Config

June 7, 2022

Contents

1 Config

org-tangle emacs config .org

2 Bookmarks

```
(map! :leader
      (:prefix ("b". "buffer")
       :desc "List bookmarks" "L" #'list-bookmarks
       :desc "Save current bookmarks to bookmark file" "w" #'bookmark-save))
```

3 key-bindings

3.1 org

```
;;not classified
(map! (:leader
  (:desc "org-table" "s h" #'eshell
   :desc "zoom" "z z" #'+hydra/text-zoom/body
   :desc "org-table" "t o" #'org-table-create
   :desc "elgantt ""g n" #'elgantt-open
   :desc "find-file" "f f" #'find-file)))

;;org slide
(map! (:leader
      (:desc "tangle" "o t" #'org-babel-tangle
       :desc "org-slide-start" "o s s" #'org-tree-slide-mode
       :desc "org-slide-right" "o s l" #'org-tree-slide-move-next-tree)))
```

3.2 screen

```
;;layout
(map! (:leader
       (:desc "" "l 1" #'split-screen-1
        :desc "clisp" "l c l" #'split-screen-3
        :desc "contestABC" "l c o" #'split-screen-4)))

;;ace-window
(setq aw-keys '(?a ?s ?d ?f ?g ?h ?j ?k ?l))
(map! :leader
      :desc "ace-window" "a c" #'ace-window)
;;smart toggle
(map! :leader
      :desc "imenu-list"
      "t l" #'imenu-list-smart-toggle)
;;visual line of numbers
```

3.3 snippet

```
(map! (:leader
      (:desc "snippets-find" "s n o" #'+snippets/find
      :desc "snippets-insert" "s n i" #'+snippets/new
      :desc "snippets-edit" "s n e" #'+snippets/edit)))
```

3.4 text-edit

```
(map! :leader
      :desc "heml kill ring"
      "k r" #'helm-show-kill-ring)

(defun move-line-up ()
  "Move up the current line."
  (interactive)
  (transpose-lines 1)
  (forward-line -2)
  (indent-according-to-mode))

(defun move-line-down ()
  "Move down the current line."
  (interactive)
  (forward-line 1)
  (transpose-lines 1)
  (forward-line -1)
  (indent-according-to-mode))

(map! (:leader
      (:desc "line-swap-down" "l d"#'move-line-down
       :desc "line-swap-down" "l u"#'move-line-up)))
```

3.5 operating somethign

```
(map! :leader
      :desc "man page"
      "d c"#'man)
```

3.6 lisp

```
(map! :leader :desc "run sly" "a a" #'sly)

(map! (:leader
       (:desc "sexp-forward" "s x f" #'sp-forward-sexp
        :desc "sexp-backward" "s x b" #'sp-backward-sexp
        :desc "sexp-kill" "s x d" #'sp-kill-sexp
        :desc "sexp-kill" "s x s" #'+default/search-other-project)))
```

3.7 godot

```
(map! :leader
      :desc  "hydra gd"
      "g d"#'gdscript-hydra-show)
```

3.8 latex

```
(map! :leader
      :desc "latex-preview"
      "l p"#'org-latex-preview)
(setq org-export-latex-coding-system 'shift_jis)
(setq org-export-latex-date-format "%Y-%m-%d")
```

```
(setq org-export-latex-classes nil)
(add-to-list 'org-export-latex-classes
  '("jsarticle"
    "\\documentclass[a4j]{jsarticle}"
    ("\\section{%s}" . "\\section*{%s}")
    ("\\subsection{%s}" . "\\subsection*{%s}")
    ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
    ("\\paragraph{%s}" . "\\paragraph*{%s}")
    ("\\subparagraph{%s}" . "\\subparagraph*{%s}")
))
(setq org-export-latex-packages-alist
  '(("AUTO" "inputenc"  t)
    ("T1"   "fontenc"   t)
    ))
```

4 dashboard

```
(custom-set-faces!
  '(doom-dashboard-banner :foreground "red"  :weight bold)
  '(doom-dashboard-footer :inherit font-lock-constant-face)
  '(doom-dashboard-footer-icon :inherit all-the-icons-red)
  '(doom-dashboard-loaded :inherit font-lock-warning-face)
  '(doom-dashboard-menu-desc :inherit font-lock-string-face)
  '(doom-dashboard-menu-title :inherit font-lock-function-name-face))
```

