Emacs Config

# 1 TODO for configs [0/5]

## 1.1 TODO Try out dap-mode

## 1.2 TODO Try org-fc

## 1.3 TODO Setup iedit

## 1.4 TODO Fix Dired Keymaps

# 2 Custom Tokens

## 2.1 Functions

Delete window or frame accordingly to prevent the Attempt to delete a... message.

(defun delete-window-or-frame (&optional window frame force)  
 (interactive)  
 (if (= 1 (length (window-list frame)))  
 (delete-frame frame force)  
 (delete-window window)))

The CTRL-L function.

(defun clear ()  
 (interactive)  
 (redraw-display)  
 (evil-force-normal-state)  
 (if (eq iedit-mode t)  
 (iedit--quit))  
 )

To prevent the evil escape key from impacting the visual mode.

(defun configure-evil-ins ()  
 "Default evil ins key"  
 (evil-escape-mode 1))  
(add-hook 'evil-insert-state-entry-hook #'configure-evil-ins)  
(add-hook 'minibuffer-mode-hook #'(lambda () (interactive) (evil-escape-mode 1) ))  
  
(defun configure-evil-exit-ins ()  
 "Default evil ins key"  
 (evil-escape-mode -1))  
(add-hook 'evil-visual-state-entry-hook #'configure-evil-exit-ins)

Cleaner buffer traversal.

(defcustom my-skippable-buffer-regexp  
 (rx bos (or "\*Messages\*" "\*scratch\*" "\*Help\*") eos)  
 "Matching buffer names are ignored by `my-next-buffer'  
 and `my-previous-buffer'."  
 :type 'regexp)  
  
(defun my-change-buffer (change-buffer)  
 "Call CHANGE-BUFFER until `my-skippable-buffer-regexp' doesn't match."  
 (let ((initial (current-buffer)))  
 (funcall change-buffer)  
 (let ((first-change (current-buffer)))  
 (catch 'loop  
 (while (string-match-p my-skippable-buffer-regexp (buffer-name))  
 (funcall change-buffer)  
 (when (eq (current-buffer) first-change)  
 (switch-to-buffer initial)  
 (throw 'loop t)))))))  
  
(setq my-skippable-buffer-regexp  
 (rx bos (or (or "\*Messages\*" "\*scratch\*" "\*Help\*")  
 (seq "\*" (zero-or-more anything)))  
 eos))  
  
(defun my-next-buffer ()  
 "Variant of `next-buffer' that skips `my-skippable-buffer-regexp'."  
 (interactive)  
 (my-change-buffer 'next-buffer))  
  
(defun my-previous-buffer ()  
 "Variant of `previous-buffer' that skips `my-skippable-buffer-regexp'."  
 (interactive)  
 (my-change-buffer 'previous-buffer))

File specific functions

(defun read-from-file (file)  
 (with-temp-buffer  
 (insert-file-contents file)  
 (read (current-buffer))))   
  
(defun open-current-file-in-vim ()  
 (interactive)  
 (async-shell-command  
 (format "foot nvim +%d %s"  
 (+ (if (bolp) 1 0) (count-lines 1 (point)))  
 (shell-quote-argument buffer-file-name))))  
  
  
(defun rename-current-buffer-file ()  
 "Renames current buffer and file it is visiting."  
 (interactive)  
 (let\* ((name (buffer-name))  
 (filename (buffer-file-name)))  
 (if (not (and filename (file-exists-p filename)))  
 (error "Buffer '%s' is not visiting a file!" name)  
 (let\* ((dir (file-name-directory filename))  
 (new-name (read-file-name "New name: " dir)))  
 (cond ((get-buffer new-name)  
 (error "A buffer named '%s' already exists!" new-name))  
 (t  
 (let ((dir (file-name-directory new-name)))  
 (when (and (not (file-exists-p dir)) (yes-or-no-p (format "Create directory '%s'?" dir)))  
 (make-directory dir t)))  
 (rename-file filename new-name 1)  
 (rename-buffer new-name)  
 (set-visited-file-name new-name)  
 (set-buffer-modified-p nil)  
 (when (fboundp 'recentf-add-file)  
 (recentf-add-file new-name)  
 (recentf-remove-if-non-kept filename))  
 (message "File '%s' successfully renamed to '%s'" name (file-name-nondirectory new-name))))))))  
  
(defun hm/convert-org-to-docx-with-pandoc ()  
 "Use Pandoc to convert .org to .docx.  
Comments:  
- The `-N' flag numbers the headers lines.  
- Use the `--from org' flag to have this function work on files  
 that are in Org syntax but do not have a .org extension"  
 (interactive)  
 (message "exporting .org to .docx")  
 (shell-command  
 (concat "pandoc -N --from org " (buffer-file-name)  
 " -o "  
 (file-name-sans-extension (buffer-file-name))  
 (format-time-string "-%Y-%m-%d-%H%M%S") ".docx")))

Open up:

(defun er-open-asm (arg)  
 "Open visited file in default external program.  
  
With a prefix ARG always prompt for command to use."  
 (interactive "P")  
 (when buffer-file-name  
 (shell-command (concat  
 (cond  
 ((and (not arg) (eq system-type 'darwin)) "open")  
 ((and (not arg) (member system-type '(gnu gnu/linux gnu/kfreebsd))) "xdg-open")  
 (t (read-shell-command "Open current file with: ")))  
 " "  
 (shell-quote-argument buffer-file-name)))))

Latex Compilation

(defun compile-latex-doc ()  
 (interactive)  
 (save-window-excursion  
 (shell-command  
 (format "pdflatex %s"  
 (shell-quote-argument buffer-file-name)))  
  
 ))  
  
(defun google-this ()  
 "Google the selected region if any, display a query prompt otherwise."  
 (interactive)  
 (browse-url  
 (concat  
 "http://www.google.com/search?ie=utf-8&oe=utf-8&q="  
 (url-hexify-string (if mark-active  
 (buffer-substring (region-beginning) (region-end))  
 (read-string "Google: "))))))  
  
(defun org-schedule-tomorrow ()  
 "Org Schedule for tomorrow (+1d)."  
 (interactive)  
 (org-schedule t "+1d"))

Extensions functions

(defun org-copy-blocks ()  
 (interactive)  
 (let ((code ""))  
 (save-restriction  
 (org-narrow-to-subtree)  
 (org-babel-map-src-blocks nil  
 (setq code (concat code (org-no-properties body)))))  
 (kill-new code)))

## 2.2 Variables

(defvar bgcolor "#11111b"  
 "The normal background of emacs.")  
(defvar grim-bgcolor "#14141f"  
 "The darker background of emacs.")  
(defvar dim-bgcolor "#1e1e2e"  
 "The darker background of emacs.")  
(defvar darker-bgcolor "#0E0E16"  
 "The darker background of emacs.")  
(defvar darkest-bgcolor "#0B0B11"  
 "The darker background of emacs.")  
(defvar dim-fgcolor "#424266"  
 "The calm foreground of emacs.")  
(defvar calm-fgcolor "#BAC2DE"  
 "The calm foreground of emacs.")  
(defvar mauve-color "#cba6f7"  
 "The blue color for emacs.")  
(defvar lavender-color "#b4befe"  
 "The blue color for emacs.")  
(defvar blue-color "#90b6f3"  
 "The blue color for emacs.")  
(defvar pink-color "#cba6f7"  
 "The pink color for emacs.")  
(defvar red-color "#f38ba8"  
 "The red color for emacs.")  
(defvar orange-color "#fab387"  
 "The red color for emacs.")  
(defvar teal-color "#a6e3a1"  
 "The pink color for emacs.")  
(defvar grim-fgcolor "#232338"  
 "The calm foreground of emacs.")  
(defvar cust-monospace "Iosevka Nerd Font"  
 "The monospace font for emacs.")  
(defvar cust-serif "Besley"  
 "The serif font for emacs.")  
(defvar cust-sans-serif "Barlow SemiCondensed"  
 "The sans font for emacs.")  
  
(defun set-custom-variables (frame)  
 "Org Schedule for tomorrow (+1d)."  
 (interactive)  
 (defvar cust-monospace "Iosevka Nerd Font"  
 "The monospace font for emacs.")  
 (defvar cust-sans-serif "Barlow SemiCondensed"  
 "The serif font for emacs.")  
 (defvar cust-serif "Besley"  
 "The sans font for emacs.")  
 (cond ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 1)  
 (setq bgcolor "#242933"  
 grim-bgcolor "#20242c"  
 dim-bgcolor "#21252d"  
 darker-bgcolor "#21252d"  
 darkest-bgcolor "#1C1F26"  
 dim-fgcolor "#333a47"  
 calm-fgcolor "#d8dee9"  
 mauve-color "#cba6f7"  
 lavender-color "#b4befe"  
 blue-color "#5e81ac"  
 pink-color "#cba6f7"  
 red-color "#bf616a"  
 orange-color "#d08770"  
 teal-color "#a3be8c"  
 grim-fgcolor "#20242c")  
 )  
  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 2)  
 (setq bgcolor "#131C19"  
 grim-bgcolor "#14141f"  
 dim-bgcolor "#1e1e2e"  
 darker-bgcolor "#101715"  
 darkest-bgcolor "#0b100e"  
 dim-fgcolor "#343c39"  
 calm-fgcolor "#444F4C"  
 mauve-color "#4d4055"  
 lavender-color "#32454d"  
 blue-color "#324448"  
 pink-color "#4d4046"  
 red-color "#333129"  
 orange-color "#4b4329"  
 teal-color "#324a43"  
 grim-fgcolor "#39413e")  
 )  
  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 3)  
 (setq bgcolor "#000000"  
 grim-bgcolor "#101010"  
 dim-bgcolor "#271c1a"  
 darker-bgcolor "#070707"  
 darkest-bgcolor "#010101"  
 dim-fgcolor "#605553"  
 calm-fgcolor "#958a88"  
 mauve-color "#a078a9"  
 lavender-color "#605553"  
 blue-color "#513833"  
 pink-color "#c3889e"  
 red-color "#C35864"  
 orange-color "#DE956F"  
 teal-color "#8faf87"  
 grim-fgcolor "#4a3f3d"  
 cust-sans-serif "ETbb"  
 cust-sans "Playfair Display"  
 ))))  
  
