Jakob Klemm - Emacs

Jakob Klemm

May 29, 2021

Contents

1 Setup

Base settings & setup.

1.1 Packages

Setup `use-package` and quelpa, since `straight` is most likely still broken.

```
(require 'package) ;; Emacs builtin

;; set package.el repositories
(setq package-archives
      '(
("org" . "https://orgmode.org/elpa/")
("gnu" . "https://elpa.gnu.org/packages/")
("melpa" . "https://melpa.org/packages/")
))

;; initialize built-in package management
(package-initialize)

;; update packages list if we are on a new install
(unless package-archive-contents
  (package-refresh-contents))

;; a list of pkgs to programmatically install
;; ensure installed via package.el
(setq my-package-list '(use-package))

;; programmatically install/ensure installed
;; pkgs in your personal list
(dolist (package my-package-list)
  (unless (package-installed-p package)
    (package-install package)))

(use-package quelpa-use-package
  :ensure t
  )
```

1.2 Defaults

Some default settings, partially copied from Harry R. Schwartz.

```
(setq gc-cons-threshold 2000000000000)
```

```
;; Treat CamelCaseSubWords as separate words in every programming
;; mode.
(add-hook 'prog-mode-hook 'subword-mode)

;; Don't assume that sentences should have two spaces after
;; periods.
(setq sentence-end-double-space nil)

;; Turn on transient-mark-mode.
(transient-mark-mode t)

;; Auto wrap text
(auto-fill-mode t)

;; selected text and start inserting your typed text.
(delete-selection-mode t)

;; If you save a file that doesn't end with a newline, automatically
;; append one.
(setq require-final-newline t)

;; Visually indicate matching pairs of parentheses.
(show-paren-mode t)
(setq show-paren-delay 0.0)

;; When you perform a problematic operation, flash the screen instead
;; of ringing the terminal bell.
(setq ring-bell-function 'ignore)
(setq visible-bell nil)

;; Don't ask `yes/no?', ask `y/n?'.
(fset 'yes-or-no-p 'y-or-n-p)

;; Ask if you're sure that you want to close Emacs.
(setq confirm-kill-emacs 'y-or-n-p)

;; Don't present the usual startup message, and clear the scratch buffer.
(setq inhibit-startup-message t)
(setq initial-scratch-message nil)
```

```
;; When something changes a file, automatically refresh the buffer
;; containing that file so they can’t get out of sync.
(global-auto-revert-mode t)

;; Move everything to trash first
(setq delete-by-moving-to-trash t)

;; No reason to use any other type. Might be disabled dependant on the
;; current mode.
(setq display-line-numbers-type ’relative)

;; Use tabs for everything (https://youtu.be/SsoOG6ZeyUI)
(setq indent-tabs-mode t)
(setq indent-line-function ’insert-tab)

;; Launch emacs in fullscreen mode
(add-to-list ’default-frame-alist ’(fullscreen . maximized))

;; Save the location within a file.
(save-place-mode t)

;; Set always to UTF-8, only display in bar if not UTF-8
(set-language-environment "UTF-8")

;; Menu bar
(tool-bar-mode 0)
(menu-bar-mode 0)
(scroll-bar-mode -1)

;; Minibuffer
(set-window-scroll-bars (minibuffer-window) nil nil)

;; Use smoth scrolling
(setq scroll-conservatively 100)

;; Highlight the current line
(global-hl-line-mode)

;; Hide the modeline
(setq mode-line-format nil)
```

```
;; Error handling
;; (setq warning-minimum-level :emergency)

;; Make it affect all buffers.
(setq-default mode-line-format nil)

;; Line wrap mode
(add-hook 'text-mode-hook 'auto-fill-mode)
(add-hook 'gfm-mode-hook 'auto-fill-mode)
(add-hook 'org-mode-hook 'auto-fill-mode)

(setq backup-directory-alist
      `((".*" . ,temporary-file-directory)))
(setq auto-save-file-name-transforms
      `((".*" ,temporary-file-directory t)))

(setq byte-compile-warnings '(cl-functions))

;; Reopen files after restart
(desktop-save-mode 1)
(savehist-mode 1)
(add-to-list 'savehist-additional-variables 'kill-ring)
```

1.3 Common

Some common libraries as general dependencies.

```
(use-package async
  :ensure t
  )
(use-package aio
  :ensure t
  )
(use-package cl-lib
  :ensure t
  )
(use-package s
  :ensure t
  )
```

```
(use-package dash
  :ensure t
  )
```

1.4 Extras

Add `resources/` to the path. Add it both to the load-path as well as custom-theme-load-path.

```
(add-to-list 'load-path "~/.emacs.d/resources/")
(add-to-list 'custom-theme-load-path "~/.emacs.d/resources/")
```

1.5 Files

Install `no-littering` to handle all temp files.

```
(use-package no-littering
  :ensure t
  )
```

2 Design

Anything related to design and looks.

