

# My Emacs config

Jay Lee

June 22, 2022

## Contents

<b>1</b>	<b>Package management</b>	<b>2</b>
1.1	MELPA . . . . .	2
1.2	Better package setup . . . . .	2
1.3	Modernized package menu . . . . .	2
<b>2</b>	<b>Pretty Emacs</b>	<b>2</b>
2.1	Theme . . . . .	2
2.2	Font . . . . .	3
2.3	Icons . . . . .	3
2.4	Modeline Theme . . . . .	3
2.5	File explorer . . . . .	3
2.6	Miscellaneous . . . . .	4
<b>3</b>	<b>Key bindings</b>	<b>4</b>
3.1	Hints . . . . .	4
3.2	Navigation . . . . .	5
3.3	Miscellaneous . . . . .	5
<b>4</b>	<b>Languages</b>	<b>5</b>
4.1	Language server protocol . . . . .	5
4.2	Lisps . . . . .	6
4.2.1	Scheme . . . . .	6
4.3	OCaml . . . . .	6
4.4	Python . . . . .	7
4.5	Org mode . . . . .	7
4.5.1	LaTeX . . . . .	8
4.6	Miscellaneous . . . . .	8
<b>5</b>	<b>Git</b>	<b>8</b>
<b>6</b>	<b>Dired</b>	<b>8</b>

## 1 Package management

### 1.1 MELPA

```
(require 'package)
(add-to-list 'package-archives
  '("melpa" . "https://melpa.org/packages/") t)
(package-initialize)
```

### 1.2 Better package setup

Use use-package

```
(unless (package-installed-p 'use-package)
  (package-refresh-contents)
  (package-install 'use-package))
(eval-and-compile
  ; Always make sure packages are installed correctly at every startup
  (setq use-package-always-ensure t
        use-package-expand-minimally t))
```

### 1.3 Modernized package menu

Use paradox.

```
(use-package paradox
  :ensure t
  :init (paradox-enable))
```

## 2 Pretty Emacs

### 2.1 Theme

Although Atom is being sunset by GitHub, the default theme it provided is one of the best-looking and easy-on-the-eyes color scheme out in the wild. doom-one theme is more actively maintained than atom-one-dark-theme.

```
(use-package doom-themes
  :ensure t
  :config
  (setq doom-themes-enable-bold t
        doom-themes-enable-italic t)
  (load-theme 'doom-one t))
```

## 2.2 Font

JuliaMono provides the most comprehensive unicode support.

```
(set-frame-font "JuliaMono 15" nil t)
; Apply the font to emacsclients
(add-to-list 'default-frame-alist '(font . "JuliaMono 15"))
```

Use D2Coding for hangul (Korean letters).

```
(set-fontset-font t 'hangul (font-spec :name "D2Coding"))
(setq face-font-rescale-alist '(("D2Coding" . 1.2)))
```

D2Coding is scaled so that a single Korean letter corresponds to two ASCII letters:

English	Korean
Emacs	이맥스
Vim	빔

## 2.3 Icons

```
(use-package all-the-icons
  :ensure t
  :if (display-graphic-p))
```

## 2.4 Modeline Theme

```
(use-package doom-modeline
  :ensure t
  :hook (after-init . doom-modeline-mode))
```

## 2.5 File explorer

A Tree layout file explorer.

```
(use-package treemacs
  :ensure t
  :bind
  (:map global-map
    ("M-0" . treemacs-select-window)
    ("C-x t 1" . treemacs-delete-other-windows)
    ("C-x t t" . treemacs)
    ("C-x t d" . treemacs-select-directory)
    ("C-x t B" . treemacs-bookmark)
    ("C-x t C-t" . treemacs-find-file)
    ("C-x t M-t" . treemacs-find-tag)))
```

```
(use-package treemacs-icons-dired
  :hook (dired-mode . treemacs-icons-dired-enable-once)
  :ensure t)
```

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

## 2.6 Miscellaneous

Seamless title bar in macOS.

```
(tool-bar-mode -1)
(use-package ns-auto-titlebar
  :ensure t
  :config (ns-auto-titlebar-mode))
```

Line numbering.

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

Get rid of irritating beep-boops.

```
(defun flash-mode-line ()
  (invert-face 'mode-line)
  (run-with-timer 0.1 nil #'invert-face 'mode-line))
(setq visible-bell nil ring-bell-function #'flash-mode-line)
```

Prettify symbols.

```
(global-prettify-symbols-mode 1)
```

## 3 Key bindings

### 3.1 Hints

Show what key bindings are available.

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

Show completions.

```
(use-package ivy
  :ensure t
  :config
  (ivy-mode)
  (setq ivy-use-virtual-buffers t))
```

## 3.2 Navigation

Incremental search using ivy.