(add-hook 'after-make-frame-functions 'set-custom-variables)

## 2.3 Faces

(defface minibuffer-face  
 '((t :height 170  
 :foreground "#BAC2DE"  
 ))  
 "Face for minibuffer."  
 :group 'minibuffer )  
  
(defface eaBattery  
 '((t :height 130  
 :foreground "#6C7096"  
 ))  
 "Face for minibuffer."  
 :group 'echo-bar )  
  
(defface eaBattery-icon  
 '((t :height 130  
 :foreground "#3f4158"  
 ))  
 "Face for minibuffer."  
 :group 'echo-bar )  
  
(defface eaBattery-charge-icon  
 '((t :height 130  
 :foreground "#3f4158"  
 ))  
 "Face for minibuffer."  
 :group 'echo-bar )  
  
(defun custom-vars-setup ()  
 "The custom variables setup"  
 (interactive)  
 (cond ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 1)  
 (set-face-attribute 'minibuffer-face nil :foreground "#958a88" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery nil :foreground "#605553" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery-icon nil :foreground "#4a3f3d" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery-charge-icon nil :foreground "#4a3f3d" :font cust-sans-serif)  
 )  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 2)  
 (set-face-attribute 'minibuffer-face nil :foreground "#444F4C" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery nil :foreground "#343c39" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery-icon nil :foreground "#2b312f" :font cust-monospace)  
 (set-face-attribute 'eaBattery-charge-icon nil :foreground "#2b312f" :font cust-monospace)  
 )  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 3)  
 (set-face-attribute 'minibuffer-face nil :foreground "#958a88" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery nil :foreground "#605553" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery-icon nil :foreground "#4a3f3d" :font cust-sans-serif)  
 (set-face-attribute 'eaBattery-charge-icon nil :foreground "#4a3f3d" :font cust-sans-serif)  
 (set-frame-parameter nil 'alpha-background 90)  
 (add-to-list 'default-frame-alist '(alpha-background . 90))  
 )  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 0)  
 (set-face-attribute 'minibuffer-face nil :foreground "#BAC2DE" :font cust-sans-serif :weight 'regular)  
 (set-face-attribute 'eaBattery nil :foreground "#6C7096" :font cust-sans-serif :weight 'semibold)  
 (set-face-attribute 'eaBattery-icon nil :foreground "#3f4158" :font cust-sans-serif :weight 'semibold)  
 (set-face-attribute 'eaBattery-charge-icon nil :foreground "#3f4158" :font cust-sans-serif :weight 'regular)))  
 )

# 3 Options

## 3.1 Auto Tangle

(org-babel-load-file (locate-user-emacs-file "init.org"))

## 3.2 Package Startup Disable

(setq package-enable-at-startup nil)

## 3.3 Emacs UI

;; EMACS UIs

Remove modeline.

(setq-default mode-line-format nil)

No cursor in inactive windows.

(setq-default cursor-in-non-selected-windows nil)

No start up screen.

;;(setq inhibit-startup-screen t)

No echo area message during startup.

(setq server-client-instructions nil)  
;;(setq inhibit-startup-echo-area-message t)

No message in scratch buffer

(setq initial-scratch-message nil)

No Initial buffer, i.e. make initial buffer the scratch buffer.

(setq initial-buffer-choice nil)

No title frame.

(setq frame-title-format nil)

No scroll bars.

(if (fboundp 'scroll-bar-mode) (set-scroll-bar-mode nil))

No toolbars at all.

(if (fboundp 'tool-bar-mode) (tool-bar-mode -1))

No menu bar.

(menu-bar-mode -1)

No limit on font lock. (tree style levels)

(setq-default font-lock-maximum-size nil)

No Blinking cursor.

(blink-cursor-mode -1)

Display line numbers globally

(global-display-line-numbers-mode t)

Enable lines being truncated.

(set-default 'truncate-lines t)

Disable most messages in echo area. Doesnt work if enabled before starting the GUI.

(setq-default inhibit-message nil)

Setting some default frame settings. For border and cursors.

(add-to-list 'default-frame-alist '(internal-border-width . 10 ))  
(add-to-list 'default-frame-alist '(internal-show-cursor . -1))

Replace the truncation symbols for file buffers with truncation enabled. (Doesn't work for org-mode)

(set-display-table-slot standard-display-table 0 ?\ )

Disable the line break symbols in fringes.

(setf (cdr (assq 'continuation fringe-indicator-alist)) '(nil nil))

To disable buttons for checkboxes

(setq-default widget-image-enable nil)

### 3.3.1 Suppressing Messages

1. Suppress EOB and BOBs

* (defun my-command-error-function (data context caller)  
   "Ignore the buffer-read-only, beginning-of-buffer,  
   end-of-buffer signals; pass the rest to the default handler."  
   (when (not (memq (car data) '(buffer-read-only  
   beginning-of-buffer  
   end-of-buffer)))  
   (command-error-default-function data context caller)))  
    
  (setq command-error-function #'my-command-error-function)

1. Suppress on Events

* (defun suppress-message-advice-around (fun &rest args)  
   (let (message-log-max)  
   (with-temp-message (or (current-message) "")  
   (apply fun args))))  
    
  ;; example: suppress any messages from `save-buffer'  
  (advice-add 'save-buffer :around 'suppress-message-advice-around)  
  (advice-add 'kill-buffer :around 'suppress-message-advice-around)  
    
  ;; Disable echoing keystrokes  
  (setq-default echo-keystrokes 0.025)  
    
  ;; Disable Evil Modes information  
  (setq-default evil-echo-state nil)

# 4 Warnings

List of warnings that the byte-compiler should issue.

;; (setq byte-compile-warnings '(not free-vars unresolved noruntime lexical make-local))  
(setq byte-compile-warnings 'nil)

Silence compiler warnings as they can be pretty disruptive.

(if (boundp 'comp-deferred-compilation)  
 (setq-default comp-deferred-compilation nil)  
 (setq-default native-comp-deferred-compilation nil))  
  
(setq-default native-comp-async-report-warnings-errors nil)  
  
;; Display Minimum warnings  
(setq-default warning-minimum-level :emergency)

# 5 Sane Defaults

(setq-default  
 ad-redefinition-action 'accept ; Silence warnings for redefinition  
 delete-by-moving-to-trash t ; Delete files to trash  
 help-window-select t ; Focus new help windows when opened  
 mouse-yank-at-point t ; Yank at point rather than cursor  
 scroll-conservatively most-positive-fixnum ; Always scroll by one line  
 select-enable-clipboard t ; Merge system's and Emacs' clipboard  
 show-trailing-whitespace nil ; Do not display trailing whitespaces  
 tab-width 2 ; Set width for tabs  
 uniquify-buffer-name-style 'forward ; Uniquify buffer names  
 ring-bell-function 'ignore ; Be quiet!  
 custom-file (locate-user-emacs-file "custom-vars.el") ; Move customization variables to a separate file and load it  
 confirm-kill-processes nil  
 sentence-end-double-space nil ; a sentence ends with only one space  
 scroll-step 1 ; Line wise scroll.  
 scroll-conservatively 101 ; Whether to recenter cursor on scroll. If the value is greater than 100, it wont.  
 indent-tabs-mode nil  
 tab-always-indent 't  
 css-fontify-colors nil  
 tab-width 2  
 )   
  
(load custom-file 'noerror 'nomessage)  
(fset 'yes-or-no-p 'y-or-n-p) ; Replace yes/no prompts with y/n  
(recentf-mode 1) ; Remember recently opened files  
(save-place-mode 1) ; Remember the last traversed point in file  
(global-auto-revert-mode 1) ; Automatically revert buffers when the underlying file is changed  
(setq global-auto-revert-non-file-buffers t) ; Auto revert other non file buffers too  
  
;; Run at full power please  
(put 'downcase-region 'disabled nil)  
(put 'upcase-region 'disabled nil)  
(put 'narrow-to-region 'disabled nil)   
(put 'dired-find-alternate-file 'disabled nil) ; Open dired in same buffer

# 6 Emacs File Managements

## 6.1 Manage backups

Keep all backups in tmp/backups/ instead of cluttering the file system.

;; File syntax: DEMO.el~  
(setq backup-directory-alist `(("." . ,(expand-file-name "tmp/backups/" user-emacs-directory))))

## 6.2 Manage Autosaves

Keep all autosaves in tmp/auto-saves/ for the same reason.

;; File syntax #DEMO.el#  
  
;; auto-save-mode doesn't create the path automatically!  
(make-directory (expand-file-name "tmp/auto-saves/" user-emacs-directory) t)  
  
(setq auto-save-list-file-prefix (expand-file-name "tmp/auto-saves/sessions/" user-emacs-directory) ; prefix for generating autosave list file name  
 auto-save-file-name-transforms `((".\*" ,(expand-file-name "tmp/auto-saves/" user-emacs-directory) t)))

## 6.3 Creating Lockfiles DISABLED

Preventing lock files from being created. The create-lockfiles option is used to protect file from being edited by another emacs session while its unsaved

;; For #DEMO.el  
;; (setq create-lockfiles nil)

## 6.4 User Directory

The default directory to store packages and other files related to emacs.

(setq user-emacs-directory (expand-file-name "~/.cache/emacs"))

## 6.5 Recentf

Directory to store the recent files list.