5 Theme

```
(modus-themes-load-vivendi)
```

6 Langs and Dev

6.1 glsl-mode

```
(use-package! glsl-mode)
(add-to-list 'auto-mode-alist '("\\.gdshader\\'" . glsl-mode))
```

6.2 plantuml

```
(setq org-plantuml-jar-path "~/.emacs.d/lib/plantuml.jar")
```

6.3 LSP

6.3.1 gdscript-mode

```
(setq gdscript-docs-local-path "/Users/yamamotoryuuji/Documents/docs/")
(setq gdscript-godot-executable "/Users/yamamotoryuuji/Desktop/Godot.app/Contents/MacOS

 (defun lsp--gdscript-ignore-errors (original-function &rest args)
  "Ignore the error message resulting from Godot not replying to the ‘JSONRPC’ request
  (if (string-equal major-mode "gdscript-mode")
      (let ((json-data (nth 0 args)))
        (if (and (string= (gethash "jsonrpc" json-data "") "2.0")
                 (not (gethash "id" json-data nil))
                 (not (gethash "method" json-data nil)))
            nil ; (message "Method not found")
          (apply original-function args)))
    (apply original-function args)))
;; Runs the function ‘lsp--gdscript-ignore-errors‘ around ‘lsp--get-message-type‘ to su
(advice-add #’lsp--get-message-type :around #’lsp--gdscript-ignore-errors)
```

6.4 shell

```
(when (memq window-system ’(mac ns x))
  (exec-path-from-shell-initialize))
```

7 LISP

7.1 RACKET

```
(add-hook ’racket-mode-hook
          (lambda ()
            (define-key racket-mode-map (kbd "<f5>") ’racket-run)))
(setq racket-program "/Applications/Racket\sv8.5/bin/racket")
```

7.2 CLISP

1. SLY

   ```
   (use-package sly)
   ```

7.2.1 COCONUT

```
(use-package! coconut-mode)
(add-to-list 'auto-mode-alist '("\\.coco\\'" . coconut-mode))
```

7.2.2 elisp

```
(use-package! request)
```

8 Org

8.1 habit

```
(require 'org-habit)
```

8.2 Directory

```
(when (string-equal system-type "darwin")

(setq org-directory "~/org")

)
(when (string-equal system-type "gnu/linux")
(setq org-directory "~/org")
)
```

8.3 Journal

```
(when (string-equal system-type "darwin")

(setq +org-capture-journal-file "~/org" )

)
(when (string-equal system-type "gnu/linux")
(setq org-journal-dir "~/MEGAsync/journal" )
```

```
)
```

```
(setq org-journal-date-format "%A, %d %B %Y")
(require 'org-journal)
```

8.4 Agenda

```
(setq org-agenda-skip-scheduled-if-done t
      org-agenda-skip-deadline-if-done t
      org-agenda-include-deadlines t
      org-agenda-block-separator #x2501
      org-agenda-compact-blocks t
      org-agenda-start-with-log-mode t)
(with-eval-after-load 'org-journal
(when (string-equal system-type "darwin")

  (setq org-agenda-files '("~/org/todo.org"
                           "~/org/elisptodo.org"
                           )))

)
(when (string-equal system-type "gnu/linux")

  (setq org-agenda-files '("~/org")))

(setq org-agenda-clockreport-parameter-plist
      (quote (:link t :maxlevel 5 :fileskip0 t :compact t :narrow 80)))
(setq org-agenda-deadline-faces
      '((1.0001 . org-warning)              ; due yesterday or before
        (0.0    . org-upcoming-deadline)))  ; due today or later
```

