Dotemacs Documentation

Release latest

Jan 19, 2020

Contents

Author Shreyas Ragavan

Contents

CHAPTER 1

1 Introduction and preamble

This is my literate, Org-mode based configuration for Emacs, which are essentially customisations built on top of the starter-kit Scimax. View a nicely rendered version with easy navigation on my website, or if you prefer: on github.

```
Scimax - Awesome editing for scientists and engineers. Scimax is an Emacs starterkit␣
↪for scientists and engineers. It provides a comprehensive configuration of Emacs␣
↪for scientific programming and publishing.

`John Kitchin <https://github.com/jkitchin>`_
```

Scimax specific variables have their own heading to make it 'easier' to experiment with other starter-kits.

The style of documentation is particularly influenced by the dotemacs config of Mathieu Marques, which I found very engaging to read.

```
Note: The configuration posted on my website and github repo are updated from time to␣
↪time, and may be older than the version I am using everyday.
```

1.1 1.1 This configuration

Scimax's init calls the `user.el` script placed in the user folder. The following snippet is placed in `user.el` to load this org file and then my encrypted personal configuration. This org file and the tangled emacs-lisp script is also available in a github repo.

```
;; Loading this file that you are viewing, which I name sr-config.org
(org-babel-load-file (expand-file-name "sr-config.org" user-emacs-directory))

;; Loading secret config containing personal information
(org-babel-load-file (expand-file-name "sr-secrets.org.gpg" user-emacs-directory))

(garbage-collect)
```

1.1.1 1.1.0.1 TODO Using this configuration

While using the Org file - you may need to set `:tangle no` in the headers for the code snippets that you do not need, and set the location of directories for org files, org agenda etc.

A bunch these scripts are not tangled and kept for testing or reference purposes. The tangled `config.el` contains the actual configuration that is used.

1.1.0.1.1 Method 1

1. Clone Scimax
2. Add the above snippet to `user.el` in the user directory. Update the file name and paths as required.
3. Place this org file in the user directory.
4. Run the provided script for installing the packages needed for Scimax. Once that is done, `user.el` will call this org file.

1.1.0.1.2 Method 2

Pick up snippets that you like from the `config.el` file, which is tangled from this org file, and only includes the snippets that I actually use.

1.1.2 1.1.0.2 TODO Overall Tasks and Areas of Improvement [0/5]

- Remove packages that are no longer used
- Switch to the use-package approach everywhere.
- Improve the documentation to make it more user friendly.
- Improve instructions to use this configuration
- Figure out how external packages can be installed.

1.1.3 1.1.0.3 Script to create symlinks of configuration in scimax/user directory

```
ln -s ~/scimax-personal/preload.el ~/scimax/user/
ln -s ~/scimax-personal/user.el ~/scimax/user/
ln -s ~/scimax-personal/sr-config.org ~/scimax/user/
ln -s ~/scimax-personal/user.el ~/scimax/user/
rm -rf ~/scimax/user/snippets
ln -s ~/scimax-personal/snippets ~/scimax/user/
ln -s ~/scimax-personal/sr-secrets.org.gpg ~/scimax/user/
ln -s ~/scimax-personal/archive ~/scimax/user/
ln -s ~/scimax-personal/external_packages ~/scimax/user/
ln -s ~/scimax-personal/preload.el ~/scimax/user/
ln -s ~/scimax-personal/mail/.mbsyncrc ~/.mbsyncrc
```

Using sudo for fastmail certification

```
ln -s ~/scimax-personal/mail/fmail.crt /etc/postfix/
```

1.2 1.2 Other literate Emacs configs

These references were used for exploration and inspiration. Other resources and references are included with the code.

1. Karl Voit
2. Mathieu Marques
3. Lee Hinman
4. Sacha Chua
5. Bernt Hansen's very detailed Org-mode config

CHAPTER 2

2 Stable Core

These are packages and functions that I know to be working as expected, and settings that I use on a daily basis.