(setq-default recentf-save-file "~/.config/emacs/recentf")

# 7 Default Paths

(add-to-list 'load-path "~/.config/emacs/packages/")

# 8 Packages

## 8.1 Elpaca Plugin Manager

(require 'elpaca-setup)

## 8.2 GCMH

(use-package gcmh)

## 8.3 Evil

### 8.3.1 Evil

(use-package evil  
 :init  
 (setq evil-undo-system 'undo-fu)  
 (setq evil-want-C-i-jump nil)  
 (setq evil-want-C-u-scroll t)  
 (setq evil-want-C-d-scroll t)  
 (setq evil-want-fine-undo t)  
 (setq evil-want-Y-yank-to-eol t)  
  
 ;; ----- Setting cursor colors  
 (setq evil-emacs-state-cursor '("#cba6f7" box))  
 (setq evil-normal-state-cursor '("#BAC2DE" box))  
 (setq evil-operator-state-cursor '("#90b6f3" (bar . 6)))   
 (setq evil-visual-state-cursor '("#6C7096" box))  
 (setq evil-insert-state-cursor '("#b4befe" (bar . 2)))  
 (setq evil-replace-state-cursor '("#eb998b" hbar))  
 (setq evil-motion-state-cursor '("#f38ba8" box))  
 :config  
 (evil-mode 1)  
 ;; INITIAL BINDINGS  
 (evil-global-set-key 'motion "j" 'evil-next-visual-line)  
 (evil-global-set-key 'motion "k" 'evil-previous-visual-line)  
 (evil-set-initial-state 'messages-buffer-mode 'normal)  
 (evil-set-initial-state 'dashboard-mode 'normal)  
 (evil-define-key 'motion help-mode-map "q" 'kill-this-buffer)  
 )

### 8.3.2 Evil Collection

(use-package evil-collection  
 :after evil  
 :config  
 (evil-collection-init))

### 8.3.3 Evil Commentary

(use-package evil-commentary  
 :after evil)

### 8.3.4 Evil Escape

(use-package evil-escape  
 :config  
 (evil-escape-mode)  
 :custom  
 (evil-escape-key-sequence "jk")  
 (evil-escape-delay 0.2))

### 8.3.5 Evil Surround

(use-package evil-surround  
 :config  
 (global-evil-surround-mode 1)  
 :after evil)

### 8.3.6 Evil Text Object anyblock

(use-package evil-textobj-anyblock  
 :config  
 (evil-define-text-object my-evil-textobj-anyblock-inner-quote  
 (count &optional beg end type)  
 "Select the closest outer quote."  
 (let ((evil-textobj-anyblock-blocks  
 '(("'" . "'")  
 ("\"" . "\"")  
 ("`" . "'")  
 ("“" . "”"))))  
 (evil-textobj-anyblock--make-textobj beg end type count nil)))  
  
 (evil-define-text-object my-evil-textobj-anyblock-a-quote  
 (count &optional beg end type)  
 "Select the closest outer quote."  
 (let ((evil-textobj-anyblock-blocks  
 '(("'" . "'")  
 ("\"" . "\"")  
 ("`" . "'")  
 ("“" . "”"))))  
 (evil-textobj-anyblock--make-textobj beg end type count t)))  
  
 (define-key evil-inner-text-objects-map "q" 'my-evil-textobj-anyblock-inner-quote)  
 (define-key evil-outer-text-objects-map "q" 'my-evil-textobj-anyblock-a-quote)  
  
 (add-hook 'lisp-mode-hook  
 (lambda ()  
 (setq-local evil-textobj-anyblock-blocks  
 '(("(" . ")")  
 ("{" . "}")  
 ("\\[" . "\\]")  
 ("\"" . "\"")  
 ))))  
  
 (define-key evil-inner-text-objects-map "u" 'evil-textobj-anyblock-inner-block)  
 (define-key evil-outer-text-objects-map "u" 'evil-textobj-anyblock-a-block)  
 )

## 8.4 Undo/Redo

(use-package undo-fu)  
(use-package undo-fu-session  
 :config  
 (setq undo-fu-session-incompatible-files '("/COMMIT\_EDITMSG\\'" "/git-rebase-todo\\'"))  
 (undo-fu-session-global-mode))

## 8.5 Helpful

(use-package helpful  
 :config  
 (setq counsel-describe-function-function #'helpful-callable)  
 (setq counsel-describe-variable-function #'helpful-variable))

## 8.6 Evil Org

(use-package evil-org  
 :diminish evil-org-mode  
 :after org  
 :config  
 (require 'evil-org-agenda)  
 (evil-org-agenda-set-keys)  
 (add-hook 'org-mode-hook 'evil-org-mode)  
 (add-hook 'evil-org-mode-hook  
 (lambda () (evil-org-set-key-theme))))

## 8.7 Catppuccin

(use-package catppuccin-theme  
 :config  
 ;; Customization  
 (setq catppuccin-flavor 'mocha) ;; or 'latte, 'macchiato, or 'mocha  
 (load-theme 'catppuccin :no-confirm)  
 (cond ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 1)  
 (catppuccin-set-color 'rosewater "#e8dee9")  
 (catppuccin-set-color 'flamingo "#81a1c1")  
 (catppuccin-set-color 'pink "#F5C2E7")  
 (catppuccin-set-color 'mauve "#b48ead")  
 (catppuccin-set-color 'red "#bf616a")  
 (catppuccin-set-color 'maroon "#E8A2AF")  
 (catppuccin-set-color 'peach "#d08770")  
 (catppuccin-set-color 'yellow "#ebcb8b")  
 (catppuccin-set-color 'green "#a3be8c")  
 (catppuccin-set-color 'teal "#B5E8E0")  
 (catppuccin-set-color 'sky "#5e81ac")  
 (catppuccin-set-color 'sapphire "#88c0d0")  
 (catppuccin-set-color 'blue "#88c0d0")  
 (catppuccin-set-color 'lavender "#81a1c1")  
 (catppuccin-set-color 'text "#eceff4")  
 (catppuccin-set-color 'subtext1 "#e5e9f0")  
 (catppuccin-set-color 'subtext0 "#e5e9f0")  
 (catppuccin-set-color 'overlay2 "#d8dee9")  
 (catppuccin-set-color 'overlay1 "#d8dee9")  
 (catppuccin-set-color 'overlay0 "#4c566a")  
 (catppuccin-set-color 'surface2 "#434c5e")  
 (catppuccin-set-color 'surface1 "#3b4252")  
 (catppuccin-set-color 'surface0 "#2e3440")  
 (catppuccin-set-color 'mantle "#242933")  
 (catppuccin-set-color 'crust "#2e3440")  
 (catppuccin-set-color 'base "#242933")  
 (catppuccin-reload))  
  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 2)  
 (catppuccin-set-color 'rosewater "#524546")  
 (catppuccin-set-color 'flamingo "#4d4046")  
 (catppuccin-set-color 'pink "#4d4046")  
 (catppuccin-set-color 'mauve "#4d4055")  
 (catppuccin-set-color 'red "#333129")  
 (catppuccin-set-color 'maroon "#34342d")  
 (catppuccin-set-color 'peach "#4b4329")  
 (catppuccin-set-color 'yellow "#434329")  
 (catppuccin-set-color 'green "#364629")  
 (catppuccin-set-color 'teal "#324a43")  
 (catppuccin-set-color 'sky "#324448")  
 (catppuccin-set-color 'sapphire "#324448")  
 (catppuccin-set-color 'blue "#324448")  
 (catppuccin-set-color 'lavender "#32454d")  
 (catppuccin-set-color 'text "#444F4C")  
 (catppuccin-set-color 'subtext1 "#424c49")  
 (catppuccin-set-color 'subtext0 "#3d4744")  
 (catppuccin-set-color 'overlay2 "#39413e")  
 (catppuccin-set-color 'overlay1 "#343c39")  
 (catppuccin-set-color 'overlay0 "#29302e")  
 (catppuccin-set-color 'surface2 "#29302e")  
 (catppuccin-set-color 'surface1 "#252c2a")  
 (catppuccin-set-color 'surface0 "#151C1A")  
 (catppuccin-set-color 'mantle "#151C1A")  
 (catppuccin-set-color 'crust "#151C1A")  
 (catppuccin-set-color 'base "#131C19")  
 (catppuccin-reload))  
  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 3)  
 (catppuccin-set-color 'rosewater "#cc897e")  
 (catppuccin-set-color 'flamingo "#ca9e97")  
 (catppuccin-set-color 'pink "#c3889e")  
 (catppuccin-set-color 'mauve "#9c6f68")  
 (catppuccin-set-color 'red "#C35864")  
 (catppuccin-set-color 'maroon "#B7535E")  
 (catppuccin-set-color 'peach "#5d4b42")  
 (catppuccin-set-color 'yellow "#5d4a40")  
 (catppuccin-set-color 'green "#8faf87")  
 (catppuccin-set-color 'teal "#5d4336")  
 (catppuccin-set-color 'sky "#605553")  
 (catppuccin-set-color 'sapphire "#605553")  
 (catppuccin-set-color 'blue "#605553")  
 (catppuccin-set-color 'lavender "#505553")  
 (catppuccin-set-color 'text "#958a88")  
 (catppuccin-set-color 'subtext1 "#8b807e")  
 (catppuccin-set-color 'subtext0 "#8b807e")  
 (catppuccin-set-color 'overlay2 "#605553")  
 (catppuccin-set-color 'overlay1 "#605553")  
 (catppuccin-set-color 'overlay0 "#4a3f3d")  
 (catppuccin-set-color 'surface2 "#3f3432")  
 (catppuccin-set-color 'surface1 "#271c1a")  
 (catppuccin-set-color 'surface0 "#010101")  
 (catppuccin-set-color 'mantle "#070707")  
 (catppuccin-set-color 'crust "#101010")  
 (catppuccin-set-color 'base "#000000")  
 (catppuccin-reload))  
  
 ((equal (read-from-file "/home/chilly/Scripts/data/themeIndex.txt") 0)  
 (catppuccin-set-color 'flamingo "#f2cdcd")  
 (catppuccin-set-color 'pink "#f5c2e7")  
 (catppuccin-set-color 'mauve "#cba6f7")  
 (catppuccin-set-color 'red "#f38ba8")  
 (catppuccin-set-color 'maroon "#eba0ac")  
 (catppuccin-set-color 'peach "#fab387")  
 (catppuccin-set-color 'yellow "#f9e2af")  
 (catppuccin-set-color 'green "#a6e3a1")  
 (catppuccin-set-color 'teal "#94e2d5")  
 (catppuccin-set-color 'sky "#89dceb")  
 (catppuccin-set-color 'sapphire "#74c7ec")  
 (catppuccin-set-color 'blue "#89b4fa")  
 (catppuccin-set-color 'lavender "#b4befe")  
 (catppuccin-set-color 'text "#cdd6f4")  
 (catppuccin-set-color 'subtext1 "#bac2de")  
 (catppuccin-set-color 'subtext0 "#a6adc8")  
 (catppuccin-set-color 'overlay2 "#9399b2")  
 (catppuccin-set-color 'overlay1 "#7f849c")  
 (catppuccin-set-color 'overlay0 "#6c7086")  
 (catppuccin-set-color 'surface2 "#585b70")  
 (catppuccin-set-color 'surface1 "#45475a")  
 (catppuccin-set-color 'surface0 "#313244")  
 (catppuccin-set-color 'mantle "#0E0E16")  
 (catppuccin-set-color 'crust "#0B0B11")  
 (catppuccin-set-color 'base "#11111B")  
 (catppuccin-reload))  
 ))