1. agenda styles

   ```
   (defun air-org-skip-subtree-if-habit ()
     "Skip an agenda entry if it has a STYLE property equal to \"habit\"."
     (let ((subtree-end (save-excursion (org-end-of-subtree t))))
       (if (string= (org-entry-get nil "STYLE") "habit")
   ```

```
        subtree-end
      nil)))

(defun air-org-skip-subtree-if-priority (priority)
  "Skip an agenda subtree if it has a priority of PRIORITY.

PRIORITY may be one of the characters ?A, ?B, or ?C."
  (let ((subtree-end (save-excursion (org-end-of-subtree t)))
        (pri-value (* 1000 (- org-lowest-priority priority)))
        (pri-current (org-get-priority (thing-at-point 'line t))))
    (if (= pri-value pri-current)
        subtree-end
      nil)))

(setq org-agenda-custom-commands
      '(("n" ""
         ((tags "PRIORITY=\"A\""
                ((org-agenda-skip-function '(org-agenda-skip-entry-if 'todo 'done)
                 (org-agenda-overriding-header "High-priority unfinished tasks:"))
          (agenda "" ((org-agenda-span 4)))
          (alltodo ""
                   ((org-agenda-skip-function
                     '(or (air-org-skip-subtree-if-priority ?A)
                          (org-agenda-skip-if nil '(scheduled deadline)))))))
        ("w" ""
         ((alltodo ""
                   (org-habit-show-habits t))))))
```

8.5 Pomodoro

```
(use-package org-pomodoro
    :after org-agenda
    :custom
    (org-pomodoro-ask-upon-killing t)
    (org-pomodoro-format "%s")
    (org-pomodoro-short-break-format "%s")
    (org-pomodoro-long-break-format  "%s")
    :custom-face
    (org-pomodoro-mode-line ((t (:foreground "#ff5555"))))
    (org-pomodoro-mode-line-break   ((t (:foreground "#50fa7b"))))
    :hook
    (org-pomodoro-started . (lambda () (notifications-notify
                                               :title "org-pomodoro"
                           :body "Let’s focus for 25 minutes!"
                           :app-icon "~/.emacs.d/img/001-food-and-restaurant.png")))
    (org-pomodoro-finished . (lambda () (notifications-notify
                                               :title "org-pomodoro"
                           :body "Well done! Take a break."
                           :app-icon "~/.emacs.d/img/004-beer.png")))
    :config
    :bind (:map org-agenda-mode-map
                ("p" . org-pomodoro)))
```

8.5.1 Startup settings

```
;;      :custom (org-bullets-bullet-list '())
(setq org-startup-folded t)

(setq
    org-superstar-headline-bullets-list '("" "" "" "" "" "" "" "")
)
```

8.5.2 Babel

1. Template

   ```
   (require 'org-tempo)
   (add-to-list 'org-structure-template-alist '("el" . "src emacs-lisp"))
   (add-to-list 'org-structure-template-alist '("cl" . "src lisp"))
   (add-to-list 'org-structure-template-alist '("aw" . "src awk"))
   (add-to-list 'org-structure-template-alist '("ba" . "src bash"))
   (add-to-list 'org-structure-template-alist '("py" . "src python"))
   (add-to-list 'org-structure-template-alist '("hs" . "src haskell"))
   (add-to-list 'org-structure-template-alist '("pl" . "src plantuml"))
   (add-to-list 'org-structure-template-alist '("js" . "src javascript"))
   ```

2. tangle

   (a) **TODO** Automatically tangle our Emacs.org config file when we save it

```
(defun efs/org-babel-tangle-config ()
  (when (string-equal (file-name-directory (buffer-file-name))
                      (expand-file-name "home/ryu/.doom.d/config.org"))
    ;; Dynamic scoping to the rescue
    (let ((org-confirm-babel-evaluate nil))
      (org-babel-tangle))))

(add-hook 'org-mode-hook (lambda () (add-hook 'after-save-hook #'efs/org-babe


(org-babel-do-load-languages
 'org-babel-load-languages
 '(lisp . t)
 '(awk . t)
 '(fish . t)
 '(python . t)
 '(haskell. t)
 '(C++ . t)
 '(dot . t)
 '(javascript . t)
 '(ditaa . t)
 '(plantuml. t)
 )
```