2.1 2.1 OS Level variables [0/0]

Since I switch between a Linux machine and a Mac frequently, it is better to define variables that can be used to set other variables depending on the OS.

```
;; Get current system's name
(defun insert-system-name()
  (interactive)
  "Get current system's name"
  (insert (format "%s" system-name))
  )

;; Get current system type
(defun insert-system-type()
  (interactive)
  "Get current system type"
  (insert (format "%s" system-type))
  )

;; Check if system is Darwin/Mac OS X
(defun system-type-is-darwin ()
  (interactive)
  "Return true if system is darwin-based (Mac OS X)"
  (string-equal system-type "darwin")
  )

;; Check if system is GNU/Linux
(defun system-type-is-gnu ()
  (interactive)
  "Return true if system is GNU/Linux-based"
```

(continues on next page)

(continued from previous page)

```
  (string-equal system-type "gnu/linux")
  )
(message "Completed OS Level variables load")
```

2.2 2.2 Browse kill ring

```
(use-package browse-kill-ring
  :ensure t
  :defer nil
)
```

2.3 2.3 Remove trailing whitespace at the end of lines

```
(add-hook 'before-save-hook 'delete-trailing-whitespace)
```

2.4 2.4 Remove '^' at the start of ivy commands

```
(setq ivy-initial-inputs-alist nil)
```

2.5 2.5 Move to the next sentence

As mentioned in the reference, by default in Emacs, a double space is set to end a sentence. This removes that. However, there are tools in scimax to move backwards. Reference: http://pragmaticemacs.com/emacs/move-to-startend-of-line-or-sentence/

```
;; sentences end with single space
(setq sentence-end-double-space nil)
```

2.6 2.6 Package installation

Though the use-package approach is a lot more elegant, I also like to have a list of all my installed packages. In any case, this is more in line with my earlier configurations. As things evolve, I will probably shift completely to the use-package method.

```
(setq package-list '(diminish
                     ;; ztree
                     ;; org-gcal
                     w3m
                     ;; org-trello
                     org-web-tools
                     auto-indent-mode
                     ob-sql-mode
```

(continues on next page)

(continued from previous page)

```
                    dash
                    org-super-agenda
                ;; workgroups2
                    switch-window
                    ess
                    ess-R-data-view
                    ;; interleave
                    deft
                    org-bookmark-heading
                    writeroom-mode
                    ;; evil
                    ;; evil-leader
                    polymode
                    poly-R
                    helm-ag
                    writegood-mode
                    artbollocks-mode
                    multiple-cursors
                    ox-reveal
                    better-defaults
                    jedi jedi-core
                    ag ein
                    ;; ein-mumamo
                    ido-vertical-mode
                    company-jedi
                    conda
                    ;; spacemacs-theme
                    ;; elfeed-goodies
                    helpful
                    browse-kill-ring
                    ivy-yasnippet
                    speed-type
                    clojure-mode
                    cider
                    helm-dash
                    org-projectile
                    bash-completion
                    elmacro
                    helm-org-rifle
                    sx define-word))
```

```
;;fetch the list of packages available
(unless package-archive-contents
  (package-refresh-contents))

;; install the missing packages
(dolist (package package-list)
  (unless (package-installed-p package)
    (package-install package)))
```

2.7 2.7 Switch-window configuration

Source link: https://github.com/dimitri/switch-window

```
(use-package switch-window
  :config
  ;;

  (require 'switch-window)

  (global-set-key (kbd "C-x o") 'switch-window)
  (global-set-key (kbd "C-x 1") 'switch-window-then-maximize)
  (global-set-key (kbd "C-x 2") 'switch-window-then-split-below)
  (global-set-key (kbd "C-x 3") 'switch-window-then-split-right)
  (global-set-key (kbd "C-x 0") 'switch-window-then-delete)

  (global-set-key (kbd "C-x 4 d") 'switch-window-then-dired)
  (global-set-key (kbd "C-x 4 f") 'switch-window-then-find-file)
  (global-set-key (kbd "C-x 4 m") 'switch-window-then-compose-mail)
  (global-set-key (kbd "C-x 4 r") 'switch-window-then-find-file-read-only)

  (global-set-key (kbd "C-x 4 C-f") 'switch-window-then-find-file)
  (global-set-key (kbd "C-x 4 C-o") 'switch-window-then-display-buffer)

  (global-set-key (kbd "C-x 4 0") 'switch-window-then-kill-buffer)

  ;; selecting minibuffer
  (setq switch-window-minibuffer-shortcut ?z)
  )
```

2.8 2.8 Create intermediate directories while saving files

Source: https://superuser.com/questions/131538/can-i-create-directories-that-dont-exist-while-creating-a-new-file-in-emacs

```
(defadvice find-file (before make-directory-maybe (filename &optional wildcards)␣
↪activate)
  "Create parent directory if not exists while visiting file."
  (unless (file-exists-p filename)
    (let ((dir (file-name-directory filename)))
      (unless (file-exists-p dir)
        (make-directory dir)))))
```

