<f1> v" . 'counsel-describe-variable)
 ("<f1> o" . 'counsel-describe-symbol)
 ("<f1> l" . 'counsel-find-library)
 ("<f2> i" . 'counsel-info-lookup-symbol)
 ("<f2> u" . 'counsel-unicode-char)
 ("C-c g" . 'counsel-git)
 ("C-c j" . 'counsel-git-grep)
 ("C-c k" . 'counsel-ag)
 ("C-S-o" . 'counsel-rhythmbox)
 :map minibuffer-local-map ("C-r" . 'counsel-minibuffer-history)))
```

4.1.2 Ivy

```
(use-package ivy :ensure t
  :init (setq ivy-use-virtual-buffers t
  enable-recursive-minibuffers t)
 :demand  :config (ivy-mode 1)
 :bind (("C-c C-r" . ivy-resume)))
```

4.1.3 Swiper

```
(use-package swiper :ensure t
  :after ivy
  :bind (("C-s" . swiper)
 ("C-r" . swiper)))
```

4.2 Pdf-tools

```
(use-package pdf-tools
  :ensure t
  :config (pdf-tools-install))
```

4.3 Centaur-tabs

```
(use-package centaur-tabs
  :ensure t
  :demand
  :config
  (centaur-tabs-mode t)
  :bind
  ("M-n" . centaur-tabs-backward)
  ("M-p" . centaur-tabs-forward))


(setq centaur-tabs-set-icons t)
(setq centaur-tabs-plain-icons t)
```

4.4 projectile

4.4.1 Main

```
  (use-package projectile
    :ensure t
    :config
    (define-key projectile-mode-map (kbd "C-c p") 'projectile-command-map)
    (projectile-mode +1)
    (setq projectile-completion-system 'ivy)
)
```

4.4.2 Counsel Projectile

```
(counsel-projectile-mode t)
```

5 Testing

5.1 Switch Window

```
  ;; (use-package switch-window
   ;; :ensure t
   ;; :bind (("C-x o" . switch-window)
;;     ("C-x 1" . switch-window-then-maximize)
;;     ("C-x 2" . switch-window-then-split-below)
;;     ("C-x 3" . switch-window-then-split-right)
;;     ("C-x 0" . switch-window-then-delete)
;;     ("C-x 4 d" . switch-window-then-dired)
;;     ("C-x 4 f" . switch-window-then-find-file)
;;     ("C-x 4 m" . switch-window-then-compose-mail)
;;     ("C-x 4 r" . switch-window-then-find-file-read-only)
;;     ("C-x 4 C-f" . switch-window-then-find-file)
;;     ("C-x 4 C-o" . switch-window-then-find-file-read-only)
;;     ("C-x 4 C-f" . switch-window-then-find-file)
;;     ("C-x 4 C-o" . switch-window-then-display-buffer)
;;     ("C-x 4 0" . switch-window-then-kill-buffer)))
```

5.2 Switch-window Quick

```
(defun quick-switch-window ()
   (interactive)
  (switch-window))
(global-set-key (kbd "C-?") 'quick-switch-window)


(setq switch-window-shortcut-style 'qwerty)
```