## 8.8 Orderless

;; Optionally use the `orderless' completion style.  
(use-package orderless  
 :init  
 ;; Configure a custom style dispatcher (see the Consult wiki)  
 ;; (setq orderless-style-dispatchers '(+orderless-consult-dispatch orderless-affix-dispatch)  
 ;; orderless-component-separator #'orderless-escapable-split-on-space)  
 (setq completion-styles '(orderless basic)  
 completion-category-defaults nil  
 completion-cycle-threshold 0  
 completion-category-overrides '((file (styles partial-completion)))))

## 8.9 Devdocs

(use-package devdocs)

## 8.10 Open with

(use-package openwith  
 :config  
 (require 'openwith)  
 (openwith-mode t)  
 (setq openwith-associations '(("\\.pdf\\'" "zathura" (file)) ("\\.pptx\\'" "libreoffice" (file)) ("\\.docx\\'" "libreoffice" (file))))  
 )

## 8.11 Corfu

### 8.11.1 Config

(use-package corfu  
 :init  
 (global-corfu-mode)  
 :custom  
 (corfu-cycle t) ;; Enable cycling for `corfu-next/previous'  
 (corfu-separator ?\s) ;; Orderless field separator  
 (corfu-quit-at-boundary nil) ;; Never quit at completion boundary  
 (corfu-quit-no-match nil) ;; Never quit, even if there is no match  
 (corfu-preview-current nil) ;; Disable current candidate preview  
 (corfu-preselect 'first) ;; Preselect the prompt  
 (corfu-on-exact-match t) ;; Configure handling of exact matches  
 (corfu-scroll-margin 5) ;; Use scroll margin  
 (corfu-minimum-width 100) ;; Use scroll margin  
 (corfu-maximum-width 190) ;; Use scroll margin  
 (corfu-auto-prefix 1)  
 (corfu-auto-delay 0.3)  
 (corfu-popupinfo-delay '(0.5 . 1.0))  
  
 :config  
 (corfu-popupinfo-mode 1)  
 (corfu-history-mode 1))  
  
(setq corfu--frame-parameters '((no-accept-focus . t)  
 (no-focus-on-map . t)  
 (min-width . t)  
 (min-height . t)  
 (border-width . 0)  
 (child-frame-border-width . 10)  
 (left-fringe . 0)  
 (right-fringe . 0)  
 (vertical-scroll-bars)  
 (horizontal-scroll-bars)  
 (menu-bar-lines . 0)  
 (tool-bar-lines . 0)  
 (tab-bar-lines . 0)  
 (no-other-frame . t)  
 (unsplittable . t)  
 (undecorated . t)  
 (cursor-type)  
 (no-special-glyphs . t)  
 (desktop-dont-save . t)))

### 8.11.2 Cape

;; Add extensions  
(use-package cape  
 :init  
 ;; Add `completion-at-point-functions', used by `completion-at-point'.  
 ;; (add-to-list 'completion-at-point-functions #'cape-dabbrev 5)  
 (add-to-list 'completion-at-point-functions #'cape-file)  
 (add-to-list 'completion-at-point-functions #'cape-history)  
 ;; (add-to-list 'completion-at-point-functions #'cape-keyword)  
 ;; (add-to-list 'completion-at-point-functions #'cape-elisp-symbol)  
 ;; (add-to-list 'completion-at-point-functions #'cape-elisp-block)  
 ;; (add-to-list 'completion-at-point-functions #'cape-line)  
 )

## 8.12 Org SuperStar

(use-package org-superstar)

## 8.13 Org Visual fill

(use-package visual-fill-column  
 :config  
  
 (defun org-mode-visual-fill ()  
 (setq visual-fill-column-width 150  
 visual-fill-column-center-text t)  
 (visual-fill-column-mode 1))  
  
 :hook (org-mode . org-mode-visual-fill))

## 8.14 Org Appear

(use-package org-appear  
 :config  
 ;; Hide org markup  
 (setq-default org-hide-emphasis-markers t)  
 (add-hook 'org-mode-hook 'org-appear-mode)  
 )

## 8.15 Org Roam

;; ROAM  
(use-package org-roam  
 :ensure t  
 :custom  
 (org-roam-directory (file-truename "~/Documents/notes"))  
 (org-roam-db-autosync-mode)  
 (org-roam-capture-templates  
 '(("d" "default" plain  
 "%?"  
 :if-new (file+head "${slug}.org" "#+title: ${title}\n#+Author:Adarsha Acharya")  
 :unnarrowed t)  
 ;; ("p" "project" plain "\* Goals\n\n%?\n\n\* Tasks\n\n\*\* TODO Add initial tasks\n\n\* Dates\n\n"  
 ;; :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n#+filetags: Project")  
 ;; :unnarrowed t)  
 ))  
 :config  
 (org-roam-setup))

## 8.16 Popper

(use-package popper  
 :init  
 (setq popper-reference-buffers  
 '("\\\*Messages\\\*"  
 "\\\*gud-test\\\*"  
 "Output\\\*$"  
 "\\\*Warnings\\\*"  
 help-mode  
 compilation-mode))  
 (popper-mode +1))

## 8.17 General

### 8.17.1 Init

(use-package general  
 :config

### 8.17.2 Global keys

(global-set-key [remap next-buffer] 'my-next-buffer)  
(global-set-key [remap previous-buffer] 'my-previous-buffer)  
  
(general-def  
 "C-j" 'nil  
 "C-k" 'nil)  
  
(general-def  
 "M-a" 'mark-whole-buffer  
 "M-p" 'popper-toggle-type  
 "M-n" 'popper-cycle  
 "M-," 'which-key-abort  
 "M-S-n" 'popper-cycle-backwards  
 "M-d" 'popper-kill-latest-popup  
 "C-;" 'embark-become  
 "C-<return>" 'embark-act  
 "<escape>" 'keyboard-escape-quit)

### 8.17.3 Leader keys definitions

(general-create-definer e/leader-keys  
 :keymaps '(normal insert visual emacs)  
 :prefix "SPC"  
 :global-prefix "C-SPC"  
 )  
  
(general-create-definer e/goto-keys  
 :keymaps '(normal insert)  
 :prefix "g"  
 :global-prefix "C-g"  
 )

### 8.17.4 Leader keys implementations

(e/leader-keys  
 "SPC" '(execute-extended-command :which-key " M-x ")  
 "k" '(lsp-ui-doc-glance :which-key " hover ")  
 )  
  
(e/leader-keys  
 "c" '(:ignore t :which-key "󰅱 code ")  
 "ca" '(lsp-execute-code-action :which-key " code actions ")  
 "cr" '(lsp-rename :which-key "󰑕 rename symbol ")  
 "ce" '(org-ctrl-c-ctrl-c :which-key "󰅱 execute code in org ")  
 "cs" '(lsp-iedit-highlights :which-key "󰅱 execute code in org ")  
 "cS" '(iedit-mode :which-key "󰅱 execute code in org ")  
 "cc" '(compile :which-key " format buffer ")  
 "cf" '(format-all-region-or-buffer :which-key " format buffer ")  
 "cF" '((lambda () (interactive) (indent-region (point-min) (point-max))) :wk " format default "))  
  
(e/leader-keys  
 "a" '(:ignore t :which-key " avy ")  
 "aa" '(evil-avy-goto-word-1 :which-key "󰀫 avy char ")  
 "al" '(avy-goto-line :which-key "󰂶 avy line ")  
 "am" '(:ignore t :which-key " avy move ")  
 "aml" '(avy-move-line :which-key "󰂶 avy move line "))  
  
(e/leader-keys  
 "f" '(:ignore t :which-key "󰈔 files ")  
 "ff" '(find-file :which-key "󰈞 find a file ")  
 "fr" '(consult-recent-file :which-key "󰣜 recent files ")  
 "fd" '(dired-jump :which-key "󰉓 open dired ")  
 "fi" '(evil-show-file-info :which-key " file info ")  
 "fot" '(org-babel-tangle :which-key "󰗆 org tangle")  
 "fn" '(org-roam-node-find :which-key "󰣜 find nodes ")  
 "fc" '(:ignore t :which-key "󰈔 current file ")  
 "fcr" '(recover-this-file :which-key "󰑕 rename current file "))  
  
(e/leader-keys  
 "o" '(:ignore t :which-key "󰉋 org ")  
 "oe" '(e/org-babel-edit :which-key "󰕪 open agendas ")  
 "od" '(hm/convert-org-to-docx-with-pandoc :which-key "󰕪 open convert org to docx ")  
 "oa" '(org-agenda :which-key "󰕪 open agendas ")  
 "oc" '(org-capture :which-key "󰄄 open capture ")  
 "oi" '(:ignore t :which-key "󰉋 org insert ")  
 "ois" '(org-schedule :which-key "󰾖 insert schedule ")  
 "oid" '(org-deadline :which-key "󰾕 insert deadline ")  
 "oil" '(org-insert-link :which-key " insert link ")  
 "on" '(org-roam-node-insert :which-key " insert link ")  
 "og" '(org-roam-graph :which-key "󱁉 Open graph "))  
  
(e/leader-keys  
 "g" '(:ignore t :which-key " get ")  
 "gi" '(consult-imenu :which-key "󰮫 get imenu ")  
 "gf" '(list-faces-display :which-key " 󰙃 get faces")  
 "gc" '(zenity-cp-color-at-point-dwim :which-key " 󰙃 colors picker")  
 "gk" '(consult-yank-from-kill-ring :which-key " get kill ring and yank "))  
  
(e/leader-keys  
 "l" '(:ignore t :which-key "󰃷 Latex Commands ")  
 "lv" '(TeX-view :which-key "󰃷 Latex View ")  
 "lc" '(compile-latex-doc :wk " Latex Compile "))  
  