2.9 2.9 Shortcuts and registers

```
(set-register ?n (cons 'file "~/my_org/notes.org"))
(set-register ?l (cons 'file "~/application_letters/letter.md"))
(set-register ?k (cons 'file "~/application_letters/Cover_letter_Shreyas_R.pdf"))
(set-register ?p (cons 'file "~/org_cv/CV_Shreyas_Ragavan.pdf"))
(set-register ?r (cons 'file "~/org_cv/CV_Shreyas_Ragavan.org"))
(set-register ?t (cons 'file "~/my_org/todo-global.org"))
(set-register ?i (cons 'file "~/dotemacs/.emacs.d/new-init.org"))
(set-register ?j (cons 'file "~/my_org/mrps_canjs.org"))
(set-register ?f (cons 'file "~/scimax/user/sr-cust/"))
(set-register ?d (cons 'file "~/my_org/datascience.org"))
(set-register ?m (cons 'file "~/my_org/"))
```

(continues on next page)

(continued from previous page)

```
(set-register ?b (cons 'file "~/my_org/blog-book.org"))
(set-register ?g (cons 'file "~/my_gits/"))
```

```
(global-set-key (kbd "M-s g") 'google-this-mode-submap)
```

```
(global-set-key (kbd "M-s i") 'ivy-yasnippet)
```

```
(global-set-key (kbd "M-s u") 'mu4e-update-mail-and-index)
(global-set-key (kbd "M-s m") 'mu4e~headers-jump-to-maildir)
(global-set-key (kbd "C-x m") 'mu4e-compose-new)
```

```
(global-set-key (kbd "C-x t") 'org-insert-todo-heading)
(global-set-key (kbd "C-c d") 'org-time-stamp)
(global-set-key (kbd "M-s s") 'org-save-all-org-buffers)
;;(global-set-key (kbd "M-s j") 'org-journal-new-entry)
```

```
(global-set-key (kbd "C-<f9>") 'sr/punch-in)
(global-set-key (kbd "M-<f9>") 'sr/punch-out)
```

```
(if system-name-is-darwin
    (progn
      (setq mac-right-command-modifier 'hyper)
      (setq mac-right-option-modifier 'super)
      )
  )

(if system-name-is-gnu
    (progn
      (setq right-command-)
      )
  )
```

```
(global-set-key (kbd "M-s f") 'frog-jump-buffer)
```

#+END_SRC

```
frog-jump-buffer
```

2.10 2.10 yanking links in org format

Source: sacha chua.

Enables inserting a URL into an org document as '[<URL>][link]' by tapping F6 after copying the URL. This is useful to reduce clutter with long links, and even include links in headings.

```
(defun my/yank-more ()
  (interactive)
  (insert "[[")
  (yank)
  (insert "][link]]"))
(global-set-key (kbd "<f6>") 'my/yank-more)
```

2.11 2.11 Export setup

```
(require 'ox-org)
(require 'ox-word)
(require 'ox-md)
(load "~/scimax/ox-ipynb/ox-ipynb.el")
```

2.12 2.12 Markdown config

Setting pandoc as the markdown command for live previews. The default command is `markdown`, which could be installed as a separate package.

```
(setq markdown-command "pandoc")
```

2.13 2.13 SLIME and lisp

Installing the SLIME package

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

Setting the location of the lisp interpreter based on the OS being used:

```
(if (system-type-is-darwin)
    (setq inferior-lisp-program "/usr/local/bin/clisp")
  )

(if (system-type-is-gnu)
    (setq inferior-lisp-program "/usr/bin/clisp")
    )
```

2.14 2.14 Expand region

- Note taken on [2019-02-07 Thu 09:27] Explore how this works, and customise it.

```
(use-package expand-region
  :ensure t
  :bind ("C-=" . er/expand-region))

(message "Loaded easier selection")
```

2.15 2.15 Hippie Expand

```
(global-set-key (kbd "M-/") (make-hippie-expand-function
                          '(try-expand-dabbrev-visible
                            try-expand-dabbrev
                            try-expand-dabbrev-all-buffers) t))
```

2.16 2.16 Theme and visuals

Since I run emacs as a daemon and call the emacsclient, the background has to be set for new frames. Additionally, I'd like the frames to launch full screen.

```
(setq default-frame-alist
      '(;; (background-color . "whitesmoke")
        ;; (foreground-color . "black")
        (fullscreen . maximized)
        ))
```

```
(setq custom-safe-themes t)
(set-background-color "whitesmoke")
```

This seems to prevent the emacs daemon from starting up. I will need a condition wherein the daemon is not affected, but the theme is applied in the case of a terminal environment.

```
(use-package gruvbox-theme
:ensure nil
:defer t
)

(if (display-graphic-p)
    (message "GUI")
  (load-theme 'gruvbox-dark-hard)
  (set-background-color "whitesmoke"))
```

