Vanilla Emacs configuration

Config for GO, Java and Flutter development, also note taking!

Pramudya Arya Wicaksana

November 3, 2023

Contents

1 Making emacs work for me **3**
1.1 How to use my config . . . 3

2 Simple config **3**

3 Package Manager **3**

4 Add PATH **4**

5 Shell **4**

6 Display and look **4**
6.1 Line number . . . 4
6.2 Theme . . . 5
6.3 Font . . . 5
6.4 Lignature . . . 5
6.5 Reduce clutter . . . 6
6.6 Basic functionality . . . 6
6.7 Icons . . . 7
6.8 Dashboard . . . 7
6.9 Modeline . . . 8
6.10 Neotree (Filetree) . . . 9
6.11 Tabbar . . . 9
6.12 Backup . . . 9
6.13 Parenthesis . . . 10

7 Keyboard related stuffs **10**
7.1 Evil mode . . . 10
7.2 Evil keybind collection . . . 11
7.3 Undo . . . 11
7.4 General keybind . . . 11
7.5 Hydra . . . 12
7.6 Which key . . . 12

8 Helm **12**

9 Third party connection **13**
9.1 Elcord . . . 13
9.2 EMMS . . . 13

10 Programming **14**
10.1 Treesitter . . . 14
10.2 LSP Mode . . . 14
10.3 LSP UI . . . 15
10.4 LSP Mapping . . . 16
10.5 DAP Mode . . . 17
10.6 Snippets . . . 18
10.7 Flycheck . . . 19
10.8 Yasnippets . . . 19
10.9 Languages . . . 19
10.9.1 GO . . . 19
10.9.2 Javascript . . . 21
10.9.3 Typescript . . . 21
10.9.4 Java . . . 22
10.9.5 Scala . . . 22
10.9.6 Yaml . . . 22
10.9.7 Dart . . . 23
10.9.8 Rust . . . 24
10.9.9 Web . . . 24
10.10 Smerge . . . 25
10.11 Projectile . . . 26
10.12 VTerm . . . 26
10.13 Magit . . . 27
10.14 Tests . . . 27
10.14.1 Jest . . . 27
10.15 Container . . . 27

10.15.1 Kubernetes 28
10.16 Blamer 28

11 Writing **29**
11.1 Org Mode 29
11.2 Org babel 30
11.3 Org bullet 30
11.4 Org save tangle 31
11.5 Calfw 31
11.6 LaTeX 31

12 Workspace **32**
12.1 Prespective-el 32

13 Keybinding **33**
13.1 Basic Keybinding 33
13.1.1 Window keybind 33
13.1.2 General keybind 33
13.1.3 Open keybind 34

1 Making emacs work for me

1.1 How to use my config

Just directly clone this repo to `$HOME/.emacs.d/` folder and open your emacs~

This config working on version emacs24

2 Simple config

```
(setq byte-compile-warnings '(cl-functions))
(global-auto-revert-mode t)
(setq-default cursor-type 'bar)
```

3 Package Manager

```
(require 'package)
(setq package-archives
      '(("melpa" . "https://melpa.org/packages/")
```

```
        ("org" . "https://orgmode.org/elpa/")
        ("elpa" . "https://elpa.gnu.org/packages/")))

(package-initialize)

(unless (package-installed-p 'use-package)
  (package-install 'use-package))

(require 'use-package)
(setq use-package-always-ensure t)
(setq package-enable-at-startup nil)
```

4 Add PATH

Adding `.zshrc` stuff as path

```
(let ((path (shell-command-to-string ". ~/.zshrc; echo -n $PATH")))
  (setenv "PATH" path)
  (setq exec-path
        (append
         (split-string-and-unquote path ":")
         exec-path)))
```

5 Shell

Shell configuration using ZSH

```
(setq explicit-shell-file-name "/usr/bin/zsh")
(setq shell-file-name "zsh")
(setq explicit-zsh-args '("--login" "--interactive"))
(defun zsh-shell-mode-setup ()
  (setq-local comint-process-echoes t))
(add-hook 'shell-mode-hook #'zsh-shell-mode-setup)
```

6 Display and look

6.1 Line number

```
(add-hook 'prog-mode-hook 'display-line-numbers-mode)
(setq display-line-numbers-type 'relative)
```

6.2 Theme

```
(use-package atom-one-dark-theme)
(load-theme 'atom-one-dark t)
(if (display-graphic-p)
    ;; Code for graphical frames
    (load-theme 'atom-one-dark t)
  ;; Code for terminal frames
  (load-theme 'atom-one-dark t))
```

6.3 Font

```
(set-frame-font "JetBrains Mono 10" nil t)
(add-to-list 'default-frame-alist '(font . "JetBrains Mono 10"))
```