(e/leader-keys  
 "x" '(:ignore t :which-key "󰃷 execute ")  
 "xr" '((lambda () (interactive) (load-file "~/.config/emacs/init.el")) :wk " Reload emacs config ")  
 "x" '(:ignore t :which-key "󰃷 execute latex commands "))  
  
(e/leader-keys  
 "i" '(:ignore t :which-key "󰡁 insert ")  
 "ii" '(nerd-icons-insert :which-key "󰭟 insert icons ")  
 "it" '(org-insert-time-stamp :which-key " insert time stamp ")  
 )  
  
(e/leader-keys  
 "b" '(:ignore t :which-key " buffer navigation ")  
 "bd" '(kill-buffer-and-window :which-key "󰆴 kill the current buffer and window ")  
 "bk" '(kill-some-buffers :which-key "󰛌 kill all other buffers and windows ")  
 "bn" '(next-buffer :which-key "󰛂 switch buffer ")  
 "bp" '(previous-buffer :which-key "󰛁 switch buffer ")  
 "bb" '(consult-buffer :which-key "󰕰 view buffers "))  
  
  
(e/leader-keys  
 "s" '(:ignore t :which-key " search ")  
 "ss" '(consult-line :which-key "󰱼 line search ")  
 "sr" '(consult-ripgrep :which-key "󰟥 search with rg ")  
 "sp" '(consult-fd :which-key " search with fd ")  
 "sd" '(dictionary-search :which-key " search in dictionary "))  
  
(e/leader-keys  
 "e" '(:ignore t :which-key "󰈈 evaluate ")  
 "eb" '(eval-buffer :which-key "󰷊 evaluate buffer ")  
 "ee" '(eval-last-sexp :which-key "󰷊 evaluate last expression ")  
 "er" '(eval-region :which-key "󰨺 evaluate region "))  
  
(e/leader-keys  
 "h" '(:ignore t :which-key "󰞋 help ")  
 "ht" '(helpful-at-point :which-key " describe this ")  
 "hF" '(describe-face :which-key "󱗎 describe face ")  
 "hf" '(helpful-function :which-key "󰯻 describe function ")  
 "hh" '(devdocs-lookup :which-key "󰯻 describe function ")  
 "hb" '(embark-bindings :which-key "󰌌 describe bindings ")  
 "hk" '(helpful-key :which-key "󰯻 describe this key ")  
 "hv" '(helpful-variable :which-key " describe variable ")  
 "hrb" '(revert-buffer-quick :which-key "󰄸 reload buffer "))  
  
(e/leader-keys  
 "t" '(:ignore t :which-key " toggles/switches ")  
 "tt" '(toggle-truncate-lines :which-key "󰖶 toggle word wrap mode ")  
 "tv" '(visual-line-mode :which-key " visual line mode ")  
 "tR" '(read-only-mode :which-key "󰑇 read only mode ")  
 "tc" '(:ignore t :which-key "󰮫 toggle corfu ")  
 "tce" '((lambda () (interactive) (setq-default corfu-auto t) (corfu-mode 1)) :wk " enable ")  
 "tcd" '((lambda () (interactive) (setq-default corfu-auto nil) (corfu-mode 1)) :wk " disable ")  
 "tf" '(flymake-mode :which-key " toggle flymake ")  
 "tb" '(breadcrumb-mode :which-key " toggle breadcrumbs ")  
 "tr" '(org-roam-buffer-toggle :which-key " Roam Buffer ")  
 "to" '(:ignore t :which-key "󰮫 toggle org ")  
 "tol" '(org-toggle-link-display :which-key " Toggle Link Display ")  
 "tm" '(minimap-mode :which-key "󰍍 minimap toggles "))  
  
(e/goto-keys  
 "n" '(flymake-goto-next-error :which-key " next error")  
 "p" '(flymake-goto-prev-error :which-key " prev error"))

### 8.17.5 Evil Mode

1. Normal State

* (general-def  
   :keymaps 'evil-normal-state-map  
   "M-d" '(duplicate-dwim :which-key " code duplicate ")  
   "C-u" #'evil-scroll-up  
   "C-d" #'evil-scroll-down  
   "C-s" (lambda () (interactive) (evil-ex "%s/"))  
   "C-l" 'clear  
   "C-n" 'iedit-next-occurrence  
   "C-S-n" 'iedit-prev-occurrence  
   "RET" 'org-open-at-point-global  
   "M-k" 'drag-stuff-up  
   "M-j" 'drag-stuff-down  
   "M-h" 'drag-stuff-left  
   "M-l" 'drag-stuff-right  
   "C-/" #'consult-line-multi  
   "gcc" #'evil-commentary-line  
   "gca" (lambda () (interactive) (comment-indent) (just-one-space) (evil-append-line 1))  
   )

1. Insert State

* (general-def  
   :keymaps 'evil-insert-state-map  
   "C-h" 'nil  
   "C-l" 'completion-at-point  
   "C-f" 'find-file-at-point  
   )

1. Visual State

* (general-def  
   :keymaps 'evil-visual-state-map  
   "gc" #'evil-commentary/org-comment-or-uncomment-region  
   ;; "C-k" 'corfu-previous  
   ;; "C-j" 'corfu-next  
   ;; "C -." 'yas-expand  
   )

1. Motion State

* (general-def  
   :keymaps 'evil-motion-state-map  
   )

### 8.17.6 Org Mode

(general-def  
 :keymaps 'org-mode-map  
 "C-h" 'nil  
 "C-S-h" 'nil  
 )

### 8.17.7 Other Modes

(general-def  
 :keymaps 'vertico-map  
 "C-l" '(lambda () (interactive) (vertico-insert) )  
 "C-S-l" '(lambda () (interactive) (vertico-insert) (minibuffer-force-complete-and-exit))  
 "C-k" #'vertico-next  
 "C-j" #'vertico-previous  
 "C-h" #'vertico-directory-up  
 )  
  
(general-def  
 :keymaps 'corfu-map  
 "C-k" 'corfu-previous  
 "C-j" 'corfu-next  
 "C-l" 'completion-at-point  
 "C-h" 'corfu-quit  
 )

### 8.17.8 End block

;; NOTE: =Information on general=  
  
;; INFO: Mode specific maps  
;; (general-def org-mode-map  
;; "C-c C-q" 'counsel-org-tag  
;; ;; ...  
;; )  
  
;; INFO: normal maps  
;; (general-define-key  
;; "M-x" 'amx  
;; "C-s" 'counsel-grep-or-swiper)  
  
;; INFO: prefix  
;; (general-define-key  
;; :prefix "C-c"  
;; ;; bind "C-c a" to 'org-agenda  
;; "a" 'org-agenda  
;; "b" 'counsel-bookmark  
;; "c" 'org-capture)  
  
;; INFO: Swap!  
;; (general-swap-key nil 'motion  
;; ";" ":")  
)

## 8.18 Rainbow Mode

(use-package rainbow-mode  
 :hook (prog-mode org-mode text-mode))

## 8.19 Rainbow Delimeters

(use-package rainbow-delimiters  
 :hook (org-mode prog-mode text-mode))

## 8.20 Whichkey

(use-package which-key  
 :config  
 ;; Set the time delay (in seconds) for the which-key popup to appear. A value of  
 ;; zero might cause issues so a non-zero value is recommended.  
 (setq which-key-idle-delay 0.3)  
  
 ;; Set the maximum length (in characters) for key descriptions (commands or  
 ;; prefixes). Descriptions that are longer are truncated and have ".." added.  
 ;; This can also be a float (fraction of available width) or a function.  
 (setq which-key-max-description-length 27)  
  
 ;; Use additional padding between columns of keys. This variable specifies the  
 ;; number of spaces to add to the left of each column.  
 (setq which-key-add-column-padding 3)  
  
 ;; The maximum number of columns to display in the which-key buffer. nil means  
 ;; don't impose a maximum.  
 (setq which-key-max-display-columns nil)  
  
 ;; Set the separator used between keys and descriptions. Change this setting to  
 ;; an ASCII character if your font does not show the default arrow. The second  
 ;; setting here allows for extra padding for Unicode characters. which-key uses  
 ;; characters as a means of width measurement, so wide Unicode characters can  
 ;; throw off the calculation.  
 (setq which-key-separator " " )  
  
 ;; Set the prefix string that will be inserted in front of prefix commands  
 ;; (i.e., commands that represent a sub-map).  
 (setq which-key-prefix-prefix " " )  
  
 ;; Set the special keys. These are automatically truncated to one character and  
 ;; have which-key-special-key-face applied. Disabled by default. An example  
 ;; setting is  
 ;; (setq which-key-special-keys '("SPC" "TAB" "RET" "ESC" "DEL"))  
 (setq which-key-special-keys nil)  
  
 ;; Show the key prefix on the left, top, or bottom (nil means hide the prefix).  
 ;; The prefix consists of the keys you have typed so far. which-key also shows  
 ;; the page information along with the prefix.  
 (setq which-key-show-prefix 'nil)  
  
 ;; Set to t to show the count of keys shown vs. total keys in the mode line.  
 (setq which-key-show-remaining-keys nil)  
  
 (setq which-key-frame-max-height 10)  
  
 (setq which-key-frame-max-width 150)  
  
 (setq which-key-popup-type 'frame)  
  
 (which-key-mode))

## 8.21 Vertico

;; Enable vertico  
(use-package vertico  
 :init  
 (vertico-mode)  
 (vertico-reverse-mode)  
 ;; Different scroll margin  
 (setq vertico-scroll-margin 2)  
  
 ;; Show more candidates  
 (setq vertico-count 5)  
  
 ;; Grow and shrink the Vertico minibuffer  
 (setq vertico-resize t)  
  
 ;; Optionally enable cycling for `vertico-next' and `vertico-previous'.  
 (setq vertico-cycle t)  
  
 (add-hook 'minibuffer-mode-hook (lambda () (interactive)  
 (setq-local face-remapping-alist '((default minibuffer-face))))))

## 8.22 Consult

(use-package consult  
 :init  
 (setq register-preview-delay 0.5  
 register-preview-function #'consult-register-format)  
  
 (advice-add #'register-preview :override #'consult-register-window)  
 (setq xref-show-xrefs-function #'consult-xref  
 xref-show-definitions-function #'consult-xref)  
 :config  
 (add-to-list 'consult-buffer-filter "\\*.\*\\*")  
 ;; Allowing single key press to begin asynchorous searches like consult-grep  
 (setq consult-async-min-input 1)  
  
 (consult-customize  
 consult-theme consult-buffer :preview-key '(:debounce 0.2 any)  
 consult-recent-file :preview-key "C-h"  
 consult-ripgrep consult-git-grep consult-grep  
 consult-bookmark consult-xref  
 consult--source-bookmark consult--source-file-register  
 consult--source-recent-file consult--source-project-recent-file  
 ;; :preview-key "M-."  
 :preview-key '(:debounce 0.4 any))  
  
 (setq consult-narrow-key "<") ;; "C-+"  
 )

## 8.23 Embark

(use-package embark)  
(use-package embark-consult)

## 8.24 Icons

(use-package nerd-icons  
 :config  
 :if (display-graphic-p))  
  
(use-package nerd-icons-completion  
 :config  
 (nerd-icons-completion-mode)  
 )  
  
(use-package nerd-icons-dired  
 :hook  
 (dired-mode . nerd-icons-dired-mode))

## 8.25 Iedit

(use-package iedit)

## 8.26 SmartParens

(use-package smartparens  
 :config  
 (sp-pair "$$" "$$") ;; latex math mode.   
  