2.1 Theme

Use the `jeykey-dark` theme, which was generated using themer. Also highlight the current line & set the point color.

```
(load-theme 'jeykey-dark t)
(set-cursor-color "#D069D6")
```

2.2 Font

Set the default font & functions for changing the font size.

```
(setq hrs/default-fixed-font "Iosevka")
(setq hrs/default-fixed-font-size 90)
(setq hrs/current-fixed-font-size hrs/default-fixed-font-size)
(set-face-attribute 'default nil
    :family hrs/default-fixed-font
    :height hrs/current-fixed-font-size)
```

```
(set-face-attribute 'fixed-pitch nil
    :family hrs/default-fixed-font
    :height hrs/current-fixed-font-size)

(setq hrs/font-change-increment 1.1)

(defun hrs/set-font-size ()
  "Change default, fixed-pitch, and variable-pitch font sizes to match respective varia
  (set-face-attribute 'default nil
      :height hrs/current-fixed-font-size)
  (set-face-attribute 'fixed-pitch nil
      :height hrs/current-fixed-font-size)
  )

(defun hrs/reset-font-size ()
  "Revert font sizes back to defaults."
  (interactive)
  (setq hrs/current-fixed-font-size hrs/default-fixed-font-size)
  (hrs/set-font-size))

(defun hrs/increase-font-size ()
  "Increase current font sizes by a factor of `hrs/font-change-increment'."
  (interactive)
  (setq hrs/current-fixed-font-size
(ceiling (* hrs/current-fixed-font-size hrs/font-change-increment)))
  (hrs/set-font-size))

(defun hrs/decrease-font-size ()
  "Decrease current font sizes by a factor of `hrs/font-change-increment', down to a mi
  (interactive)
  (setq hrs/current-fixed-font-size
(max 1
     (floor (/ hrs/current-fixed-font-size hrs/font-change-increment))))
  (hrs/set-font-size))

(define-key global-map (kbd "C-)") 'hrs/reset-font-size)
(define-key global-map (kbd "C-+") 'hrs/increase-font-size)
(define-key global-map (kbd "C-=") 'hrs/increase-font-size)
(define-key global-map (kbd "C-_") 'hrs/decrease-font-size)
(define-key global-map (kbd "C--") 'hrs/decrease-font-size)
```

```
(hrs/reset-font-size)
```

2.3 Margins

Use centered text everywhere, except for excluded buffers.

```
(defcustom perfect-margin-ignore-regexps
  '("^minibuf" "^[*]" "Minibuf" "[*]" "magit" "mu4e")
  "List of strings to determine if window is ignored.
Each string is used as regular expression to match the window buffer name."
  :group 'perfect-margin)

(defcustom perfect-margin-ignore-filters
  '(window-minibuffer-p)
  "List of functions to determine if window is ignored.
Each function is called with window as its sole arguemnt, returning a non-nil value inc
  :group 'perfect-margin)

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

2.4 Modeline

Use feebeline as a *in-minibuffer-modeline*.

```
(use-package    feebleline
  :ensure t
  :config       (setq feebleline-msg-functions
      '((feebleline-line-number         :post "" :fmt "%5s")
(feebleline-column-number       :pre ":" :fmt "%-2s")
(feebleline-file-directory      :face feebleline-dir-face :post "")
(feebleline-file-or-buffer-name :face font-lock-keyword-face :post "")
(feebleline-file-modified-star  :face font-lock-warning-face :post "")
(feebleline-git-branch          :face feebleline-git-face :pre " ")
(feebleline-project-name        :align right)
((lambda () (format-time-string "%H:%M")) :align right)
)
```

```
    )
  (feebleline-mode 1)
  )
```

2.5 Rainbow

Install rainbow-delimiters & enable it for programming & org-mode.

```
(use-package rainbow-delimiters
  :ensure t
  :config
  (add-hook 'org-mode-hook #'rainbow-delimiters-mode)
  (add-hook 'prog-mode-hook #'rainbow-delimiters-mode)
  )
```

2.6 Icons

Install icons using `all-the-icons-install-fonts`

```
(use-package all-the-icons
  :ensure t
  )
```

2.7 Symbols

Enable prettify-symbols mode & set custom symbols for `org-mode`.

```
(setq-default prettify-symbols-alist '(("#+BEGIN_SRC" . "")
       ("#+END_SRC" . "")
       ("#+begin_src" . "")
       ("#+end_src" . "")
       ("#+TITLE:" . "")
       ("#+title:" . "")
       ("#+SUBTITLE:" . "")
       ("#+subtitle:" . "")
       ("#+DATE:" . "")
       ("#+date:" . "")
       ("#+PROPERTY:" . "")
       ("#+property:" . "")
       ("#+OPTIONS:" . "")
       ("#+options:" . "")
       ("#+LATEX_HEADER:" . "")
```

```
        ("#+latex_header:" . "")
        ("#+LATEX_CLASS:" . "")
        ("#+latexx_class:" . "")
        ("#+ATTR_LATEX:" . "")
        ("#+attr_latex:" . "")
        ("#+LATEX:" . "")
        ("#+latex:" . "")
        ("#+ATTR_HTML:" . "")
        ("#+attr_html:" . "")
        ("#+BEGIN_QUOTE:" . "")
        ("#+begin_quote:" . "")
        ("#+END_QUOTE:" . "")
        ("#+end_quote:" . "")
        ("#+CAPTION:" . "")
        ("#+caption:" . "")
        (":PROPERTIES:" . "")
        (":properties:" . "")
        ("#+AUTHOR:" . "A")
        ("#+author:" . "A")
        ("#+IMAGE:" . "I")
        ("#+image:" . "I")
        ("#+LANGUAGE:" . "L")
        ("#+language:" . "L")
        ))

(setq prettify-symbols-unprettify-at-point 'right-edge)
(add-hook 'org-mode-hook 'prettify-symbols-mode)
(global-prettify-symbols-mode 1)
```

3 Navigation

General settings & packages for navigating buffers and files.

3.1 Vertico

Partially copied from `SystemCrafters`.