6.4 Lignature

Show programming lignature

```
(dolist (char/ligature-re
         `((?-  . ,(rx (or (or "-->" "-<<" "->>" "-|" "-~" "-<" "->") (+ "-"))))
           (?/  . ,(rx (or (or "/==" "/=" "/>" "/**" "/*") (+ "/"))))
           (?*  . ,(rx (or (or "*>" "*/") (+ "*"))))
           (?<  . ,(rx (or (or "<<=" "<<-" "<|||" "<==>" "<!--" "<=>" "<||" "<|>" "<-<
                               "<==" "<=<" "<-|" "<~>" "<=|" "<~~" "<$>" "<+>" "</>"
                               "<*>" "<->" "<=" "<|" "<:" "<>"  "<$" "<-" "<~" "<+"
                               "</" "<*")
                           (+ "<"))))
           (?:  . ,(rx (or (or ":?>" "::=" ":>" ":<" ":?" ":=") (+ ":"))))
           (?=  . ,(rx (or (or "=>>" "==>" "=/=" "=!=" "=>" "=:=") (+ "="))))
           (?!  . ,(rx (or (or "!==" "!=") (+ "!"))))
           (?>  . ,(rx (or (or ">>-" ">>=" ">=>" ">]" ">:" ">-" ">=") (+ ">"))))
           (?&  . ,(rx (+ "&")))
           (?|  . ,(rx (or (or "|->" "|||>" "||>" "|=>" "||-" "||=" "|-" "|>"
                               "|]" "|}" "|=")
                           (+ "|"))))
           (?.  . ,(rx (or (or ".?" ".=" ".-" "..<") (+ "."))))
           (?+  . ,(rx (or "+>" (+ "+"))))
           (?\[ . ,(rx (or "[<" "[|")))
           (?\{ . ,(rx "{|"))
```

```
          (?\? . ,(rx (or (or "?." "?=" "?:") (+ "?"))))
          (?#  . ,(rx (or (or "#_(" "#[" "#{" "#=" "#!" "#:" "#_" "#?" "#(")
                          (+ "#"))))
          (?\; . ,(rx (+ ";")))
          (?_  . ,(rx (or "_|_" "__")))
          (?~  . ,(rx (or "~~>" "~~" "~>" "~-" "~@")))
          (?$  . ,(rx "$>"))
          (?^  . ,(rx "^="))
          (?\] . ,(rx "]#"))))
  (let ((char (car char/ligature-re))
        (ligature-re (cdr char/ligature-re)))
    (set-char-table-range composition-function-table char
                          `([,ligature-re 0 font-shape-gstring]))))
```

6.5 Reduce clutter

Remove the toolbar. It's ugly and I never use it. Also remove the scroll bars; below, I set up the fringe to show my position in a buffer.

```
(when (window-system)
  (add-to-list 'image-types 'svg)
  (tool-bar-mode -1)
  (menu-bar-mode -1)
  (scroll-bar-mode 1))
```

When running emacs in a terminal, remove the menu bar.

```
(when (not (window-system))
  (tool-bar-mode -1)
  (load-theme 'atom-one-dark t)
  (menu-bar-mode -1))
```

6.6 Basic functionality

Default of yes or no to y n p

```
;; Y/N
(defalias 'yes-or-no-p 'y-or-n-p)
```

Clipboard functionality

```
(use-package clipetty)
(global-clipetty-mode)
```

6.7 Icons

Language and all other icons pack

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

6.8 Dashboard

Startup page using Fuyuko Mayuzumi image as banner and show bunch of useful shortcuts!

Figure 1: Fuyuko Mayuzumi

```
(use-package dashboard
  :ensure t
```

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

(setq dashboard-banner-logo-title "Personal Development Environment")
(setq dashboard-startup-banner "~/.config/bspwm/assets/greeting.png" )
(setq dashboard-center-content t)
(setq dashboard-icon-type 'all-the-icons) ;; use `all-the-icons' package
(setq dashboard-show-shortcuts nil)
(setq dashboard-image-banner-max-height 200)
(setq dashboard-agenda-time-string-format "%Y-%m-%d %H:%M")
(setq dashboard-items '((projects . 7)
                        (bookmarks . 3)
                        (agenda . 2)))
(setq initial-buffer-choice (lambda () (get-buffer "*dashboard*")))
```

6.9 Modeline

Modeline using vanilla emacs with only limited info that i really need

```
(setq display-battery-mode t)
(setq display-time-mode t)
(setq display-time t)
(setq-default mode-line-format
              '(
                ""
                "%e"
                mode-line-front-space
                mode-line-frame-identification
                mode-line-buffer-identification
                " "
                mode-line-position
                (:eval
                 (if vc-mode
                     (let* ((noback (replace-regexp-in-string (format "^ %s" (vc-backe
                            (face (cond ((string-match "^ -" noback) 'mode-line-vc)
                                        ((string-match "^ [:@]" noback) 'mode-line-vc-
                                        ((string-match "^ [!\\?]" noback) 'mode-line-v
                       (format " %s" (substring noback 2)))))
                "  "
```

```
(:eval (when (and (bound-and-true-p lsp-mode) (lsp--client-servers))
         (format "LSP[%s]" (mapconcat 'lsp--client-server-id (lsp--cli
       )
mode-line-misc-info
mode-line-end-spaces
))
```

6.10 Neotree (Filetree)

Using neotree for folder tree management

```
(use-package neotree :ensure t)
(setq neo-smart-open t)
(setq neo-theme (if (display-graphic-p) 'icons 'arrow))
(setq x-underline-at-descent-line t)
```

6.11 Tabbar

Tab bar configurations