The same font is named differently in Antergos (Linux) and in the Mac OS.

```
;; For Linux
(if (system-type-is-gnu)
    (set-face-attribute 'default nil :family "ttf-iosevka" :height 130 ))

;; For Mac OS
(if (system-type-is-darwin)
    (set-face-attribute 'default nil :family "Iosevka Type" :height 160 ))
```

Source: http://pragmaticemacs.com/emacs/get-that-spacemacs-look-without-spacemacs/

```
(use-package spaceline
  :demand t
  :init
  (setq powerline-default-separator 'arrow-fade)
  :config
  (disable-theme 'smart-mode-line-light)
  (require 'spaceline-config)
  (spaceline-emacs-theme)
  (spaceline-toggle-buffer-position-off)
)
```

- Note taken on [2019-02-07 Thu 08:20] These settings have to be cleaned up and the code optimised.

```
(setq org-hide-leading-stars t)
;;(setq org-alphabetical-lists t)
(setq org-src-fontify-natively t)  ;; you want this to activate coloring in blocks
(setq org-src-tab-acts-natively t) ;; you want this to have completion in blocks
(setq org-hide-emphasis-markers t) ;; to hide the *,=, or / markers
(setq org-pretty-entities t)       ;; to have \alpha, \to and others display as utf8␣
↪http://orgmode.org/manual/Special-symbols.html

;; Highlighting lines in the agenda, where the cursor is placed.
(add-hook 'org-agenda-mode-hook (lambda () (hl-line-mode 1)))

;; Setting up clean indenting below respective headlines at startup. - from the org␣
↪mode website
(setq org-startup-indented t)

;; ;; use org bullets from emacsist
;; (use-package org-bullets
;;   :ensure t
;;   :init
;;   :config
;;   (add-hook 'org-mode-hook (lambda () (org-bullets-mode 1))))
```

source: Sacha Chua

```
(setq org-fontify-done-headline t)
(custom-set-faces
 '(org-done ((t (:foreground "DarkGreen"
                         :weight normal
                         :strike-through t))))
 '(org-headline-done
   ((((class color) (min-colors 16) (background dark))
     (:foreground "LightSalmon" :strike-through t)))))
```

Source : SO link ,

```
(set-face-attribute 'org-todo nil
                    :box '(:line-width 2
                           :color "black"
                           :style released-button)
                    :inverse-video t
                    )
(set-face-attribute 'org-done nil
                    :box '(:line-width 2
                           :color "black"
                           :style released-button)
                    :inverse-video t
                    )
(set-face-attribute 'org-priority nil
                    :inherit font-lock-keyword-face
                    :inverse-video t
                    :box '(:line-width 2
                           :color "black"
                           :style released-button)
                    )
```

2.17 2.17 TODO mu4e

- Note taken on [2019-02-12 Tue 14:53] The use-package documentation specifies a method to do this via use-package itself, without enclosing the whole snippet within a if clause.
- Note taken on [2019-02-07 Thu 20:43] The mu4e config has to be broken down and the send email with htmlize has to be evaluated.
- Note taken on [2019-02-07 Thu 09:04] As of now, I do not acess my email on different computers via Emacs. The end goal is to setup a mail server via VPS and store my email online, which can then be searched via Emacs and mu4e from any location.

```
(if (system-type-is-darwin)
    (progn
      (add-to-list 'load-path "/usr/local/share/emacs/site-lisp/mu4e")
      (require 'mu4e)
      (require 'mu4e-contrib)
      (require 'org-mu4e)

      (setq
       mue4e-headers-skip-duplicates  t
       mu4e-view-show-images t
       mu4e-view-show-addresses 't
       mu4e-compose-format-flowed t
       ;;mu4e-update-interval 200
       message-ignored-cited-headers 'nil
       mu4e-date-format "%y/%m/%d"
       mu4e-headers-date-format "%Y/%m/%d"
       mu4e-change-filenames-when-moving t
       mu4e-attachments-dir "~/Downloads/Mail-Attachments/"
       mu4e-maildir (expand-file-name "~/my_mail/fmail")
       message-citation-line-format "On %Y-%m-%d at %R %Z, %f wrote..."
       mu4e-index-lazy-check t
       ;; After Years. I've finally found you.
       mu4e-compose-dont-reply-to-self t
       mu4e-headers-auto-update nil
       )

      ;; mu4e email refiling loations
      (setq
       mu4e-refile-folder "/Archive"
       mu4e-trash-folder  "/Trash"
       mu4e-sent-folder   "/Sent"
       mu4e-drafts-folder "/Drafts"
       )

      ;; setup some handy shortcuts
      (setq mu4e-maildir-shortcuts
            '(("/INBOX"   . ?i)
          ("/Sent"    . ?s)
          ("/Archive" . ?a)
          ("/Trash"   . ?t)))

      ;;store link to message if in header view, not to header query
      (setq org-mu4e-link-query-in-headers-mode nil
            org-mu4e-convert-to-html t) ;; org -> html
```