```
(defun dw/minibuffer-backward-kill (arg)
  "When minibuffer is completing a file name delete up to parent
      folder, otherwise delete a word"
  (interactive "p")
  (if minibuffer-completing-file-name
      (if (string-match-p "/." (minibuffer-contents))
  (zap-up-to-char (- arg) ?/)
(delete-minibuffer-contents))
    (backward-kill-word arg)))

(use-package vertico
  :ensure t
  :custom-face
  (vertico-current ((t (:background "#3a3f5a"))))
  :bind (:map vertico-map
      ("C-j" . vertico-next)
      ("C-k" . vertico-previous)
      ("C-f" . vertico-exit)
      :map minibuffer-local-map
      ("C-l" . dw/minibuffer-backward-kill))
  :init
  (vertico-mode)

  ;; Optionally enable cycling for ‘vertico-next’ and ‘vertico-previous’.
  (setq vertico-cycle t)
  )
```

3.2 Improved completion

Corf & Orderless for improved completion in region.

```
(use-package corfu
  :ensure t
  :bind (:map corfu-map
      ("C-j" . corfu-next)
      ("C-k" . corfu-previous)
      ("C-f" . corfu-insert))
  :custom
  (corfu-cycle t)
  :config
  (corfu-global-mode))

(use-package orderless
  :ensure t
  :init
  (setq completion-styles '(orderless)
completion-category-defaults nil
completion-category-overrides '((file (styles .
  (partial-completion))))))
```

3.3 History

Save commands between restarts.

```
;; Persist history over Emacs restarts. Vertico sorts by history position.
(use-package savehist
  :init
  (savehist-mode)
  )
```

3.4 Search

Use consult for improved search.

```
(use-package consult
  :ensure t
  :bind (("C-s" . consult-line)
 ("M-s" . consult-imenu)
 :map minibuffer-local-map
 ("C-r" . consult-history))
  :custom
  (consult-project-root-function #'dw/get-project-root)
  (completion-in-region-function #'consult-completion-in-region)
  :config
  (consult-preview-at-point-mode)
  )
```

3.5 Annotations

Add *margin notes* in vertico buffers.

```
(use-package marginalia
  :after vertico
  :ensure t
  :custom
  (marginalia-annotators '(marginalia-annotators-heavy marginalia-annotators-light nil
  :init
  (marginalia-mode)
  )
```

3.6 Actions

Execute actions in the minibuffer.

```
(use-package embark
  :ensure t
  :bind (
 :map minibuffer-local-map
 ("C-d" . embark-act))
  :config
  ;; Show Embark actions via which-key
  (setq embark-action-indicator
(lambda (map) (which-key--show-keymap "Embark" map nil nil 'no-paging)
  #'which-key--hide-popup-ignore-command)
embark-become-indicator embark-action-indicator)
  )
```

3.7 Buffers

Use `bufler` to manage buffers.

```
(use-package bufler
  :ensure t
  :config
  (bufler-mode)
  )
```

3.8 Keys

Use `which-key` for `embark` suggestions.

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

3.9 Windows

Use `ace-windows` for quickly switching between multiple windows.

```
(use-package ace-window
  :ensure t
  :init
  (setq aw-scope 'frame ; limit to single frame
aw-keys '(?a ?o ?e ?u ?i ?d ?h ?t ?n)
)
  )
```

3.10 Miniframe

Instead of a complete posframe use `mini-frame` to display the minibuffer in the center of the screen.

```
(use-package mini-frame
  :ensure t
  :config
  (custom-set-variables
   '(mini-frame-show-parameters
     '((top . 0.4)
       (width . 0.5)
       (left . 0.5))))
  (mini-frame-mode t)
  )
```

3.11 Scrolling

Use `good-scroll` to move more easily through files.

```
(setq scroll-margin 8)

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

3.12 Helpers

Easy helper functions for quickly opening new buffers.

```
(defun hrs/split-window-below-and-switch ()
  "Split the window horizontally, then switch to the new pane."
  (interactive)
  (split-window-below)
  (balance-windows)
  (other-window 1)
  (bufler-switch-buffer)
  )

(defun hrs/split-window-right-and-switch ()
  "Split the window vertically, then switch to the new pane."
  (interactive)
  (split-window-right)
  (balance-windows)
  (other-window 1)
  (bufler-switch-buffer)
  )
```

3.13 Evil

Setup `evil` and all extra packages. Most binds are stored in the next section.

```
(use-package evil
  :ensure t
  :init
  (setq evil-move-beyond-eol t)
  (setq evil-want-keybinding nil)
  ;; (setq evil-want-integration t) ;; required by evil-collection
  ;; (setq evil-search-module 'evil-search)
  (setq evil-ex-complete-emacs-commands nil)
  (setq evil-vsplit-window-right t) ;; like vim's 'splitright'
  (setq evil-split-window-below t) ;; like vim's 'splitbelow'
  ;; (setq evil-shift-round nil)
  ;; (setq evil-want-C-u-scroll t)
  :config
  (evil-mode)
  (use-package evil-commentary
    :ensure t
    :bind (:map evil-normal-state-map
("gc" . evil-commentary)))

  (use-package evil-leader
    :ensure t
    :config
    (global-evil-leader-mode)
    (evil-leader/set-leader "<SPC>")
    )

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

(use-package general
  :ensure t
  )

(setq evil-emacs-state-modes nil)
(setq evil-insert-state-modes nil)
(setq evil-motion-state-modes nil)

(setq evil-normal-state-modes
      (append evil-emacs-state-modes
      evil-insert-state-modes
      evil-normal-state-modes
      evil-motion-state-modes)
      )
```