```
(tab-bar-mode 1)                          ;; enable tab bar
(setq tab-bar-show 1)                     ;; hide bar if <= 1 tabs open
(setq tab-bar-close-button-show nil)      ;; hide tab close / X button
(setq tab-bar-new-tab-choice "*dashboard*");; buffer to show in new tabs
(setq tab-bar-tab-hints t)                ;; show tab numbers
(defun neko/current-tab-name ()
  (alist-get 'name (tab-bar--current-tab)))
```

6.12 Backup

```
(setq
 backup-by-copying t      ; don't clobber symlinks
 backup-directory-alist
 '(("." . "~/.saves/"))    ; don't litter my fs tree
 delete-old-versions t
 kept-new-versions 6
 kept-old-versions 2
 version-control t)       ; use versioned backups
```

6.13 Parenthesis

All parenthesis config here

```
(use-package smartparens
  :config
  (smartparens-global-mode 1))

(require 'paren)
(show-paren-mode t)
(setq show-paren-delay 0)
(set-face-attribute 'show-paren-match nil :background nil :underline t)
(setq show-paren-style 'parenthesis) ; can be 'expression' 'parenthesis' and 'mixed'
```

7 Keyboard related stuffs

7.1 Evil mode

Evil binding! for those who come from VIM

```
(use-package evil
  :init
  (setq evil-want-integration t)
  (setq evil-want-keybinding nil)
  (setq evil-want-C-u-scroll t)
  (setq evil-want-C-i-jump t)
  (setq evil-want-Y-yank-to-eol 1)
  :config
  (evil-mode 1)
  (define-key evil-insert-state-map (kbd "C-h") 'evil-delete-backward-char-and-join)

  ;; Use visual line motions even outside of visual-line-mode buffers
  (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))

(use-package evil-escape
  :init
  (evil-escape-mode)
```

```
  :config
  (setq-default evil-escape-key-sequence "jk")
  )

(evil-set-initial-state 'pdf-view-mode 'normal)

(evil-define-key 'normal 'lsp-ui-doc-mode
  [?K] #'lsp-ui-doc-focus-frame)
(evil-define-key 'normal 'lsp-ui-doc-frame-mode
  [?q] #'lsp-ui-doc-unfocus-frame)
```

7.2 Evil keybind collection

Evil binding collection to match with evil mode

```
(use-package evil-collection
  :ensure t
  :after evil
  :config
  (evil-collection-init))
```

7.3 Undo

Undo and redo functionality

```
(use-package undo-tree
  :ensure t
  :after evil
  :diminish
  :config
  (evil-set-undo-system 'undo-tree)
  (global-undo-tree-mode 1))
(setq undo-tree-history-directory-alist '(("." . "~/.emacs.d/undo")))
```

7.4 General keybind

General keybind for emacs using `neko/leader-keys` to bind the keyboard combination

```
(use-package general
  :ensure t)
  :config
  (general-create-definer neko/leader-keys
    :keymaps '(normal visual emacs)
    :prefix "SPC"
    :global-prefix "SPC")
(general-auto-unbind-keys t)
(define-key minibuffer-local-completion-map (kbd "SPC") 'self-insert-command)
```

7.5 Hydra

This is a package for GNU Emacs that can be used to tie related commands into a family of short bindings with a common prefix - a Hydra.

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

7.6 Which key

Key hints

```
(use-package which-key
  :diminish (which-key-mode)
  :config
  (setq which-key-idle-delay 0.3)
  (which-key-mode 1))
```

8 Helm

Emacs framework for incremental completions and narrowing selections

```
(use-package helm
  :config
  (global-set-key (kbd "M-x") #'helm-M-x)
  (global-set-key (kbd "C-x C-f") #'helm-find-files)
  )

(neko/leader-keys
  "h"  '(helm-command-prefix :which-key "Helm"))
```

9 Third party connection

9.1 Elcord

Discord RCP client

```
(use-package elcord :ensure t
  :config
  (setq elcord-display-line-numbers nil)
  (setq elcord-editor-icon "Emacs (Material)")
  )
```

9.2 EMMS

Multimedia for emacs

```
(use-package emms
  :config
  (require 'emms-setup)
  (require 'emms-player-mpd)
  (emms-all)
  (setq emms-player-list '(emms-player-mpd))
  (setq emms-player-list '(emms-player-vlc)
        emms-info-functions '(emms-info-native))
  (add-to-list 'emms-info-functions 'emms-info-mpd)
  (add-to-list 'emms-player-list 'emms-player-mpd)
  (setq emms-player-mpd-server-name "localhost")
  (setq emms-player-mpd-server-port "6600")
  :custom
  (emms-source-file-default-directory "~/Music/")
  :bind
  (("<f5>" . emms-browser)
   ("<M-f5>" . emms)
   ("<XF86AudioPrev>" . emms-previous)
   ("<XF86AudioNext>" . emms-next)
   ("<XF86AudioPlay>" . emms-pause))
  )
```

10 Programming

10.1 Treesitter

Highlighting for emacs

```
(use-package tree-sitter)
(use-package tree-sitter-langs)
(global-tree-sitter-mode)
```

10.2 LSP Mode

LSP backend for every language i use