 (require 'smartparens-config)  
 (add-hook 'text-mode-hook 'smartparens-mode)  
 (add-hook 'prog-mode-hook 'smartparens-mode)  
 (add-hook 'org-mode-hook 'smartparens-mode))  
(use-package evil-smartparens  
 :hook (smartparens-mode))

## 8.27 Yasnippets

(use-package yasnippet  
 :config  
 (yas-global-mode))  
  
(use-package yasnippet-snippets)

## 8.28 LSPs

### 8.28.1 LSP mode

(use-package lsp-mode  
 :init  
 (setq read-process-output-max (\* 1024 1024)) ;; 1mb

The one additional feature we have used below is Cape’s cape-capf-buster. This isn’t entirely necessary, but it gives some nicer behavior when you alter the original text during completion (e.g. while completing get, you delete back to ge).

(defun my/lsp-mode-setup-completion ()  
 (setf (alist-get 'styles (alist-get 'lsp-capf completion-category-defaults))  
 '(flex))) ;; Configure flex  
  
(defun lsp-ui-doc--handle-hr-lines nil  
 (let (bolp next before after)  
 (goto-char 1)  
 (while (setq next (next-single-property-change (or next 1) 'markdown-hr))  
 (when (get-text-property next 'markdown-hr)  
 (goto-char next)  
 (setq bolp (bolp)  
 before (char-before))  
 (delete-region (point) (save-excursion (forward-visible-line 1) (point)))  
 (setq after (char-after (1+ (point))))  
 (insert  
 (concat  
 (and bolp (not (equal before ?\n)) (propertize "\n" 'face '(:height 0.5)))  
 (propertize " "  
 ;; :align-to is added with lsp-ui-doc--fix-hr-props  
 'display '(space :height (1))  
 'lsp-ui-doc--replace-hr t  
 ;; 'face '(:background "dark grey")  
 )  
 ;; :align-to is added here too  
 (propertize " " 'display '(space :height (1)))  
 (and (not (equal after ?\n)) (propertize " \n" 'face '(:height 0.2)))))))))  
:hook  
(lsp-completion-mode . my/lsp-mode-setup-completion)  
(prog-mode . lsp-deferred)  
  
:config  
(setq lsp-ui-doc-enable nil)  
(setq lsp-ui-doc-show-with-cursor nil)  
(setq lsp-ui-doc-show-with-mouse nil)  
(setq lsp-lens-enable nil)  
(setq lsp-idle-delay 0.0)  
(setq lsp-headerline-breadcrumb-enable nil)  
(setq lsp-ui-sideline-enable nil)  
(setq lsp-ui-sideline-show-code-actions nil)  
(setq lsp-ui-sideline-show-hover nil)  
(setq lsp-diagnostics-provider :flymake)  
(setq lsp-ui-sideline-enable nil) ; To disable the entire sideline  
(setq lsp-modeline-code-actions-enable nil)  
(setq lsp-modeline-diagnostics-mode nil)  
(setq lsp-ui-sideline-enable nil)  
(setq lsp-ui-sideline-show-diagnostics nil)  
(setq lsp-eldoc-enable-hover nil) ; Eldoc  
(setq lsp-signature-auto-activate nil) ;; you could manually request them via `lsp-signature-activate`  
(setq lsp-signature-render-documentation nil)  
(setq lsp-completion-provider :none) ;; we use Corfu!  
(setq lsp-completion-show-detail nil)  
  
(setq lsp-ui-doc-frame-parameters  
 '((left . -1)  
 (no-focus-on-map . t)  
 (min-width . 0)  
 (width . 0)  
 (min-height . 0)  
 (height . 0)  
 (internal-border-width . 15)  
 (vertical-scroll-bars . nil)  
 (horizontal-scroll-bars . nil)  
 (right-fringe . 0)  
 (menu-bar-lines . 0)  
 (tool-bar-lines . 0)  
 (tab-bar-lines . 0)  
 (tab-bar-lines-keep-state . 0)  
 (line-spacing . 0)  
 (unsplittable . t)  
 (undecorated . t)  
 (bottom . -1)  
 (visibility . nil)  
 (mouse-wheel-frame . nil)  
 (no-other-frame . t)  
 (inhibit-double-buffering . t)  
 (drag-internal-border . t)  
 (no-special-glyphs . t)  
 (desktop-dont-save . t)))  
  
:commands (lsp lsp-deferred))

### 8.28.2 LSP UI

(use-package lsp-ui  
 :config  
 (setq lsp-ui-doc-position 'at-point)  
 (setq lsp-ui-doc-border darker-bgcolor)  
 )

## 8.29 Format all

(use-package format-all  
 :commands format-all-mode  
 :hook (prog-mode . format-all-mode)  
 :config  
 (setq-default format-all-formatters '(("C" (astyle "--mode=c"))  
 ("Shell" (shfmt "-i" "4" "-ci")))))

## 8.30 Org Modern

(use-package org-modern  
 :hook (org-mode . org-modern-mode)  
 :config  
 (setq  
 org-modern-star '( "" "  " "  " "  ")  
 org-modern-list '((42 . "◦") (43 . "•") (45 . "–"))  
 org-modern-block-name nil  
 org-modern-keyword nil  
 org-modern-todo t  
 org-modern-table nil  
 )  
 (set-face-attribute 'org-modern-done nil :foreground dim-fgcolor :background bgcolor :weight 'bold :slant 'normal :height 130 :inherit 'nil)  
 (set-face-attribute 'org-modern-todo nil :background darker-bgcolor :foreground blue-color :weight 'bold :height 130 :inherit 'fixed-pitch)  
 (set-face-attribute 'org-modern-time-inactive nil :foreground dim-fgcolor :background darker-bgcolor :height 130 :inherit 'nil)  
 (set-face-attribute 'org-modern-time-inactive nil :foreground dim-fgcolor :background grim-bgcolor :height 130 :inherit 'nil)  
 (set-face-attribute 'org-modern-time-active nil :background dim-fgcolor :foreground darker-bgcolor :height 130 :inherit 'nil)  
 (set-face-background 'fringe (face-attribute 'default :background))  
  
 )

## 8.31 Org Visual fill

(use-package visual-fill-column  
 :config  
  
 (defun org-mode-visual-fill ()  
 (setq visual-fill-column-width 150  
 visual-fill-column-center-text t)  
 (visual-fill-column-mode 1))  
  
 :hook (org-mode . org-mode-visual-fill))

## 8.32 Org Appear

(use-package org-appear  
 :config  
 ;; Hide org markup  
 (setq-default org-hide-emphasis-markers t)  
 (add-hook 'org-mode-hook 'org-appear-mode)  
 )

## 8.33 Org Roam

;; ROAM  
(use-package org-roam  
 :ensure t  
 :custom  
 (org-roam-directory (file-truename "~/Documents/collegeNotes/"))  
 (org-roam-db-autosync-mode)  
 (org-roam-capture-templates  
 '(("d" "default" plain  
 "%?"  
 :if-new (file+head "${slug}.org" "#+title: ${title}\n#+Author:Adarsha Acharya")  
 :unnarrowed t)  
 ;; ("p" "project" plain "\* Goals\n\n%?\n\n\* Tasks\n\n\*\* TODO Add initial tasks\n\n\* Dates\n\n"  
 ;; :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n#+filetags: Project")  
 ;; :unnarrowed t)  
 ))  
 :config  
 (org-roam-setup))

## 8.34 Whichkey

(use-package which-key  
 :config  
 ;; Set the time delay (in seconds) for the which-key popup to appear. A value of  
 ;; zero might cause issues so a non-zero value is recommended.  
 (setq which-key-idle-delay 0.3)  
  
 ;; Set the maximum length (in characters) for key descriptions (commands or  
 ;; prefixes). Descriptions that are longer are truncated and have ".." added.  
 ;; This can also be a float (fraction of available width) or a function.  
 (setq which-key-max-description-length 27)  
  
 ;; Use additional padding between columns of keys. This variable specifies the  
 ;; number of spaces to add to the left of each column.  
 (setq which-key-add-column-padding 3)  
  
 ;; The maximum number of columns to display in the which-key buffer. nil means  
 ;; don't impose a maximum.  
 (setq which-key-max-display-columns nil)  
  
 ;; Set the separator used between keys and descriptions. Change this setting to  
 ;; an ASCII character if your font does not show the default arrow. The second  
 ;; setting here allows for extra padding for Unicode characters. which-key uses  
 ;; characters as a means of width measurement, so wide Unicode characters can  
 ;; throw off the calculation.  
 (setq which-key-separator " " )  
  
 ;; Set the prefix string that will be inserted in front of prefix commands  
 ;; (i.e., commands that represent a sub-map).  
 (setq which-key-prefix-prefix " " )  
  
 ;; Set the special keys. These are automatically truncated to one character and  
 ;; have which-key-special-key-face applied. Disabled by default. An example  
 ;; setting is  
 ;; (setq which-key-special-keys '("SPC" "TAB" "RET" "ESC" "DEL"))  
 (setq which-key-special-keys nil)  
  