8.6 org-roam

8.6.1 org-roam-capture-template

```
(after! org-roam
(setq org-roam-capture-templates
      '(("d" "default" plain
         "%?"
         :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n")
         :unnarrowed t)

        ("l" "programming language" plain
         "* Characteristics\n\n- Family: %?\n- Inspired by: \n\n* Reference:\n\n"
```

```
          :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n")
          :unnarrowed t)

         ("b" "book notes" plain
          "\n* Source\n\nAuthor: %^{Author}\nTitle: ${title}\nYear: %^{Year}\n\n* Summar
          :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n")
          :unnarrowed t)
         ("p" "project" plain "* Goals\n\n%?\n\n* Tasks\n\n** TODO Add initial tasks\n\n
          :if-new (file+head "%<%Y%m%d%H%M%S>-${slug}.org" "#+title: ${title}\n#+fileta
          :unnarrowed t)
         )))
```

8.6.2 dir-option

```
(defun inuru ()
  (interactive)
  (let ((select '((me . roam) (share . loggg))))
    (ivy-read "wiki" select
    :require-match t
    :action (lambda (choice)
              (setq org-roam-directory (concat "/Users/yamamotoryuuji/Dropbox/"
                                               (symbol-name (cdr choice)))))))
  (org-roam-db-sync))
```

8.6.3 org-roam-ui

```
(setq org-roam-directory "/Users/yamamotoryuuji/Dropbox/roam")
(use-package org-roam-bibtex
  :after org-roam
  :config
  (require 'org-ref))

(use-package! websocket
    :after org-roam)

(use-package! org-roam-ui
    :after org ;; or :after org
         normally we'd recommend hooking orui after org-roam, but since org-roam does 
```

```
          a hookable mode anymore, you're advised to pick something yourself
          if you don't care about startup time, use
    :hook (after-init . org-roam-ui-mode)
    :config
    (setq org-roam-ui-sync-theme t
          org-roam-ui-follow t
           org-roam-ui-update-on-save t
          org-roam-ui-open-on-start t))
```

8.6.4 org-roam-dialies

```
(setq org-roam-dailies-directory "/Users/yamamotoryuuji/Dropbox/roam/journal")
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; (setq org-roam-dialies-capture-template                                    ;;
;;       '(("d" "default" entry "* %<%I:%H%p>: %?"                            ;;
;;         :if-new (file+head "%S<%Y-%m-%d>.org" "#+title: %<%Y-%m%d>\n?")))) ;;
;;;;;;;;;;f;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(setq org-roam-dailies-capture-templates
      '(("d" "Journal" entry "* %<%H: %M>\n"
         :if-new (file+head+olp "%<%Y-%m-%d>.org"
              "#+title: %<%Y-%m-%d>\n#+filetags: %<:%Y:%B:>\n"
           ("Journal")))
        ("b" "books" entry "* books"
         :if-new (file+head+olp "%<%Y-%m-%d>.org"
              "#+title: %<%Y-%m-%d>\n#+filetags: %<:%Y:%B:>\n"
           ("Journal")))


        ("m" "Most Important Thing" entry "* TODO %? :mit:"
         :if-new (file+head+olp "%<%Y-%m-%d>.org"
        "#+title: %<%Y-%m-%d>\n#+filetags: %<:%Y:%B:>\n"
        ("Most Important Thing(s)")))))
```

8.7 elgantt

```
(use-package! elgantt)

(setq elgantt-user-set-color-priority-counter 0)
(elgantt-create-display-rule draw-scheduled-to-deadline
  :parser ((elgantt-color . ((when-let ((colors (org-entry-get (point) "ELGANTT-COLOR")
                               (s-split " " colors)))))
  :args (elgantt-scheduled elgantt-color elgantt-org-id)
  :body ((when elgantt-scheduled
           (let ((point1 (point))
                 (point2 (save-excursion
                           (elgantt--goto-date elgantt-scheduled)
                           (point)))
                 (color1 (or (car elgantt-color)
                             "black"))
                 (color2 (or (cadr elgantt-color)
                             "red")))
             (when (/= point1 point2)
               (elgantt--draw-gradient
                color1
                color2
                (if (< point1 point2) point1 point2) ;; Since cells are not necessaril
                (if (< point1 point2) point2 point1) ;; chronological order, make sure
                nil
                `(priority ,(setq elgantt-user-set-color-priority-counter
                                  (1- elgantt-user-set-color-priority-counter))
                           ;; Decrease the priority so that earlier entries take
                           ;; precedence over later ones (note: it doesn't matter if th
                           :elgantt-user-overlay ,elgantt-org-id)))))))

(setq elgantt-header-type 'outline
      elgantt-insert-blank-line-between-top-level-header t
      elgantt-startup-folded nil
      elgantt-show-header-depth t
      elgantt-draw-overarching-headers t)
```