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

Use numbering to move frames.

```
(use-package window-numbering
  :ensure t
  :config (window-numbering-mode))
```

## 3.3 Miscellaneous

Use command as meta in macOS.

```
(setq mac-command-modifier 'meta)
```

# 4 Languages

## 4.1 Language server protocol

Settings for LSP.

```
(use-package lsp-mode
  :ensure t
  :init (setq lsp-keymap-prefix "C-c l")
  :hook
  ((tuareg-mode . lsp)
   (lsp-mode . lsp-enable-which-key-integration))
  :commands lsp)
```

```
(use-package lsp-ui
  :ensure t
  :after lsp-mode)
```

```
(use-package lsp-ivy
  :ensure t
  :after (lsp-mode ivy)
  :commands lsp-ivy-workspace-symbol)
```

```
;; (use-package lsp-treemacs
;;   :ensure t
;;   :after (lsp-mode treemacs)
;;   :commands lsp-treemacs-errors-list)
```

## 4.2 Lisps

Pseudo-structural editing.

```
(use-package paredit
  :ensure t
  :init
  (autoload 'enable-paredit-mode "paredit"
    "Turn on pseudo-structural editing of Lisp code."
    t)
  :config
  (add-hook 'emacs-lisp-mode-hook #'enable-paredit-mode)
  (add-hook 'eval-expression-minibuffer-setup-hook #'enable-paredit-
mode)
  (add-hook 'ielm-mode-hook #'enable-paredit-mode)
  (add-hook 'lisp-mode-hook #'enable-paredit-mode)
  (add-hook 'lisp-interaction-mode-hook #'enable-paredit-mode)
  (add-hook 'scheme-mode-hook #'enable-paredit-mode))

Prettify lambda.

(defun prettify-lambda ()
  "Prettify lambda"
  (push '("lambda" . 955) prettify-symbols-alist))
```

### 4.2.1 Scheme

Set scheme interpreter to Chicken Scheme.

```
(setq scheme-program-name "csi")

Use geiser.

(use-package geiser-chicken
  :ensure t)

Prettify symbols.

(add-hook 'scheme-mode-hook #'prettify-lambda)
```

## 4.3 OCaml

These packages are installed via ‘opam’, not from MELPA.

```
(require 'opam-user-setup "~/.emacs.d/opam-user-setup.el")
(use-package ocamlformat
  :ensure nil
  :custom (ocamlformat-enable 'enable-outside-detected-project)
  :bind (:map tuareg-mode-map
    ("C-M-<tab>" . ocamlformat))
  :hook (before-save . ocamlformat-before-save))
```

## 4.4 Python

Use elpy.

```
(use-package elpy
  :ensure t
  :init (elpy-enable))
```

## 4.5 Org mode

Font size and symbols.

```
(use-package org-superstar
  :ensure t
  :config
  ;; hide #+TITLE:
  (setq org-hidden-keywords '(title))
  ;; set basic title font
  (set-face-attribute 'org-level-8 nil :weight 'bold :inherit 'default)
  ;; Low levels are unimportant = no scaling
  (set-face-attribute 'org-level-7 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-6 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-5 nil :inherit 'org-level-8)
  (set-face-attribute 'org-level-4 nil :inherit 'org-level-8)
  ;; Top ones get scaled the same as in LaTeX (\large, \Large, \LARGE)
  (set-face-attribute 'org-level-3 nil :inherit 'org-level-8 :height 1.2) ;\large
  (set-face-attribute 'org-level-2 nil :inherit 'org-level-8 :height 1.44) ;\Large
  (set-face-attribute 'org-level-1 nil :inherit 'org-level-8 :height 1.728) ;\LARGE
  ;; Only use the first 4 styles and do not cycle.
  (setq org-cycle-level-faces nil)
  (setq org-n-level-faces 4)
  ;; Document Title, (\huge)
  (set-face-attribute 'org-document-title nil
    :height 2.074
    :foreground 'unspecified
    :inherit 'org-level-8)
  (add-hook 'org-mode-hook (lambda () (org-superstar-mode 1))))
```

Prettify symbols.

```
(add-hook
 'org-mode-hook
 (lambda ()
  "Prettify Org mode symbols"
  (push '("[ ]" . " ") prettify-symbols-alist)
  (push '("[X]" . " ") prettify-symbols-alist)
  (push '("[-]" . " ") prettify-symbols-alist)))
```

Do not open a new window when editing source.

```
(setq org-src-window-setup 'current-window)
```

Babel.

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((scheme . t)
  (python . t)))
(setq org-confirm-babel-evaluate nil)
```

#### 4.5.1 $\LaTeX$

```
(use-package ox
 :ensure nil
 :config
 (setq org-format-latex-options
 (plist-put org-format-latex-options :scale 1.5))
 (setq org-latex-create-formula-image-program 'dvisvgm)
 (setq org-preview-latex-default-process 'dvisvgm))
```

### 4.6 Miscellaneous

Visually match parentheses.

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

## 5 Git

Use Magit.

```
(use-package magit
 :ensure t
 :bind (("C-c g" . magit-file-dispatch))) ; instead of C-c M-g, as recommended by the manual
```

## 6 Dired

```
(use-package dired
 :ensure nil
 :config (setq dired-kill-when-opening-new-dired-buffer t))
```



## 7 Terminal and shell

Use vterm.

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