3.14 Binds

Define all binds in a custom mini-mode.

```
(global-set-key (kbd "C-x j") 'kill-buffer-and-window)
(global-set-key (kbd "C-x o") 'ace-window)
(global-set-key (kbd "<f5>") 'home-file)
(global-set-key (kbd "<f6>") 'projects-file)

(global-set-key (kbd "§") 'helm-resume)

;; Partially copied from https://github.com/jbranso/evil-dvorak/blob/master/evil-dvorak

(define-minor-mode dvorak-mode
  "Evil dvorak mode allows you to use evil using the dvorak keyboard layout.  Contribut
  nil
  :global t
  :lighter " ED"
  :keymap (make-sparse-keymap))

(defun turn-on-dvorak-mode ()
  "Enable evil-dvorak-mode in the current buffer."
  (dvorak-mode 1))

(defun turn-off-dvorak-mode ()
  "Disable evil-dvorak-mode in this buffer."
  (dvorak-mode -1))

(define-globalized-minor-mode global-dvorak-mode
  dvorak-mode turn-on-dvorak-mode
  "Global mode to let you use evil with dvorak friendly keybindings.")

(global-dvorak-mode 1)

(general-create-definer my-leader-def
  ;; :prefix my-leader
  :prefix "SPC")

(evil-define-key 'motion org-agenda-mode-map
  "h" 'org-agenda-earlier
  "l" 'org-agenda-later
  "j" 'org-agenda-next-line
  "k" 'org-agenda-previous-line
  "ö" 'org-agenda-goto-today
  )

(when (string-equal system-name "jeykeyarch")
  (evil-define-key '(visual normal motion) dvorak-mode-map
    "t" 'evil-next-line
    "n" 'evil-previous-line
    "h" 'evil-backward-char
    "s" 'evil-forward-char

    "H" 'evil-scroll-page-up
    "S" 'evil-scroll-page-down
    "T" 'evil-backward-paragraph
    "N" 'evil-forward-paragraph

    "l" 'evil-backward-word-begin
    "r" 'evil-forward-word-end
    "g" 'evil-first-non-blank
    "q" 'evil-end-of-line

    "m" 'evil-insert
    "z" 'evil-open-below
    "v" 'evil-delete-char
    "w" 'kill-line

    "M" 'evil-append
    "Z" 'evil-open-above
    "V" 'kill-word
    "W" 'kill-comment

    "f" 'yank
    "d" 'undo
    "b" 'kill-ring-save

    "p" 'ivy/refile
    "y" 'ivy/last

    ";" 'agenda/super
    "," 'todo/done
    "." 'todo/done

    ;; "Temporary quick binds.
    "j" 'kill-buffer-and-window
    "'" 'mu4e-headers-search-bookmark
    )
  )

(when (string-equal system-name "hunter")
  ;; Quick access (selection)
  (my-leader-def
    :keymaps 'normal
    "a" 'agenda/super
    "e" 'ivy/refile
    "r" 'ivy/last
    "t" 'todo/todo
    "d" 'org-deadline
    "k" 'org-schedule
    "h" 'home-file
    "j" 'projects-file
    "c" 'org-capture
    "x" 'bufler-switch-buffer
    )
  )

;; Buffers
;; 1
(my-leader-def
  :keymaps 'normal
  "bs" 'save-buffer
  "bk" 'kill-current-buffer
  "bj" 'kill-buffer-and-window
  "bb" 'bufler-switch-buffer
  "bf" 'find-file
  "br" 'org-recoll-search
  "bh" 'previous-buffer
  )

;; Windows & Navigation
;; 2
(my-leader-def
  :keymaps 'normal
  "wv" 'evil-window-vsplit
  "wk" 'hrs/split-window-below-and-switch
  "wc" 'hrs/split-window-right-and-switch
  "wo" 'find-file-other-window
  "wk" 'kill-current-buffer
  "wo" 'ace-window
  "wj" 'kill-buffer-and-window
  "wg" 'dumb-jump-go-other-window
  "wh" 'dump-jump-go
  "wb" 'dumb-jump-back
  )

;; Search
;; 3
(my-leader-def
  :keymaps 'normal
  "ss" 'consult-line
  "sS" 'consult-imenu
  "sr" 'replace-string
  )

;; Admin
;; 4
(my-leader-def
  :keymaps 'normal
  "qq" 'save-buffers-kill-terminal
  "qv" 'emacs-version
  "qi" 'emacs-init-time
  "qu" 'emacs-uptime
  "qe" 'eshell
  )
```

```
;; Org-mode
;; 5
(my-leader-def
  :keymaps 'normal
  "oi" 'org-cycle
  "oa" 'org-agenda
  "oc" 'org-capture
  "od" 'org-deadline
  "os" 'org-schedule
  "ot" 'org-todo
  "oz" 'org-set-tags-command
  "oe" 'org-set-effort
  "ox" 'todo/done
  "or" 'org-refile
  "og" 'ivy/refile
  "ob" 'ivy/last
  "ol" 'org-insert-link
  "oö" 'org-store-link
  "oo" 'org-open-at-point
  "op" 'org-link-open-as-file
  "of" 'org-agenda-file-to-front
  "ow" 'org-export-dispatch
  "oh" 'hoth-total
  "oy" 'org-archive-subtree
  )

;; Magit & VCS
;; 6
(my-leader-def
  :keymaps 'normal
  "gg" 'magit-status
  "gi" 'magit-init
  "gm" 'git-messenger:popup-message
  "gp" 'magit-pull
  )

;; Org-roam + Content (drill)
;; 7
(my-leader-def
  :keymaps 'normal
  "nl" 'org-roam
  "ni" 'org-roam-insert
  "nf" 'org-roam-find-file
  "nc" 'org-roam-capture
  "nr" 'org-roam-random-note
  "ns" 'org-roam-server-mode
  "nd" 'org-drill
  )

;; Email / Com
;; 8
(my-leader-def
  :keymaps 'normal
  "mo" 'mu4e
  "mc" 'mu4e-compose-new
  "mm" 'message-send-and-exit
  "ma" 'mail-add-attachment
  "ms" 'mml-secure-message-sign-pgp
  "me" 'mml-secure-message-encrypt-pgp
  "mj" 'mu4e~headers-jump-to-maildir
  "ml" 'mu4e~view-browse-url-from-binding
  "mf" 'mu4e~view-save-attach-from-binding
  )
```