 ;; Show the key prefix on the left, top, or bottom (nil means hide the prefix).  
 ;; The prefix consists of the keys you have typed so far. which-key also shows  
 ;; the page information along with the prefix.  
 (setq which-key-show-prefix 'nil)  
  
 ;; Set to t to show the count of keys shown vs. total keys in the mode line.  
 (setq which-key-show-remaining-keys nil)  
  
 (setq which-key-frame-max-height 10)  
  
 (setq which-key-frame-max-width 150)  
  
 (setq which-key-popup-type 'frame)  
  
 (which-key-mode))

## 8.35 Drag Stuff

(use-package drag-stuff  
 :hook (org-mode . drag-stuff-mode)  
 :hook (prog-mode . drag-stuff-mode))

## 8.36 Plugins END

(elpaca-wait)

# 9 Extra Packages

## 9.1 Echo Bar

(require 'echo-bar)  
(echo-bar-mode)

## 9.2 Zenity Color Picker

(require 'zenity-color-picker)

# 10 Emacs Features

## 10.1 Pairs

;; Custom pairs for electric pair  
;; (defvar org-electric-pairs '((?~ . ?~)) "Electric pairs for org-mode.")  
;; (electric-pair-mode 1)  
;; (show-paren-mode 1)  
  
;; Disable the autocompletion of pairs <>  
;; (add-hook 'org-mode-hook (lambda () (setq-local electric-pair-inhibit-predicate `(lambda (c) (if (char-equal c ?<) t (,electric-pair-inhibit-predicate c))))))  
  
;; (defun org-add-electric-pairs ()  
;; (interactive)  
  
;; (setq show-paren-when-point-inside-paren 't)  
;; (setq show-paren-highlight-openparen 'nil)  
;; (setq electric-pair-preserve-balance 't)  
;; (setq show-paren-style 'parenthesis)  
;; (setq electric-pair-pairs (append electric-pair-pairs org-electric-pairs))  
;; (setq electric-pair-text-pairs electric-pair-pairs))  
  
;; (add-hook 'org-mode-hook (lambda () (org-add-electric-pairs)))

## 10.2 Org Mode

### 10.2.1 Org Src

;; How is a buffer opened when calling `org-edit-special'.  
(setq org-src-window-setup 'current-window)  
(defun e/org-babel-edit ()  
 "Edit python src block with lsp support by tangling the block and  
 then setting the org-edit-special buffer-file-name to the  
 absolute path. Finally load the lsp."  
 (interactive)  
  
 ;; org-babel-get-src-block-info returns lang, code\_src, and header  
 ;; params; Use nth 2 to get the params and then retrieve the :tangle  
 ;; to get the filename  
 (setq mb/tangled-file-name (expand-file-name (assoc-default :tangle (nth 2 (org-babel-get-src-block-info)))))  
  
 ;; tangle the src block at point  
 (org-babel-tangle '(4))  
 (org-edit-special)  
  
 ;; Now we should be in the special edit buffer with python-mode. Set  
 ;; the buffer-file-name to the tangled file so that pylsp and  
 ;; plugins can see an actual file.  
 (setq-local buffer-file-name mb/tangled-file-name)  
 (lsp)  
 )

### 10.2.2 Org Agenda

(with-eval-after-load 'org  
 (setq org-directory "~/Documents/collegeNotes")  
 (setq org-agenda-files '("~/Documents/collegeNotes")) ; DO not add backslash at the end  
 ;; (setq org-agenda-block-separator 32) ; Make it space  
 (setq org-agenda-block-separator " ") ; Make it space  
 (setq org-agenda-window-setup 'current-window) ; Make it space  
 ;; (setq org-fancy-priorities-list '("" "󰉀" ""))  
 (setq org-agenda-prefix-format "%c ")  
  
 (setq org-agenda-include-all-todo nil)  
 (setq org-agenda-skip-scheduled-if-done t)  
 (setq org-agenda-skip-deadline-if-done t)  
 (setq org-agenda-include-diary t)  
 (setq org-agenda-info t)  
 (setq org-agenda-columns-add-appointments-to-effort-sum t)  
 (setq org-agenda-default-appointment-duration 60)  
 (setq org-agenda-mouse-1-follows-link t)  
 (setq org-agenda-skip-unavailable-files t)  
 (setq org-agenda-use-time-grid nil)  
  
  
 (setq org-agenda-todo-keyword-format "%-6s")  
 (setq org-agenda-custom-commands  
 '(("v" "A better agenda view"  
 ((agenda "")  
 (alltodo ""  
 ((org-agenda-overriding-header "Unscheduled tasks")))  
 ))  
 ))  
 (defvar org-agenda--todo-keyword-regex  
 (cl-reduce (lambda (cur acc)  
 (concat acc "\\|" cur))  
 (mapcar (lambda (entry) (concat "\\\* " entry))  
 '("TODO" "HOMEWORK" "DONE")))  
 "Regex which filters all TODO keywords")  
  
 )  
(defun org-agenda-change-font ()  
 (interactive)  
 (setq buffer-face-mode-face '(:family "Barlow SemiCondensed" :height 140 :foreground "#424266" ))  
 (buffer-face-mode))  
  
(add-hook 'org-agenda-mode-hook #'org-agenda-change-font)

### 10.2.3 Miscelleneous

(setq org-ellipsis " ⋅")  
  
(defun org-config (frame)  
 "Configure Org mode things. Intended for `after-make-frame-functions'."  
 (setq org-agenda-start-with-log-mode nil)  
 (setq org-log-done 'nil)  
 (setq org-agenda-span 10)  
 (setq org-agenda-start-on-weekday nil)  
 (custom-set-variables  
 ;; custom-set-variables was added by Custom.  
 ;; If you edit it by hand, you could mess it up, so be careful.  
 ;; Your init file should contain only one such instance.  
 ;; If there is more than one, they won't work right.  
 '(org-emphasis-alist  
 (quote  
 (("\*" bold)  
 ("/" italic)  
 ("\_" underline)  
 ("=" org-verbatim verbatim)  
 ("~" org-code verbatim)  
 ("+"  
 (:strike-through t))  
 ))))  
  
 (setq org-log-into-drawer t)  
 (setf (cdr (assoc 'file org-link-frame-setup)) 'find-file)  
 (setq org-startup-folded 'nofold)  
 )  
(add-hook 'after-make-frame-functions 'org-config)

### 10.2.4 Org Capture

(setq org-capture-templates  
 `(("t" "Task" entry (file+olp "~/Documents/notes/home.org" "Inbox")  
 "\* TODO %?\n %i")  
 ("h" "Homework" entry (file+olp "~/Documents/notes/home.org" "Inbox")  
 "\* TODO %?\n %i")))

### 10.2.5 Org tempo

(require 'org-tempo)  
  
;; ShortCuts  
(add-to-list 'org-structure-template-alist '("sh" . "src shell :results verbatim"))  
(add-to-list 'org-structure-template-alist '("el" . "src emacs-lisp \n "))  
(add-to-list 'org-structure-template-alist '("py" . "src python :results output \n"))  
(add-to-list 'org-structure-template-alist '("cpp" . "src C++ :results verbatim \n\n #include <iostream>\n using namespace std;\n\n int main(){\n return 0;\n}"))  
(add-to-list 'org-structure-template-alist '("cl" . "src C :results verbatim \n\n #include <iostream>\n using namespace std;\n\n int main(){\n return 0;\n}"))  
(add-to-list 'org-structure-template-alist '("asm" . "src asm :results verbatim"))

## 10.3 Dired

(setq dired-use-ls-dired nil)  
(setq dired-kill-when-opening-new-dired-buffer t)  
(setq-default dired-kill-when-opening-new-dired-buffer 't)  
(setq dired-listing-switches "-Agho --group-directories-first")  
  
(defun config-dired ()  
 "Dired hook."  
 (evil-collection-define-key 'normal 'dired-mode-map  
 "l" 'dired-find-alternate-file  
 "h" 'dired-up-directory  
 "c" 'dired-create-empty-file  
 "Q" 'kill-buffer-and-window  
 )  
 (face-remap-add-relative 'default '(:family "Barlow Semi Condensed")))  
  
(add-hook 'dired-mode-hook 'config-dired)  
(add-hook 'dired-mode-hook 'dired-hide-details-mode)

## 10.4 Flymake

(setq flymake-fringe-indicator-position nil)

## 10.5 Treesit

;; Syntax Highlighting  
(require 'treesit)  
  
(add-to-list 'treesit-language-source-alist '(bash "https://github.com/tree-sitter/tree-sitter-bash.git"))  
(add-to-list 'major-mode-remap-alist '(sh-mode . bash-ts-mode))  
(add-to-list 'major-mode-remap-alist '(shell-script-mode . bash-ts-mode))  
  
(add-to-list 'treesit-language-source-alist '(python "https://github.com/tree-sitter/tree-sitter-python.git"))  
(add-to-list 'major-mode-remap-alist '(python-mode . python-ts-mode))  
  
(add-to-list 'treesit-language-source-alist '(cpp "https://github.com/tree-sitter/tree-sitter-cpp.git"))  
(add-to-list 'major-mode-remap-alist '(c++-mode . c++-ts-mode))  
  
(add-to-list 'treesit-language-source-alist '(c "https://github.com/tree-sitter/tree-sitter-c"))  
(add-to-list 'major-mode-remap-alist '(c-mode . c-ts-mode))  
  
(add-to-list 'treesit-language-source-alist '(css "https://github.com/tree-sitter/tree-sitter-css.git"))  
(add-to-list 'major-mode-remap-alist '(css-mode . css-ts-mode))  
  
(add-to-list 'treesit-language-source-alist '(html "https://github.com/tree-sitter/tree-sitter-html.git"))  
(add-to-list 'major-mode-remap-alist '(html-mode . html-ts-mode))  
  
(setq treesit-font-lock-level 4)