```
(use-package lsp-mode
  :init
  ;; set prefix for lsp-command-keymap (few alternatives - "C-l", "C-c l")
  (setq lsp-keymap-prefix "C-c l")
  (setq lsp-lens-enable t)
  (setq lsp-ui-sideline-enable t)

  :custom
  (lsp-completion-provider :none)
  (lsp-completion-show-detail t)
  (lsp-completion-show-kind t)
  (lsp-eldoc-enable-hover t)
  (lsp-eldoc-render-all nil)
  (lsp-enable-file-watchers t)
  (lsp-enable-imenu t)
  (lsp-enable-symbol-highlighting t)
  (lsp-enable-xref t)
  (lsp-headerline-breadcrumb-enable t)
  (lsp-idle-delay 0.4)
  (lsp-lens-enable t)
  (lsp-modeline-diagnostics-enable t)
  (lsp-semantic-tokens-apply-modifiers t)
  (lsp-semantic-tokens-enable t)
  (lsp-semantic-tokens-warn-on-missing-face nil)
  (lsp-signature-auto-activate t)
  (lsp-signature-render-documentation t)
```

```
  :hook (
         (lsp-mode . lsp-enable-which-key-integration))
  :commands lsp)

;; optionally
(use-package lsp-ui :commands lsp-ui-mode)
;; if you are helm user
(use-package helm-lsp :commands helm-lsp-workspace-symbol)
;; if you are ivy user
(use-package lsp-ivy :commands lsp-ivy-workspace-symbol)

;; optionally if you want to use debugger
(use-package dap-mode)
;; (use-package dap-LANGUAGE) to load the dap adapter for your language
(setq-default ;; alloc.c
 gc-cons-threshold (* 20 1204 1204)
 gc-cons-percentage 0.5)
(setq gc-cons-threshold (* 2 1000 1000)) ; Increase the garbage collection threshold


;; optional if you want which-key integration
(use-package which-key
  :config
  (which-key-mode))
```

10.3 LSP UI

LSP UI for more beautiful looks

```
(use-package lsp-ui
  :ensure t

  :custom
  (lsp-ui-doc-alignment 'frame)
  (lsp-ui-doc-delay 0.2)
  (lsp-ui-doc-enable t)
  (lsp-ui-doc-header nil)
  (lsp-ui-doc-include-signature t)
  (lsp-ui-doc-max-height 45)
```

```
  (lsp-ui-doc-position 'top)
  (lsp-ui-doc-show-with-cursor nil)
  (lsp-ui-doc-show-with-mouse nil)
  (lsp-ui-doc-use-webkit nil)
  (lsp-ui-peek-always-show nil)
  (lsp-ui-sideline-enable t)
  (lsp-ui-sideline-show-code-actions t)
  (lsp-ui-sideline-show-diagnostics t)
  (lsp-ui-sideline-show-hover nil)

  :config
  (setq lsp-ui-sideline-ignore-duplicate t)
  (add-hook 'lsp-mode-hook 'lsp-ui-mode)
  )

(defun my/lsp-ui-doc-scroll-up ()
  "Scroll up inside LSP UI documentation frame."
  (interactive)
  (scroll-up-command))

(defun my/lsp-ui-doc-scroll-down ()
  "Scroll down inside LSP UI documentation frame."
  (interactive)
  (scroll-down-command))

(defun my/setup-lsp-ui-doc-scroll-keys ()
  "Set up custom scroll keys for LSP UI documentation frame."
  (local-set-key (kbd "j") 'my/lsp-ui-doc-scroll-down)
  (local-set-key (kbd "k") 'my/lsp-ui-doc-scroll-up))

(add-hook 'lsp-ui-doc-frame-mode-hook 'my/setup-lsp-ui-doc-scroll-keys)
```

10.4 LSP Mapping

Map using `Which key` for LSP related key bindings

```
(neko/leader-keys
  "l"  '(:ignore t :which-key "LSP")
  "lg" '(lsp-goto-type-definition :which-key "Go to definition")
```

```
  "li" '(lsp-goto-implementation :which-key "Go to implementation")
  "lc" '(helm-lsp-code-actions :which-key "Code action")
  "ll" '(lsp-avy-lens :which-key "Code lens")
  "lr" '(lsp-rename :which-key "Code lens")
  "lf" '(lsp-format-buffer :which-key "Format buffer")
  "lD" '(lsp-ui-peek-find-definitions :which-key "Goto definition")
  "ld" '(lsp-find-declaration :which-key "Find declaration")
  "lt" '(lsp-find-type-definition :which-key "Find type")
  "le" '(helm-lsp-diagnostics :which-key "Error diagnostics")
  "ls" '(lsp-signature-help :which-key "Signature help")
  "lR" '(lsp-find-references :which-key "Find references")
  "lm" '(lsp-ui-imenu :which-key "Imenu")
  "lo" '(lsp-describe-thing-at-point :which-key "Documentation")
  "la" '(lsp-ui-peek-find-implementation :which-key "Code implement"))
```

10.5 DAP Mode

DAP Debugging for some languages