4 Programming

General settings & packages for programming, including all programming major-modes.

4.1 Flycheck

Global syntax checking.

```
(use-package flycheck
  :ensure t
  :config
  (global-flycheck-mode)
  )
```

4.2 Magit

Use `magit` with some additional packages.

```
(use-package magit
  :ensure t
  :config
  (global-set-key (kbd "C-x g") 'magit-status)
  (global-set-key (kbd "C-x p") 'magit-init)
  (use-package magit-todos
    :ensure t
    :config
    (magit-todos-mode t)
    )
  (use-package git-messenger
    :ensure t
    :config
    (global-set-key (kbd "C-x m") 'git-messenger)
    )
  )
```

4.3 LSP

Setup `LSP` & `LSP-UI`, mainly for Elixir, later also for Rust.

```
(add-to-list 'exec-path "~/.tools/elixir-ls")

(setq lsp-ui-doc-max-height 52
      lsp-ui-doc-max-width 64
      lsp-ui-doc-position 'at-point
      lsp-ui-doc-position 'bottom
      lsp-ui-doc-show-with-mouse t
      lsp-ui-doc-show-with-cursor t
      )

(use-package lsp-mode
  :ensure t
  :commands lsp
  :init
  (setq lsp-headerline-breadcrumb-enable nil)
  (setq lsp-signature-auto-activate nil)
  :hook
  (elixir-mode . lsp)
  )

(use-package lsp-ui
  :ensure t
  :commands lsp-ui-mode
  :config
  (lsp-ui-doc-enable t)
  (lsp-ui-mode)
  (setq lsp-ui-doc-max-height 128
lsp-ui-doc-max-width 64
lsp-ui-doc-position 'top
lsp-ui-doc-show-with-mouse t
lsp-ui-doc-show-with-cursor t
)
  )
```

4.4 Smartparens

Automatically insert following parens.

```
(use-package smartparens
  :ensure t
  :hook
  (after-init . smartparens-global-mode)
  :config
  (require 'smartparens-config)
  (sp-pair "=" "=" :actions '(wrap))
  (sp-pair "+" "+" :actions '(wrap))
  (sp-pair "<" ">" :actions '(wrap))
  (sp-pair "$" "$" :actions '(wrap))
  )
```

4.5 Company

Used not just for programming, but easier to configure here.

```
(use-package company
  :ensure t
  :config
  (setq company-idle-delay 0.3)
  (add-hook 'after-init-hook 'global-company-mode)
  )

(use-package company-box
  :ensure t
  ;;:custom (company-box-icons-alist 'company-box-icons-all-the-icons)
  :hook (company-mode . company-box-mode)
  )
```

4.6 Snippets

Use yasnippets and the snippets for that.

```
(use-package yasnippet
  :ensure t
  :config
  (use-package yasnippet-snippets
    :ensure t
    )
  (yas-global-mode 1)
  (setq yas-indent-line 'auto)
  )
```

4.7 Format

Use `format-all` to language specific formatting.

```
(use-package format-all
  :ensure t
  :bind ("C-c C-f" . format-all-buffer)
  )
```

4.8 Modes

Collection of programming major modes.

```
(use-package web-mode
  :ensure t
  :config
  (add-hook 'web-mode-hook
    (lambda ()
      (rainbow-mode)
      (rspec-mode)
      (setq web-mode-markup-indent-offset 2)))
  )

(use-package elixir-mode
  :ensure t
  )

(use-package rust-mode
  :ensure t
  )

(use-package markdown-mode
  :ensure t
  )

(use-package systemd
  :ensure t
  :mode
  ("\\.service\\'" "\\.timer\\'" "\\.target\\'" "\\.mount\\'"
   "\\.automount\\'" "\\.slice\\'" "\\.socket\\'" "\\.path\\'"
   "\\.netdev\\'" "\\.network\\'" "\\.link\\'"))

(use-package yaml-mode
  :ensure t
  :mode ("\\.yaml\\'" "\\.yml\\'")
  :custom-face
  (font-lock-variable-name-face ((t (:foreground "violet"))))
  )
```

5 Writing

`org-mode` config for writing & productivity.