# 11 Styling

## 11.1 Default

(add-to-list 'default-frame-alist '(font . "Iosevka Nerd Font Medium"))  
(defun configure-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (set-face-attribute 'default nil :font cust-monospace :height 150)  
 (set-face-attribute 'fixed-pitch nil :font cust-monospace :height 150)  
 (set-face-attribute 'variable-pitch nil :font cust-sans-serif :height 170)  
 (set-face-attribute 'font-lock-comment-face nil :slant 'italic)  
 (set-face-attribute 'font-lock-keyword-face nil :slant 'italic)  
 (set-face-attribute 'line-number nil :font cust-monospace :height 120)  
 (set-face-attribute 'link nil :background darker-bgcolor :slant 'oblique :weight 'regular :overline 'nil :underline 'nil :family cust-serif )  
 (set-face-attribute 'show-paren-match nil :foreground dim-fgcolor :background 'unspecified :underline 'nil)  
 (set-face-attribute 'show-paren-match-expression nil :background grim-bgcolor :foreground 'unspecified :inherit 'nil)  
 (set-face-attribute 'help-key-binding nil :font cust-sans-serif :weight 'semibold :background darker-bgcolor :foreground dim-fgcolor :box 'nil)  
 (set-face-attribute 'header-line nil :background bgcolor :foreground dim-fgcolor)  
 (set-face-attribute 'window-divider nil :background bgcolor :foreground bgcolor)  
 )  
  
(add-hook 'server-after-make-frame-hook 'configure-font)

## 11.2 Org mode

(defun configure-org-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (set-face-attribute 'org-block nil :background darker-bgcolor :font cust-monospace)  
 (set-face-attribute 'org-verbatim nil :background 'unspecified :foreground dim-fgcolor :inherit 'fixed-pitch)  
 (set-face-attribute 'org-block-end-line nil :background darker-bgcolor :foreground darker-bgcolor :font cust-sans-serif :height 200)  
 (set-face-attribute 'org-block-begin-line nil :background darker-bgcolor :foreground darker-bgcolor :font cust-sans-serif :height 200)  
 (set-face-attribute 'org-meta-line nil :slant 'normal :height 90 :foreground bgcolor :font cust-serif)  
 (set-face-attribute 'org-drawer nil :foreground bgcolor)  
 (set-face-attribute 'org-todo nil :background bgcolor :foreground teal-color :weight 'bold :font cust-sans-serif :height 200)  
 (set-face-attribute 'org-agenda-diary nil :foreground blue-color :weight 'bold :font cust-sans-serif)  
  
 (set-face-attribute 'org-document-info-keyword nil :foreground bgcolor)  
 (set-face-attribute 'org-level-1 nil :height 235 :family cust-serif :weight 'semibold :foreground lavender-color)  
 (set-face-attribute 'org-level-2 nil :height 220 :family cust-serif :weight 'semibold :foreground lavender-color)  
 (set-face-attribute 'org-level-3 nil :height 205 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-level-4 nil :height 190 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-level-5 nil :height 190 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-level-6 nil :height 190 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-level-7 nil :height 190 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-level-8 nil :height 190 :family cust-serif :weight 'regular :foreground blue-color)  
 (set-face-attribute 'org-table nil :background darker-bgcolor :inherit 'fixed-pitch)  
  
 (set-face-attribute 'org-document-title nil :height 260 :font cust-sans-serif :foreground blue-color)  
 (set-face-attribute 'org-ellipsis nil :slant 'normal :foreground dim-fgcolor)  
 (set-face-attribute 'org-done nil :slant 'normal :strike-through 't :foreground dim-fgcolor)  
  
 (set-face-attribute 'org-agenda-date nil :font cust-sans-serif :weight 'regular :height 200 :foreground dim-fgcolor)  
 (set-face-attribute 'org-agenda-date-today nil :font cust-sans-serif :weight 'semibold :height 200 )  
 (set-face-attribute 'org-agenda-done nil :font cust-serif :weight 'regular :height 190 :strike-through 't)  
 (set-face-attribute 'org-agenda-structure nil :font cust-serif :weight 'regular :height 230 :foreground lavender-color)  
 )  
  
(add-hook 'org-mode-hook #'configure-org-font)

## 11.3 Vertico

(defun configure-vertico-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (interactive)  
 (set-face-attribute 'vertico-current nil :foreground blue-color :weight 'semibold :background darker-bgcolor :family cust-sans-serif)  
 (set-face-attribute 'vertico-multiline nil :weight 'semibold :height 170 :family cust-sans-serif)  
 (set-face-attribute 'minibuffer-prompt nil :foreground mauve-color :weight 'semibold :background bgcolor :height 190 :family cust-sans-serif)  
 (set-face-attribute 'minibuffer-face nil :height 170 )  
 )  
(add-hook 'server-after-make-frame-hook #'configure-vertico-font)

## 11.4 Corfu

(defun configure-corfu-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
  
 (set-face-attribute 'corfu-default nil :height 150 :background darker-bgcolor :foreground dim-fgcolor :weight 'semibold :family "Iosevka Nerd Font")  
 (set-face-attribute 'corfu-current nil :height 150 :foreground calm-fgcolor :background bgcolor :weight 'semibold :family "Iosevka Nerd Font")  
 (set-face-attribute 'corfu-annotations nil :height 150 :foreground grim-fgcolor :weight 'semibold :family "Iosevka Nerd Font")  
 )  
(add-hook 'server-after-make-frame-hook 'configure-corfu-font)

## 11.5 Smart Parens

(defun configure-parens-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (set-face-attribute 'sp-show-pair-enclosing nil :background darkest-bgcolor :foreground 'unspecified :inherit 'nil)  
 (set-face-attribute 'sp-pair-overlay-face nil :background darkest-bgcolor :foreground 'unspecified :inherit 'nil))  
(add-hook 'smartparens-mode-hook #'configure-parens-font)

## 11.6 LSP

(defun configure-lsp-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
  
 (set-face-attribute 'lsp-ui-doc-background nil :background darker-bgcolor )  
 (set-face-attribute 'lsp-face-highlight-textual nil :foreground 'unspecified :background grim-fgcolor :inherit 'nil)  
 (set-face-attribute 'lsp-face-highlight-write nil :underline 'nil :foreground 'unspecified :background grim-fgcolor :inherit 'nil)  
 (set-face-attribute 'lsp-face-highlight-read nil :underline 'nil :foreground 'unspecified :background grim-fgcolor :inherit 'nil)  
 )  
(add-hook 'server-after-make-frame-hook 'configure-lsp-font)

## 11.7 Flymake

(defun configure-flymake-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `lsp-mode'."  
 (set-face-attribute 'flymake-error nil :background "#42232c" :foreground "#F38BA8" :underline 'nil :weight 'bold)  
 (set-face-attribute 'flymake-note nil :background "#262d25" :foreground "#A6E3A1" :underline 'nil :weight 'bold)  
 (set-face-attribute 'flymake-warning nil :background "#453e29" :foreground "#F8D782" :underline 'nil :weight 'bold))  
(add-hook 'flymake-mode-hook #'configure-flymake-font)

## 11.8 Dired

(defun configure-dired-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (set-face-attribute 'dired-header nil :height 250 :weight 'normal)  
 )  
(add-hook 'dired-mode-hook #'configure-dired-font)

## 11.9 Evil

(defun configure-evil-font ()  
 "Configure font given initial non-daemon FRAME.  
 Intended for `after-make-frame-functions'."  
 (set-face-attribute 'evil-ex-info nil :foreground red-color :slant 'oblique :family "Barlow Semi Condensed" )  
 (set-face-attribute 'evil-ex-substitute-matches nil :background blue-color :foreground darker-bgcolor :strike-through 't :underline 'nil )  
 (set-face-attribute 'evil-ex-substitute-replacement nil :background teal-color :foreground darker-bgcolor :underline 'nil ))  
  
(add-hook 'server-after-make-frame-hook 'configure-evil-font)

# 12 Hooks

(add-hook 'focus-out-hook 'garbage-collect)  
(add-hook 'server-after-make-frame-hook 'custom-vars-setup)  
  
(add-hook 'org-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
(add-hook 'org-agenda-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
(add-hook 'term-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
(add-hook 'dired-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
(add-hook 'shell-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
(add-hook 'treemacs-mode-hook #'(lambda () (display-line-numbers-mode -1)))  
  
;; Corfu  
(add-hook 'eshell-mode-hook  
 (lambda ()  
 (setq corfu-auto t) ;; Enable auto completion  
 (setq-local corfu-auto nil)))  
  
(add-hook 'prog-mode-hook  
 (lambda ()  
 (setq corfu-auto nil) ;; Enable auto completion  
 ))  
  
(add-hook 'org-mode-hook  
 (lambda ()  
 (org-indent-mode 1)  
 (variable-pitch-mode 1) ;; Enable Variable pitch  
 (setq corfu-auto nil) ;; Enable auto completion  
 ))  
  
(defun corfu-enable-always-in-minibuffer ()  
 "Enable Corfu in the minibuffer if Vertico/Mct are not active."  
 (unless (or (bound-and-true-p mct--active)  
 (bound-and-true-p vertico--input)  
 (eq (current-local-map) read-passwd-map))  
 (setq-local corfu-auto nil) ;; Enable/disable auto completion  
 (setq-local corfu-echo-delay nil ;; Disable automatic echo and popup  
 corfu-popupinfo-delay nil)  
 (corfu-mode 1)))  
  
;; SRC  
(add-hook 'minibuffer-setup-hook #'corfu-enable-always-in-minibuffer 1)  
(add-hook 'org-src-mode-hook #'(lambda () (interactive) (setq header-line-format 'nil)))  
(add-hook 'org-capture-mode-hook #'(lambda () (interactive) (setq header-line-format 'nil)))  
  
;; LaTeX  
(add-hook 'LaTeX-mode-hook 'visual-line-mode)  
(add-hook 'LaTeX-mode-hook 'turn-on-reftex)  
(add-hook 'LaTeX-mode-hook #'(lambda () (interactive)  
 (lsp)  
 (prettify-symbols-mode 1) ))  
  
;; (add-hook 'prog-mode-hook #'(lambda () (interactive)  
;; (add-hook 'evil-insert-state-exit-hook #'(lambda () (interactive) (flymake-mode 1)))  
;; (add-hook 'evil-insert-state-entry-hook #'(lambda () (interactive) (flymake-mode -1)))  
;; ))

# 13 Ending init

;; -------------------------------------------------------------------------------- ;;  
;; Completed init.el ;;  
;; -------------------------------------------------------------------------------- ;;