8.8 reading

```
(defconst ladicle/org-journal-dir "~/roam/journal/")
```

```
(defconst ladicle/org-journal-file-format (concat ladicle/org-journal-dir "%Y%m%d.org")

(defvar org-code-capture--store-file "")
(defvar org-code-capture--store-header "")

;; This function is used in combination with a coding template of org-capture.
(defun org-code-capture--store-here ()
  "Register current subtree as a capture point."
  (interactive)
  (message "the header is stored")
  (setq org-code-capture--store-file (buffer-file-name))
  (setq org-code-capture--store-header (nth 4 (org-heading-components))))

;; This function is used with a capture-template for (function) type.
;; Look for headline that registered at ‘org-code-capture--store-header‘.
;; If the matching subtree is not found, create a new Capture tree.
(defun org-code-capture--find-store-point ()
  "Find registered capture point and move the cursor to it."
  (let ((filename (if (string= "" org-code-capture--store-file)
                      (format-time-string ladicle/org-journal-file-format)
                    org-code-capture--store-file)))
    (set-buffer (org-capture-target-buffer filename)))
  (goto-char (point-min))
  (unless (derived-mode-p 'org-mode)
    (error
     "Target buffer \"%s\" for org-code-capture--find-store-file should be in Org mode
     (current-buffer))
    (current-buffer))
  (if (re-search-forward org-code-capture--store-header nil t)
      (goto-char (point-at-bol))
    (goto-char (point-max))
    (or (bolp) (insert "\n"))
    (insert "* Capture\n")
    (beginning-of-line 0))
  (org-end-of-subtree))

;; Capture templates for code-reading
(add-to-list 'org-capture-templates
      '("u" "code-link"
         plain
         (function org-code-capture--find-store-point)
         "% {Summary}\n%(with-current-buffer (org-capture-get :original-buffer) (browse
         :immediate-finish t))

(add-to-list 'org-capture-templates
        '("p" "just-code-link"
         plain
         (function org-code-capture--find-store-point)
         "%a"
         :immediate-finish t))

;;keybinding
(map! (:leader
      (:desc "counsel capture" "c p" #'counsel-org-capture
       :desc "counsel capture"
      "y c" #'org-code-capture--store-here)))
```

8.9 babel

```
(add-hook 'org-mode-hook #'org-modern-mode)
```

8.10 publish

```
(setq easy-hugo-basedir "~/chiple.github.io/")
(doom! :lang
       (org +hugo))
(use-package ox-hugo
  :ensure t
  :after ox)
(setq easy-hugo-url "https://chiple.github.io")
(setq easy-hugo-sshdomain "https://chiple.github.io")
(setq easy-hugo-root "/")
(setq easy-hugo-previewtime "300")
(setq easy-hugo-postdir "content/post")
(setq org-hugo-base-dir "~/chiple.github.io/")
```

8.11 org-benrify

Org Snippet

```
(defun list-headings()
  (interactive)
(defun get-existing-heading-in-buffer ()
  (save-excursion
  (goto-char (point-min))
  (let ((head '()))
    (while (re-search-forward "^*" (point-max) t)
      (add-to-list 'head (list (replace-regexp-in-string "\n" "" (thing-at-point 'line
      )
    head)))

(ivy-read "headings" (get-existing-heading-in-buffer)
          :action (lambda (x) (goto-char (cadr x))))
)
(map! :leader
      :desc "heading list of current buffer"
      "l h" #'list-headings)
```

9 PREFERENCE

```
(defun my-pretty-lambda ()
  (setq prettify-symbols-alist '(("lambda" . 955))))
(add-hook 'python-mode-hook 'my-pretty-lambda)
(add-hook 'python-mode-hook 'prettify-symbols-mode)
(add-hook 'org-mode-hook 'my-pretty-lambda)
(add-hook 'org-mode-hook 'prettify-symbols-mode)
(add-hook 'lisp-mode-hook 'my-pretty-lambda)
(add-hook 'lisp-mode-hook 'prettify-symbols-mode)
(add-hook 'emacs-lisp-mode-hook 'my-pretty-lambda)
(add-hook 'emacs-lisp-mode-hook 'prettify-symbols-mode)
```

```
(defun font-set-yay ()
(set-fontset-font t 'japanese-jisx0208 (font-spec :family "")))

(set-fontset-font t 'japanese-jisx0208 (font-spec :family ""))
(add-hook 'emacs-startup-hook 'font-set-yay)
```