5.1 Base

General settings & config.

```
     (setq
      org-directory "~/documents/"
      org-archive-location "~/documents/archive/2021.org::* From %s"
      )

     (add-hook 'org-mode 'org-toggle-inline-images)
     (setq org-image-actual-width '(600))
     (setq-default org-display-inline-images t)
     (setq-default org-startup-with-inline-images t)

     (setq org-file-apps
   '((auto-mode . emacs)
     (directory . emacs)
     ("\\.mm\\'" . default)
     ("\\.x?html?\\'" . default)
     ("\\.pdf\\'" . default))
   ;;("\\.pdf\\'" . emacs))
   )
     (setq org-ellipsis "  "
   org-adapt-indentation nil
   org-fontify-quote-and-verse-blocks t
   org-startup-folded t
   org-priority-highest ?A
   org-priority-lowest ?C
   org-priority-faces
   '((?A . 'all-the-icons-red)
     (?B . 'all-the-icons-orange)
     (?C . 'all-the-icons-yellow))
   org-src-tab-acts-natively t
   org-hide-emphasis-markers t
   org-src-window-setup 'current-window
   org-return-follows-link t
   org-confirm-babel-evaluate nil
   org-use-speed-commands t
   org-catch-invisible-edits 'show
   )
(add-hook 'org-mode-hook 'org-indent-mode)
```

5.2 Tables

Use prettier tables.

```
(require 'org-pretty-table)
(add-hook 'org-mode-hook 'org-pretty-table-mode)
```

5.3 Looks

Use nicer faces for headings & deadlines.

```
(setq org-agenda-deadline-faces
      '((1.001 . error)
(1.0 . org-warning)
(0.5 . org-upcoming-deadline)
(0.0 . org-upcoming-distant-deadline)))

(custom-set-faces
  '(org-level-1 ((t (:inherit outline-1 :height 1.60))))
  '(org-level-2 ((t (:inherit outline-2 :height 1.40))))
  '(org-level-3 ((t (:inherit outline-3 :height 1.20))))
  '(org-level-4 ((t (:inherit outline-4 :height 1.0))))
  '(org-level-5 ((t (:inherit outline-5 :height 1.0))))
)
```

5.4 Appear

Use `org-appear` for nicer symbols in text.

```
(use-package org-appear
  :ensure t
  :hook (org-mode . org-appear-mode)
  :init (setq org-hide-emphasis-markers t
      org-appear-autoemphasis t
      org-appear-autolinks t
      org-appear-autosubmarkers t)
  )
```

5.5 Spellcheck

Enable `hunspell` & `flyspell` for all `org-mode` buffers.

```
(setq ispell-program-name "hunspell")

(setq ispell-local-dictionary "en_US")
(setq ispell-local-dictionary-alist
      '(("en_US" "[[:alpha:]]" "[^[:alpha:]]" "[']" nil ("-d" "en_US") nil utf-8)
("de_DE" "[[:alpha:]]" "[^[:alpha:]]" "[']" nil ("-d" "de_DE" "-a" "-i" "UTF-8") nil ut

(add-hook 'text-mode-hook #'flyspell-mode)
(add-hook 'org-mode-hook #'flyspell-mode)

(add-hook 'ispell-change-dictionary-hook #'flyspell-buffer)
```

5.6 Superstar

Use better stars for headings and for TODOs.

```
(use-package org-superstar
  :ensure t
  :config
  (setq org-superstar-headline-bullets-list '("" "" "" "" "" "" "" "")
;;org-superstar-headline-bullets-list '("" "" "" "" "" "" "" "" "" "")
org-superstar-prettify-item-bullets t
org-superstar-configure-like-org-bullets t
org-hide-leading-stars nil
org-superstar-leading-bullet ?\s
;; Enable custom bullets for TODO items
org-superstar-special-todo-items t
org-superstar-todo-bullet-alist '(("TODO" " ")
  ("NEXT" " ")
  ("STATIC" "» ")
  ("BLOCKED" " ")
  ("DONE" " ")
  ("PAL" " ")
  )
)
  (add-hook 'org-mode-hook (lambda () (org-superstar-mode 1)))
  )
```

5.7 Productivity

General productivity settings & capture templates.

```
(setq
 org-log-done 'time
 org-todo-keywords
 '((sequence "TODO(t)" "NEXT(n)" "|" "DONE(d)")
   (sequence "STATIC(s)" "BLOCKED(b)" "|" "PAL(p)"))
 org-todo-keyword-faces
      '(("TODO" . (:foreground "#af1212" :weight bold))
("NEXT" . (:foreground "#a8fa80" :weight bold))
("BLOCKED" . (:foreground "#b213c4" :weight bold))
("PAL" . (:foreground "#30bb03" :weight bold))
("STATIC" . (:foreground "#eaa222" :weight bold))
("DONE" . (:foreground "#ffffff" :weight bold))
)
      org-capture-templates '(("x" "Inbox TODO" entry (file "~/documents/aggregation.o
       "* TODO %?\n  %i\n  %a")
      ("c" "Common" entry (file+headline "~/documents/active.org" "Common")
       "* TODO %?\n%U\n   %c" :empty-lines 1)
      )

      org-tag-alist '(("drill" . ?d))
      org-refile-targets '(("~/documents/active.org" :maxlevel . 1)
   ("~/documents/completed.org" :maxlevel . 1)
   )
      )
```