```
(use-package dap-mode
         :ensure t
         ;; Uncomment the config below if you want all UI panes to be hidden by defaul
         ;; :custom
         ;; (lsp-enable-dap-auto-configure nil)
         ;; :config
         ;; (dap-ui-mode 1)
         :commands dap-debug
         :config
         (dap-tooltip-mode 1)
         ;; use tooltips for mouse hover
         ;; if it is not enabled `dap-mode' will use the minibuffer.
         (tooltip-mode 1)
         ;; displays floating panel with debug buttons
         ;; requies emacs 26+
         (dap-ui-controls-mode 1)
         ;; Set up Node debugging
         (require 'dap-hydra)
         (general-define-key
```

```
          :keymaps 'lsp-mode-map
          :prefix lsp-keymap-prefix
          "d" '(dap-hydra t :wk "debugger"))
         )

(neko/leader-keys
  "d"  '(:ignore t :which-key "Debugging")
  "ds" '(dap-debug t :wk "Start debug")
  "db" '(dap-breakpoint-toggle t :wk "Toggle breakpoint")
  "dd" '(dap-hydra t :wk "Debugger"))
```

10.6 Snippets

Template system for emacs

```
(use-package company
  :after lsp-mode
  :hook (lsp-mode . company-mode)
  :bind (
         :map company-active-map
         ("<tab>" . company-complete-common-or-cycle)
         ("C-j" . company-select-next-or-abort)
         ("C-k" . company-select-previous-or-abort)
         ("C-l" . company-other-backend)
         ("C-h" . nil)
         )
  (:map lsp-mode-map
        ("<tab>" . company-indent-or-complete-common))
  :custom
  (company-minimum-prefix-length 1)
  (company-idle-delay 0.0)
  :config
  (add-hook 'after-init-hook 'global-company-mode)
  )

(setq company-backends '((company-capf company-yasnippet)))

(use-package company-box
  :hook (company-mode . company-box-mode))
```

10.7 Flycheck

Flycheck an error or info

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

10.8 Yasnippets

Snippets for emacs

```
(use-package yasnippet :ensure t
  :config
  ;; (add-to-list 'yas-snippet-dirs "~/.emacs.d/snippets/yasnippet-golang/")
  (setq yas-snippet-dirs
        '("~/.emacs.d/snippets"))
        (yas-global-mode 1)
        )
```

10.9 Languages

10.9.1 GO

GO configuration language using `gopls`

```
(use-package go-mode
  :ensure t
  :init
  (with-eval-after-load 'projectile
    (add-to-list 'projectile-project-root-files-bottom-up "go.mod")
    (add-to-list 'projectile-project-root-files-bottom-up "go.sum"))
  :hook
  ((go-mode . lsp-deferred)
   (go-mode . flycheck-mode)
   (go-mode . company-mode))
  :config
  (setq gofmt-command "gofmt")
  (require 'lsp-go))

(defun lsp-go-install-save-hooks ()
  (add-hook 'before-save-hook #'lsp-format-buffer t t)
  (add-hook 'before-save-hook #'lsp-organize-imports t t))
(add-hook 'go-mode-hook #'lsp-go-install-save-hooks)
```

```
(add-hook 'go-mode-hook #'lsp-deferred)
(add-hook 'go-mode-hook #'yas-minor-mode)

(add-hook 'go-mode-hook (lambda()
                          (flycheck-golangci-lint-setup)
                          (setq flycheck-local-checkers '((lsp . ((next-checkers . (go
```

Flycheck for go

```
(use-package flycheck-golangci-lint
  :hook (go-mode . flycheck-golangci-lint-setup))

(defun my/get-config-path (config-file-name)
  "get the path to config-file-name in the current project as a string, when in `go-mo
  (when (eq major-mode 'go-mode)
    (let* ((project-root (projectile-project-root))
           (config-path (concat project-root config-file-name)))
      (if (file-exists-p config-path)
          config-path
        (error "configuration file '%s' not found in project root '%s'" config-file-na

(setq my/config-path (my/get-config-path ".golangci.yaml"))
(setq flycheck-golangci-lint-config my/config-path)
```

GO tags, Gen test and doctor

```
(use-package go-add-tags :ensure t)
(use-package go-gen-test :ensure t)
(use-package godoctor)
```

GO Snippets

```
(use-package go-snippets)
```

Go treesitter

```
(add-to-list 'tree-sitter-major-mode-language-alist
             '(go-mode . go))

;; Hook Tree-sitter mode into your major modes
(add-hook 'go-mode-hook #'tree-sitter-hl-mode)
```

10.9.2 Javascript

1. RJSX

   This mode derives from js2-mode, extending its parser to support JSX syntax according to the official spec. This means you get all of the js2 features plus proper syntax checking and highlighting of JSX code blocks.

   ```
   (use-package rjsx-mode)
   (add-hook 'js-mode-hook #'lsp)
   ```

2. React snippet

   ReactJS snippets

   ```
   (use-package react-snippets)
   ```

10.9.3 Typescript

Typescript mode

```
(use-package typescript-mode
  :after tree-sitter
  :config
  (define-derived-mode typescriptreact-mode typescript-mode
    "TypeScript TSX")