10 screen

```
(defun split-screen-1 ()
  (interactive)
  (progn
  (evil-window-split)
  (next-window-any-frame)
  (shrink-window 15)
  (evil-window-vsplit)
  (eshell)
  (next-window-any-frame)
  (org-agenda :key "n")
  (next-window-any-frame)
    ))

(defun split-screen-2 ()
  (interactive)
  (progn
  (evil-window-vsplit)
  (evil-window-split)
  (shrink-window 15)
  (evil-window-vsplit)
  (eshell)
  (next-window-any-frame)
  (org-agenda :key "n")
  (next-window-any-frame)
    ))

(defun split-screen-3 ()
  (interactive)
  (progn
  (evil-window-vsplit)
  (find-file "~/edu/clisp")
  (next-window-any-frame)
  (sly)
  (evil-window-vsplit)
  (org-roam-ref-find "clisp")
  ))

(defun split-screen-4 ()
  (interactive)
  (progn
    (let ((contest-num (read-string "What is the number of contest? :"))
        (dir-name nil))
  (evil-window-vsplit)
  (setq dir-name (concat "~/edu/python/abc" contest-num))
  (mkdir dir-name)
  (find-file (concat dir-name "/a.py"))
  (next-window-any-frame)
  (eshell)
  (next-window-any-frame)


    )))

  (use-package ace-window
   :custom-face
    (aw-leading-char-face ((t (:height 4.0 :foreground "#f1fa8c")))))
```

11 tools

11.1 vocacb

```
(defun append-string-to-file (s filename)
  (with-temp-buffer
    (insert s)
    (insert "\n")
    (write-region (point-min) (point-max) filename t)))

(defun ankki ()
  (interactive)
  (progn
    (let ((word (read-string "Type in the word you don't know: ")))
      (append-string-to-file word "~/Documents/words.txt")
      )
    (async-shell-command "python3 ~/.doom.d/asdf.py")
    )
  )
```