5.8 Files

Quick jump to common files in GPS.

```
(global-set-key [f1] 'agenda/super)

(defun todo/done ()
  (interactive)
  (org-todo 'done))

(defun todo/todo  ()
  (interactive)
  (org-todo "NEXT")
  (org-mark-ring-push)
  (ivy/refile-to "~/documents/active.org" "Today")
  (org-mark-ring-goto)
  ;;(org-priority-up)
  ;;(org-deadline nil (org-read-date nil nil "+1d"))
  )

(defun home-file ()
    (interactive)
    (find-file "~/documents/active.org")
    )

(defun projects-file ()
    (interactive)
    (find-file "~/documents/aggregation.org")
    )
```

5.9 Agenda

Agenda & superagenda setup + helpers.

```
(defun agenda/super (&optional arg)
  (interactive "P")
  (org-agenda arg "d"))
```

Just use a single agenda for everything.

```
(use-package org-super-agenda
  :ensure t
  :init
  (setq org-agenda-custom-commands
'(("d" "Super Agenda - Day"
   ((agenda "" ((org-agenda-span 'day)
(org-super-agenda-groups
 '((:name "Today"
  :time-grid t
  :date today
  :scheduled today
    :order 1)))))
    (alltodo "" ((org-agenda-overriding-header "Next")
 (org-agenda-files '("~/documents/active.org"))
 (org-super-agenda-groups
  '((:name ""
   :todo "NEXT"
   :order 1)
    (:discard (:anything))
    ))))
    (alltodo "" ((org-agenda-overriding-header "Projects")
 (org-agenda-files '("~/documents/active.org"))
 (org-super-agenda-groups
  '((:name ""
   :todo ("TODO" "STATIC" "BLOCKED")
   :order 2)
    (:discard (:anything))
    )
    )))
    (alltodo "" ((org-agenda-overriding-header "Other")
 (org-super-agenda-groups
  '((:name ""
   :file-path "aggregation"
   :order 5)
    (:discard (:anything t)))
    )))
    )
   )
  )
)
  :config
  (org-super-agenda-mode 1)
  )

(setq
 org-agenda-start-on-weekday nil
 org-agenda-start-day "0d"
 org-agenda-skip-scheduled-if-done t
 org-agenda-skip-deadline-if-done t
 org-agenda-include-deadlines t
 org-agenda-current-time-string "← now"
 )
```

5.10 Refile

Use quick refile helpers for current GPS implementation.

```
;; https://emacs.stackexchange.com/questions/8045/org-refile-to-a-known-fixed-location
(defun ivy/refile-to (file headline)
  "Move current headline to specified location"
  (let ((pos (save-excursion
       (find-file file)
       (org-find-exact-headline-in-buffer headline))))
    (org-refile nil nil (list headline file nil pos))))

(defun ivy/refile ()
  "Move current headline to bookmarks"
  (interactive)
  (org-mark-ring-push)
  (ivy/refile-to "~/documents/active.org" "Today")
  (org-mark-ring-goto))

(defun ivy/last ()
  "Move current headline to bookmarks"
  (interactive)
  (org-mark-ring-push)
  (ivy/refile-to "~/documents/completed.org" "Week")
  (org-mark-ring-goto))
```

5.11 Roam

Use `org-roam` for the personal knowledge base.

```
(use-package org-roam
  :ensure t
  :commands (org-roam-insert org-roam-find-file org-roam-switch-to-buffer org-roam)
  :hook
  (after-init . org-roam-mode)
  :init
  (setq
   org-roam-directory (file-truename "~/documents/vaults/database/")
   org-roam-db-location "~/documents/vaults/org-roam.db"
   org-roam-db-gc-threshold most-positive-fixnum
   )
  :config
  (setq org-roam-capture-templates
'(("d" "default" plain (function org-roam--capture-get-point)
   "%?"
   :file-name "${slug}"
   :head "#+TITLE: ${title}\n"
   :immediate-finish t
   :unnarrowed t)
   ))
  (use-package org-roam-server
    :ensure t
    :config
    (setq org-roam-server-host "127.0.0.1"
org-roam-server-port 8080
org-roam-server-authenticate nil
org-roam-server-export-inline-images t
org-roam-server-serve-files nil
org-roam-server-served-file-extensions '("pdf" "mp4" "ogv" "jpg" "png")
org-roam-server-network-poll t
org-roam-server-network-arrows nil
org-roam-server-network-label-wrap-length 20))
  )
```

5.12 Repetition

`org-drill` for spaced repetition.

```
(use-package org-drill
  :ensure t
  :config
  (setq org-drill-use-visible-cloze-face-p t)
  (setq org-drill-hide-item-headings-p t)
  )
```

5.13 LaTeX

Inline LaTeX using `org-fragtog`.

```
(setq-default org-startup-with-latex-preview t)
```

```
(use-package org-fragtog
  :ensure t
  :config
  (add-hook 'org-mode-hook 'org-fragtog-mode)
  (setq org-latex-preview-ltxpng-directory "~/.ltxpng/")
  )
```

5.14 Export

All export targets in ox.