  ;; use our derived mode for tsx files
  (add-to-list 'auto-mode-alist '("\\.tsx?\\'" . typescriptreact-mode))
  ;; by default, typescript-mode is mapped to the treesitter typescript parser
  ;; use our derived mode to map both .tsx AND .ts -> typescriptreact-mode -> treesitt
  (add-to-list 'tree-sitter-major-mode-language-alist '(typescriptreact-mode . tsx)))

(add-hook 'typescript-mode-hook #'lsp)
```

10.9.4 Java

Java configuration for Spring boot etc

```
(use-package lsp-java
  :if (executable-find "mvn")
  :hook
  ((java-mode . company-mode)
   (java-mode . flycheck-mode)
   (java-mode . lsp-deferred))
  :config
  (use-package request :defer t)
  (add-hook 'java-mode-hook #'lsp-java-boot-lens-mode)
  (setq lsp-java-java-path
        "/usr/lib/jvm/java-17-openjdk-amd64/bin/java")
  :custom
  (lsp-java-server-install-dir (expand-file-name "~/.emacs.d/eclipse.jdt.ls/server/"))
  (lsp-java-workspace-dir (expand-file-name "~/.emacs.d/eclipse.jdt.ls/workspace/")))
(require 'lsp-java-boot)

(add-to-list 'tree-sitter-major-mode-language-alist
             '(java-mode . java))

(add-hook 'java-mode-hook #'tree-sitter-hl-mode)
```

10.9.5 Scala

Scala

```
(use-package scala-mode
  :interpreter ("scala" . scala-mode)
  :hook
  ((scala-mode . company-mode)
   (scala-mode . lsp-deferred))
  )
(use-package lsp-metals)
```

10.9.6 Yaml

```
(use-package yaml-mode
```

```
  :mode "\\.ya?ml\\'")
(add-hook 'yaml-mode-hook
          (lambda ()
            (define-key yaml-mode-map "\C-m" 'newline-and-indent)))
```

10.9.7 Dart

Dart for mobile development

```
(setq package-selected-packages
      '(dart-mode lsp-mode lsp-dart lsp-treemacs flycheck company
                  ;; Optional packages
                  lsp-ui company hover))

(use-package dart-mode)

;; export ANDROID_HOME=$HOME/Android
;; export PATH=$ANDROID_HOME/cmdline-tools/tools/bin/:$PATH
;; export PATH=$ANDROID_HOME/platform-tools/:$PATH

;; export PATH="$PATH:$HOME/Android/flutter/bin/"


(setq lsp-dart-sdk-dir "~/flutter/bin/cache/dart-sdk/")


(add-hook 'dart-mode-hook 'lsp)
```

Dart LSP

```
(use-package lsp-dart)
(use-package hover)
```

Flutter

```
(use-package flutter
  :after dart-mode
  :custom
  (flutter-sdk-path "~/flutter/"))
```

10.9.8 Rust

Rust setup

```
(use-package rust-mode
  :hook
  ((rust-mode . company-mode)
   (rust-mode . lsp-deferred))
  :config
  (setq lsp-rust-server 'rust-analyzer)
  (setq rust-format-on-save t)
  (add-hook 'rust-mode-hook
            (lambda () (setq indent-tabs-mode nil)))
  (add-hook 'rust-mode-hook
            (lambda () (prettify-symbols-mode)))
  )

(use-package cargo-mode
  :config
  (add-hook 'rust-mode-hook 'cargo-minor-mode))

(add-to-list 'tree-sitter-major-mode-language-alist
             '(rust-mode . rust))

;; Hook Tree-sitter mode into your major modes
(add-hook 'rust-mode-hook #'tree-sitter-hl-mode)
```

10.9.9 Web

Web related languages, PHP, etc

```
(use-package php-mode
  :hook
  (php-mode . lsp-deffered)
  (php-mode . company-mode))
(add-to-list 'auto-mode-alist '("\\.blade.php\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.phtml\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.blade\\'" . php-mode))
(add-to-list 'auto-mode-alist '("\\.php\\'" . php-mode))
(use-package phpunit)
```

10.10 Smerge

Merge make it easy with smerge-mode

```
(use-package hydra)
(use-package smerge-mode
  :config
  (defhydra hydra-smerge (:color red :hint nil)
    "
Navigate       Keep               other
----------------------------------------
_j_: previous  _RET_: current       _e_: ediff
_k_: next      _m_: mine  <<       _u_: undo
_j_: up        _o_: other >>       _r_: refine
_k_: down      _a_: combine        _q_: quit
               _b_: base
"
    ("k" smerge-next)
    ("j" smerge-prev)
    ("RET" smerge-keep-current)
    ("m" smerge-keep-mine)
    ("o" smerge-keep-other)
    ("b" smerge-keep-base)
    ("a" smerge-keep-all)
    ("e" smerge-ediff)
    ("J" previous-line)
    ("K" forward-line)
    ("r" smerge-refine)
    ("u" undo)
    ("q" nil :exit t))

  (defun enable-smerge-maybe ()
    (when (and buffer-file-name (vc-backend buffer-file-name))
      (save-excursion
        (goto-char (point-min))
        (when (re-search-forward "^<<<<<<< " nil t)
          (smerge-mode +1)
          (hydra-smerge/body))))))


(neko/leader-keys
```

```
    "gm"  '(scimax-smerge/body :which-key "Toggle smerge")
    )
```

10.11 Projectile

Project wide system management

```
(use-package projectile
  :diminish projectile-mode
  :config (projectile-mode)
  :custom ((projectile-completion-system 'ivy)))
  ;; NOounsTE: Set this to the folder where you keep your Git repos!