11.2 TODO

-
-

11.2.1 keymap

	key	func name	shape
create	SPC a j k	arrow down right	>
	SPC a j h	arrow down left	<
manuplate	SPC a p v	arrow path vertically	or <>
	SPC a p h	arrow path horizontally	or

```
(defun yajirushi-add ()
  (interactive)
  (let ((length (cl-parse-integer(read-string "put the arrow length here: " "3") :radi
        (result ""))
    (cl-do ((num 1 (1+ num)))
        ((> num length))
      (if (equal num length)
          (setq result (concat result ">"))
        (setq result (concat result ">\n"))))
    (with-current-buffer
        (insert result)
      (number-to-string (line-number-at-pos)))
    ))
;;
(defun yajirushi-new-line ()
  (interactive)
  (cl-case (char-after)
    ((?)
     (forward-line -1)
     (let ((line-content (thing-at-point 'line t)))
       (insert line-content)))
    ((?)

     (forward-line 1)
     (let ((line-content (thing-at-point 'line t)))
       (insert "\n")
       (forward-line -1)
       (insert "")
       ))

    ((?)
     (let ((line-content (thing-at-point 'line t))
           (end (point)))
       (beginning-of-line)
       (let* ((start (point))
              (offset (- end start)))
         (forward-line 1)
         (insert line-content)
         (forward-line -1)
         (cl-do ((num 0 (1+ num)))
             ((> num offset))
           (cl-case (char-after)
             ((?)
              (delete-forward-char 1)
              (insert "")
              (forward-char -1)
              )
             ((?)
              (delete-forward-char 1)
              (insert "")
              (forward-char -1)
              )
             ((?)
              (delete-forward-char 1)
              (insert " ")
              (forward-char -1)
              )
             ((?)
              (delete-forward-char 1)
              (insert " ")
              (forward-char -1)
              )
             )

           (forward-char 1)
```

```
          )
        )))))
;;X
(defun yajirushi-go-upward ()
  (let ((end (point)))
    (beginning-of-line)
    (let* ((start (point))
           (offset (- end start))
           )
      (forward-line -1)
      (goto-char (+ offset (point)))
      )
    ))
;;
(defun yajirushi-go-left ()
  (interactive)
  (while (not (equal (thing-at-point 'char t) ""))
    (forward-char -1)))

(defun yajirushi-go-right ()
  (interactive)
  (while (not (equal (thing-at-point 'char t) ""))
    (forward-char 1)))
;;(yajirushi-go-upward)
;;
;;Hub
(defun detect-box ()
  (interactive)
  (let ((start) (top-left) (bottom-right))
    (setq start (point))
  (cl-case (char-after)
    ((?)
     (yajirushi-go-left)
     (while (not (equal (thing-at-point 'char t) ""))
       (yajirushi-go-upward))
     (setq top-left (point))
     (goto-char start)
     (yajirushi-go-right)
     (setq bottom-right (point))
     ))
  (print top-left)
  (print bottom-right)
  )
)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; (defun adjust-box-shape () ;;
;;   (interactive))           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;
;;
(defun moji-tree ()
  (interactive)
  (let ((word (cl-parse-integer(read-string "put string here: " ))
        (result ""))
        (with-current-buffer
        (insert result)
      (number-to-string (line-number-at-pos)))
    )
                        ))

(defun yajirushi-rotate ()
  (interactive)
  (cl-case (char-after)
    ;;
    ((?)
     (delete-forward-char 1)
     (insert ""))
    ((?)
     (delete-forward-char 1)
     (insert ""))
    ((?)
     (delete-forward-char 1)
     (insert ""))
    ;;
    ((?)
     (delete-forward-char 1)
     (insert "")
     (forward-char -1)
     )
    ((?)
     (delete-forward-char 1)
     (insert ""))
    ))

(defun yajirushi-expand ()
  (interactive)
  (cl-case (char-after)
    ((?)
     (insert ""))))

;;TODO;;;;;;;;;;;;;;;;;;;;;;;;
;; (defun box-display ()     ;;
;;   (interactive)           ;;
;;)                          ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;key-bind
(map! (:leader
      (:desc "" "a j l" #'yajirushi-add
      :desc "" "a r" #'yajirushi-rotate
      :desc "" "a x" #'yajirushi-expand
      :desc "" "a o" #'yajirushi-new-line)))
```

12 ivy

```
(use-package ivy-posframe
      :config
    (ivy-posframe-mode 1))
(setq ivy-posframe-parameters
      '((left-fringe . 10)
        (right-fringe . 10)))
```

13 beacon

```
(use-package beacon
  :custom
     (beacon-color "white")
    :config
    (beacon-mode 1)
    )
```

14 easy-draw

```
(with-eval-after-load 'org
  (require 'edraw-org)
  (edraw-org-setup-default))
```

15 workspace

```
(defun open-this-buffer-in-workspece ()
  (interactive)
  (let ((where-i-was (current-buffer)))
    (+workspace/new)
    (switch-to-buffer where-i-was)))

(map! (:leader
       (:desc "to-workspace" "w z"#'open-this-buffer-in-workspece
        :desc "to-workspace" "w d"#'+workspace/delete)))
```