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

(use-package htmlize
  :ensure t)

(use-package ox-pandoc
  :ensure t
  )

(use-package ox-hugo
  :ensure t
  )

(use-package plantuml-mode
  :ensure t
  :config
  (setq org-plantuml-jar-path (expand-file-name "~/.tools/plantuml.jar"))
  (add-to-list 'org-src-lang-modes '("plantuml" . plantuml))
  )

(use-package ox-reveal
  :ensure t
  :custom ((org-reveal-root "https://cdn.jsdelivr.net/npm/reveal.js")
   (org-reveal-mathjax t)
   (org-reveal-ignore-speaker-notes nil)
   (org-reveal-note-key-char nil)))

(setq TeX-parse-self t)
(setq TeX-auto-save t)
```

```
(setq TeX-PDF-mode t)

(add-hook 'LaTeX-mode-hook
  (lambda ()
    (LaTeX-math-mode)
    (setq TeX-master t)))
```

Export settings for PDFs

```
(use-package ob-elixir
  :ensure t
  )

(use-package ob-rust
  :ensure t
  )

(org-babel-do-load-languages
 'org-babel-load-languages
 '((emacs-lisp . t)
   (elixir . t)
   (latex . t)
   ))

(setq org-src-fontify-natively t)

(setcdr (assoc "\\.pdf\\'" org-file-apps) "brave %s")
```

5.15 Title

Easy brute force title page.

```
(defun jk/title-title ()
  (car (org-roam--extract-titles-title))
  )

(defun jk/title-author ()
  (cdr (car (org-roam--extract-global-props '("AUTHOR"))))
  )
(defun jk/title-image ()
  (cdr (car (org-roam--extract-global-props '("IMAGE"))))
  )
(defun jk/title-subtitle ()
  (cdr (car (org-roam--extract-global-props '("SUBTITLE"))))
  )

(defun jk/title-compose ()
  (interactive)
  (insert (concat "
#+LATEX_HEADER: \\usepackage[utf8]{inputenc}
#+LATEX_HEADER: \\usepackage[dvipsnames]{xcolor}
#+LATEX_HEADER: \\usepackage{tikz}
#+LATEX_HEADER: \\usepackage[]{babel}
\\begin{titlepage}
    \\begin{center}
\\begin{tikzpicture}[remember picture,overlay]
    \\node[anchor=north west,yshift=-1.5pt,xshift=1pt]%
    at (current page.north west)
    {\\includegraphics[scale=1]{~/.tools/"
  (jk/title-image)
  ".png}};
\\end{tikzpicture}

\\vspace{2.2cm}

\\Huge
\\textbf{"
  (jk/title-title)
  "}

\\vspace{3.0cm}
\\LARGE"
  (jk/title-subtitle)
  "
\\vspace{4.2cm}"

  (jk/title-author)

"\\
\\vfill
\\Large
Baden, Schweiz\\
\\today
    \\end{center}
\\end{titlepage}
\\tableofcontents
\\newpage"
   )
  )
  )
```

6 Communication

mu4e and packages collection.

```
(setq mu4e-maildir (expand-file-name "~/.mail"))

(add-to-list 'load-path "/usr/share/emacs/site-lisp/mu4e")
(require 'mu4e)
(require 'smtpmail)

(setq mu4e-completing-read-function 'ivy-completing-read)
(setq mail-user-agent 'mu4e-user-agent)

(setq user-mail-address "jakob@jeykey.net"
      user-full-name  "Jakob Klemm"

      mu4e-get-mail-command "mbsync -c ~/.tools/.mbsyncrc -a"
      mu4e-update-interval  300
      mu4e-index-update-in-background t
      mu4e-main-buffer-hide-personal-addresses t

      send-mail-function 'smtpmail-send-it
      message-send-mail-function 'message-smtpmail-send-it
      starttls-use-gnutls t

      mu4e-sent-messages-behavior 'delete
      mu4e-view-show-addresses t

      message-kill-buffer-on-exit t

      mu4e-attachment-dir  "~/documents/vaults/ram"

      mu4e-sent-folder "/global/Sent"
      mu4e-drafts-folder "/global/Drafts"
      mu4e-trash-folder "/global/Trash"
      message-signature
      (concat
       "Jakob Klemm\n"
       "https://github.com/jakobklemm"
       "https://jeykey.net\n")
      mml-secure-openpgp-sign-with-sender t
      mml-secure-openpgp-encrypt-to-self t
      mml-secure-smime-sign-with-sender "jakob@jeykey.net"

      mu4e-view-prefer-html t

      )

(load-file "~/.tools/mail.el")

(setq smtpmail-starttls-credentilas my-mu4e-account-alist)
(setq smtpmail-default-smtp-server "smtp.gmail.com"
      smtpmail-smtp-server "smtp.gmail.com"
      smtpmail-smtp-service 587
      smtpmail-debug-info t)

(use-package mu4e-alert
  :ensure t
  :config
  (mu4e-alert-set-default-style 'libnotify)
  (add-hook 'after-init-hook #'mu4e-alert-enable-notifications)
  )

(defun my-mu4e-set-account ()
  "Set the account for composing a message."
  (let* ((account
  (if mu4e-compose-parent-message
      (let ((maildir (mu4e-message-field mu4e-compose-parent-message :maildir)))
(string-match "/\\(.*?\\)/" maildir)
(match-string 1 maildir))
    (completing-read (format "Compose with account: (%s) "
     (mapconcat #'(lambda (var) (car var))
my-mu4e-account-alist "/"))
     (mapcar #'(lambda (var) (car var)) my-mu4e-account-alist)
     nil t nil nil (caar my-mu4e-account-alist))))
 (account-vars (cdr (assoc account my-mu4e-account-alist))))
    (if account-vars
(mapc #'(lambda (var)
  (set (car var) (cadr var)))
      account-vars)
      (error "No email account found"))))

(add-hook 'mu4e-compose-pre-hook 'my-mu4e-set-account)
(add-hook 'mu4e-view-mode-hook 'visual-line-mode)
(add-hook 'mu4e-compose-mode-hook 'visual-line-mode)
```