  (neko/leader-keys
    "f"  '(:ignore t :which-key "Projectile Find")
    "fs" '(projectile-grep :which-key "Find String")
    "fr" '(projectile-replace :which-key "Find Replace")
    "fa" '(projectile-replace-regexp :which-key "Find Replace Project")
    "ft" '(projectile-find-test-file :which-key "Find Tests")
    "ff" '(projectile-find-file :which-key "Find File"))
```

10.12 VTerm

Terminal emulator

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

Open at bottom

```
(defun vterm-split-window-below ()
  (interactive)
  (vterm)
  (split-window-below -12)
  (previous-buffer)
  (other-window 1))

(defun vterm-toggle ()
  (interactive)
  (if (eq major-mode 'vterm-mode)
```

```
      (delete-window)
    (vterm-split-window-below)))

(neko/leader-keys
  "ot" '(vterm-toggle :which-key "Open Vterm"))
```

10.13 Magit

Git inside emacs!

```
(use-package magit
  :ensure t
  :bind ("C-x g" . magit-status)
  :custom
  (magit-displey-buffer-function #'magit-display-buffer-same-window-except-diff-v1))

(defun open-magit-in-vertical-split ()
  (interactive)
  (magit-status))

(neko/leader-keys
    "g" '(:ignore t :which-key "Git")
    "gs" '(open-magit-in-vertical-split :which-key "Magit"))
```

10.14 Tests

10.14.1 Jest

Used for jest unit test TS/JS

```
(use-package jest)
```

10.15 Container

Container Docker config

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

(neko/leader-keys
  "c"  '(:ignore t :which-key "Container")
  "cd" '(docker-compose :which-key "Docker Compose")
```

```
  "ci" '(docker-images :which-key "Docker Images")
  )

(use-package dockerfile-mode)
```

10.15.1 Kubernetes

Kubernetes setups

```
(use-package kubernetes
  :ensure t
  :commands (kubernetes-overview)
  :config
  (setq kubernetes-poll-frequency 3600
        kubernetes-redraw-frequency 3600))
```

K8s Setup

```
(use-package k8s-mode
  :ensure t
  :hook (k8s-mode . yas-minor-mode))
```

10.16 Blamer

Blamer

```
(use-package blamer
  :ensure t
  :bind (("s-i" . blamer-show-commit-info)
         ("C-c i" . blamer-show-posframe-commit-info))
  :defer 20
  :custom
  (blamer-idle-time 0.3)
  (blamer-min-offset 70)
  :custom-face
  (blamer-face ((t :foreground "#7a88cf"
                   :background nil
                   :height 100
                   :italic t))))
```

11 Writing

All of writing stuff

11.1 Org Mode

Org mode style writing

```
(use-package org
  :config
  (setq org-ellipsis " ...")
  (setq org-agenda-files
        '("~/Orgs/")))

(defun nolinum ()
  (global-display-line-numbers-mode 0))

(add-hook 'org-mode-hook 'nolinum)

(global-set-key (kbd "C-c c") #'org-capture)


(setq org-capture-templates '(("t" "Todo [inbox]" entry
                               (file+headline "~/Orgs/inbox.org" "Tasks")
                               "* TODO %i%?")
                              ("T" "Tickler" entry
                               (file+headline "~/Orgs/tickler.org" "Tickler")
                               "* %i%? \n %U")
                              ("b" "Braindump" entry
                               (file+headline "~/Orgs/braindump.org" "Braindump")
                               "* %? \n")))

(setq org-refile-targets '(("~/Orgs/agenda.org" :maxlevel . 3)
                           ("~/Orgs/someday.org" :level . 1)
                           ("~/Orgs/tickler.org" :maxlevel . 2)))

(setq org-todo-keywords
      '((sequence "TODO(t)" "WAITING(n)" "|" "DONE(d)" "CANCEL(c)")))


(setq org-agenda-breadcrumbs-separator "  "
```

```
      org-agenda-current-time-string "  now"
      org-agenda-time-grid '((weekly today require-timed)
                             (800 1000 1200 1400 1600 1800 2000)
                             "---" "")
      org-agenda-prefix-format '((agenda . "%i %-12:c%?-12t%b% s")
                                 (todo . " %i %-12:c")
                                 (tags . " %i %-12:c")
                                 (search . " %i %-12:c")))

(setq org-agenda-format-date (lambda (date) (concat "\n" (make-string (window-width) 9
                                                     "\n"
                                                     (org-agenda-format-date-aligned da
(setq org-cycle-separator-lines 2)
(setq org-agenda-category-icon-alist
      `(("Work" ,(list (all-the-icons-faicon "cogs")) nil nil :ascent center)
        ("Personal" ,(list (all-the-icons-material "person")) nil nil :ascent center)
        ("Calendar" ,(list (all-the-icons-faicon "calendar")) nil nil :ascent center)
        ("Reading" ,(list (all-the-icons-faicon "book")) nil nil :ascent center)))
```