16 leave node name to journal

```
(defun extract-link-name (link-content)
  (let ((brace link-content))
    (string-match "\\]\\[\\(.*\\)\\]\\]" brace)
    (match-string 1 brace)))

;;(extract-link-name "[[hazure][asdf]]")
(defun get-exsting-link-name ()
  (save-excursion
    (goto-char (point-min))
    (let ((rect-bracketed '()))
      (while (re-search-forward "^\\[" (point-max) t)
        (add-to-list 'rect-bracketed
                     (extract-link-name (thing-at-point 'line t))))
      rect-bracketed)))
```

```
(defun linkp (name)
  (if (member name (get-exsting-link-name))
      t
    nil))

(defun get-today-file ()
  ;;get the file name of current date
  (let ((file-name (org-journal--get-entry-path))
        year month date)
    (string-match "[0-9]+" file-name)
    (setq file-name (match-string 0 file-name))
    (setq year (substring file-name 0 4))
    (setq month (substring file-name 4 6))
    (setq date (substring file-name 6 8))
    (format "%s-%s-%s.org" year month date)))

(unless (file-exists-p (format "%s/%s" org-roam-dailies-directory (get-today-file)))
  (org-roam-dailies-capture-today :KEYS "d") (save-buffer))

(defun visited-nodep (buffer)
  "This argument is just for the org journal of today.
TODO This can be generic.
if nil, just put the tag bottom of the org file
else, just put the link to the * visited node"
  (set-buffer buffer)
  (save-excursion
    (let ((nodes ’()))
      (goto-char (point-min))
      (while (re-search-forward "^*" (point-max) t)
        (add-to-list ’nodes (replace-regexp-in-string "\n" "" (thing-at-point ’line t)
      (if (member "* visited" nodes)
          t
        nil)
      )))


(defun get-node-name (str)
  (string-match "-.*" str)
  (print (substring (match-string 0 str) 1 (length (match-string 0 str))))
  )
```

```
(defun write-to (buffer)
  (with-current-buffer
      (let ((new-node (buffer-name)))
        (set-buffer buffer)
        (goto-char (point-max))
        ;;This can be more short, joining the unless into one statement.
        ;;But this is easy to read and write.
        (unless (visited-nodep buffer)
          (save-excursion
            (goto-char (point-max))
            (insert "* visited")))

        (unless (file-exists-p (format "%s/%s" org-roam-dailies-directory (get-today-fi
          (print "no-today fie"))

        (unless (linkp (get-node-name new-node))
          (save-excursion
            (re-search-forward "* visited" (point-max) t)
            (insert (format "\n[[%s][%s]]\n" (concat org-roam-directory "/" new-node)
        (print (current-buffer)))))

(add-hook 'org-roam-capture-new-node-hook (lambda () (write-to (get-today-file))))
(add-hook 'org-roam-find-file-hook (lambda () (write-to (get-today-file))))
```

17 others

```
(use-package ob)
(use-package dashboard
  :ensure t
  :config
  (dashboard-setup-startup-hook))

(setq dashboard-startup-banner "~/graph/transparent.png")
(add-to-list 'custom-theme-load-path "./.doom.d/themes/yamamotoryuuji-theme.el")
(setq cutom-theme-directory "~/.doom.d/themes")
```

17.1 competitive

(oj-test)

```
(defun pyt-test ()
  (interactive)
(oj--exec-script "oj t -c \"python3 main.py\""))
(defun pyt-submit()
  (interactive)
(oj--exec-script "oj submit main.py"))

(map! (:leader
      (:desc "delete-content-of-double-quote" "o j p s" #'pyt-submit
       :desc "delete-content-of-double-quote" "o j p t" #'pyt-test)))
```

18 cursor move

18.1 “”

```
(defun go-and-delete-in-double-quote ()
  (interactive)
  (re-search-forward "\"" (line-end-position) t)
  (kill-region (mark) (1- (point)))
  )
(map! :leader
      :desc "delete-content-of-double-quote"
      "d l w" #'go-and-delete-in-double-quote)
```

18.2)

```
(defun goto-end-of-parenthesis ()
  (interactive)
  (set-mark (point))
  (re-search-forward ")" (line-end-position) t)
  (kill-region (mark) (1- (point))))
(map! :leader
      :desc "delete-content-of-double-quote"
      "d l )" #'goto-end-of-parenthesis)
```

19 others

```
(add-hook 'org-mode-hook #'org-inline-anim-mode)
(defun inline-img-wrap ()
  (interactive)
  (org-inline-anim-animate 4))

(map! :leader
      :desc "added the prefix"
      "a n" #'inline-img-wrap)
```

20 shell

```
;; load environment value
(load-file (expand-file-name "~/.emacs.d/shellenv.el"))
(dolist (path (reverse (split-string (getenv "PATH") ":")))
  (add-to-list 'exec-path path))

;; set eshell aliases
(setq eshell-command-aliases-list
      (append
       (list
        (list "mknd" "bash ~/tools/gd_sousa/mknd.bash")
        (list "adnd" "bash ~/tools/gd_sousa/add_node_to_scene.bash"))
       eshell-command-aliases-list))
```