11.2 Org babel

Execute script inside org `#+begin_src lang`

```
(setq org-babel-python-command "python3")
(org-babel-do-load-languages
 'org-babel-load-languages
 '((python . t)
   (shell . t)
   (js . t)
   (latex . t)
   (calc . t)
   ))
```

11.3 Org bullet

Beautify bulletpoint

```
(use-package org-bullets
  :after org
  :hook (org-mode . org-bullets-mode)
  :custom
```

```
  (org-bullets-bullet-list '("" "" "" "" "" "" "")))
```

11.4 Org save tangle

Auto tangle if saved

```
(defun neko/org-babel-tangle-config ()
  (when (string-equal (buffer-file-name)
                      (expand-file-name "~/.emacs.d/README.org"))
    (let ((org-confirm-babel-evaluate nil))
      (org-babel-tangle))))

(add-hook 'org-mode-hook (lambda () (add-hook 'after-save-hook #'neko/org-babel-tangle
```

11.5 Calfw

Beautiful calendar

```
(use-package calfw :ensure t
  :config
  (setq cfw:face-item-separator-color nil
        cfw:render-line-breaker 'cfw:render-line-breaker-none
        cfw:fchar-junction ?
        cfw:fchar-vertical-line ?
        cfw:fchar-horizontal-line ?
        cfw:fchar-left-junction ?
        cfw:fchar-right-junction ?
        cfw:fchar-top-junction ?
        cfw:fchar-top-left-corner ?
        cfw:fchar-top-right-corner ?)
  )
(use-package calfw-org :ensure t
  :commands (cfw:open-org-calendar
             cfw:org-create-source
             cfw:org-create-file-source
             cfw:open-org-calendar-withkevin)
  )
(require 'calfw-org)
```

11.6 LaTeX

All latex retaled writing stuffs

```
(use-package org-fragtog)

(add-hook 'org-mode-hook
          'org-fragtog-mode)
```

:foreground default :background default :scale 3.0 :html-foreground Black :html-backgro

12 Workspace

Workspaces related configuration

12.1 Prespective-el

Prespective for each workspaces

```
(use-package perspective
  :bind
  ("C-x k" . persp-kill-buffer*)         ; or use a nicer switcher, see below
  :custom
  (persp-mode-prefix-key (kbd "C-c M-p"))  ; pick your own prefix key here
  :init
  (persp-mode))

(neko/leader-keys
  "p"  '(:ignore t :which-key "Perspective")
  "ps" '(persp-switch :which-key "Switch perspective")
  "pm" '(persp-merge :which-key "Merge perspective")
  "pb" '(persp-list-buffers :which-key "Buffers perspective")
  "pk" '(persp-kill :which-key "Kill perspective")
  )

(neko/leader-keys
  "]" '(persp-next :which-key "Next perspective")
  "[" '(persp-prev :which-key "Prev perspective")
  )
```

13 Keybinding

13.1 Basic Keybinding

13.1.1 Window keybind

Window movement and management keybind

```
(defun reload-dotemacs ()
  (interactive)
  (load-file "~/.emacs.d/init.el"))
(neko/leader-keys
  "<left>" '(tab-bar-switch-to-prev-tab :which-key "Prev Tab")
  "<right>" '(tab-bar-switch-to-next-tab :which-key "Next Tab")
  "n" '(tab-bar-new-tab :which-key "New Tab")
  "w"  '(:ignore t :which-key "Window")
  "ws" '(evil-window-split :which-key "Split")
  "wj" '(evil-window-down :which-key "Go Bottom")
  "wk" '(evil-window-up :which-key "Go Top")
  "wh" '(evil-window-left :which-key "Go Left")
  "wl" '(evil-window-right :which-key "Go Right")
  "wv" '(evil-window-vsplit :which-key "Vsplit")
  "wq" '(delete-window :which-key "Quit")
  "wr" '(reload-dotemacs :which-key "Reload")
  "wb" '(switch-to-buffer :which-key "Switch Buffer"))
```

13.1.2 General keybind

Generic wide system keybinding

```
(neko/leader-keys
  ";" '(helm-M-x :which-key "Meta")
  "/" '(comment-region :which-key "Comment region")
  "s" '(evil-save :which-key "Save")
  "b" '(:ignore t :which-key "Buffer")
  "bb" '(counsel-switch-buffer :which-key "Switch buffer")
  "bk" '(kill-buffer :which-key "Kill buffer")
  "qq" '(kill-buffer-and-window :which-key "Kill buffer")
  "ba" '(kill-other-buffers :which-key "Kill other buffer except this"))
```

13.1.3 Open keybind

```
(neko/leader-keys
  "o" '(:ignore t :which-key "Open")
  "oa" '(org-agenda :which-key "Org Agenda")
  "oc" '(cfw:open-org-calendar :which-key "Calendar")
  "oe" '(neotree :which-key "Neotree")
  "od" '(dashboard-open :which-key "Dashboard")
  "oD" '(dired :which-key "Dired"))
```

If $a^2 = b$ and $b = 2$, then the solution must be either

$$a = +\sqrt{2}$$

or

$$a = -\sqrt{2}$$

.