(continues on next page)

(continued from previous page)

```
      (autoload 'mu4e "mu4e" "mu for Emacs." t)

      ;; Config for sending email
      (setq
       message-send-mail-function 'message-send-mail-with-sendmail
       send-mail-function 'sendmail-send-it
       message-kill-buffer-on-exit t
       )

      ;; allow for updating mail using 'U' in the main view:
      (setq mu4e-get-mail-command  "mbsync -q fins")

      ;; Stolen from https://github.com/djcb/mu/issues/1431 and found thanks to␣
↪parsnip in #emacs
      (defun my-mu4e-mbsync-current-maildir (msg)
    (interactive)
    (let* ((maildir (downcase (substring (plist-get msg :maildir) 1)))
           (mu4e-get-mail-command (format "mbsync %s" maildir)))
      (mu4e-update-mail-and-index t)))

      ;; Enabling view in browser for HTML heavy emails that don't render well
      (add-to-list 'mu4e-view-actions
                '("ViewInBrowser" . mu4e-action-view-in-browser) t)
      (add-to-list 'mu4e-view-actions
                  '("mbsync maildir of mail at point" . my-mu4e-mbsync-current-
↪maildir) t)

      ;; Don't keep asking for confirmation for every action
      (defun my-mu4e-mark-execute-all-no-confirm ()
    "Execute all marks without confirmation."
    (interactive)
    (mu4e-mark-execute-all 'no-confirm))
      ;; mapping x to above function
      (define-key mu4e-headers-mode-map "x" #'my-mu4e-mark-execute-all-no-confirm)

      ;; source: http://matt.hackinghistory.ca/2016/11/18/sending-html-mail-with-mu4e/

      ;; this is stolen from John but it didn't work for me until I
      ;; made those changes to mu4e-compose.el
      (defun htmlize-and-send ()
    "When in an org-mu4e-compose-org-mode message, htmlize and send it."
    (interactive)
    (when
        (member 'org~mu4e-mime-switch-headers-or-body post-command-hook)
      (org-mime-htmlize)
      (org-mu4e-compose-org-mode)
      (mu4e-compose-mode)
      (message-send-and-exit)))

      ;; This overloads the amazing C-c C-c commands in org-mode with one more␣
↪function
      ;; namely the htmlize-and-send, above.
      (add-hook 'org-ctrl-c-ctrl-c-hook 'htmlize-and-send t)))
```

```
(use-package mu4e
```

(continues on next page)

(continued from previous page)

```
:ensure nil
:hook
 ((mu4e-view-mode . visual-line-mode)
   (mu4e-compose-mode . (lambda ()
                          (visual-line-mode)
                          (use-hard-newlines -1)
                          (flyspell-mode)))
   (mu4e-view-mode . (lambda() ;; try to emulate some of the eww key-bindings
                       (local-set-key (kbd "<tab>") 'shr-next-link)
                       (local-set-key (kbd "<backtab>") 'shr-previous-link)))
   (mu4e-headers-mode . (lambda ()
                          (interactive)
                          (setq mu4e-headers-fields
                                `((:human-date . 25) ;; alternatively, use :date
                                  (:flags . 6)
                                  (:from . 22)
                                  (:thread-subject . ,(- (window-body-width) 70)) ;;␣
↪alternatively, use :subject
                                  (:size . 7))))))
:custom
(mu4e-update-interval 150)
(message-kill-buffer-on-exit t)

)
```

```
(use-package mu4e-alert
    :defer t
    :config
    (when (executable-find "notify-send")
      (mu4e-alert-set-default-style 'libnotify))
    :hook
    ((after-init . mu4e-alert-enable-notifications)
     (after-init . mu4e-alert-enable-mode-line-display)))
```

2.18 2.18 Multiple Cursors

```
(use-package multiple-cursors
  :ensure t
  :config
  (global-set-key (kbd "C-S-c C-S-c") 'mc/edit-lines)
  (global-set-key (kbd "C->") 'mc/mark-next-like-this)
  (global-set-key (kbd "C-<") 'mc/mark-previous-like-this)
  (global-set-key (kbd "C-c C-<") 'mc/mark-all-like-this)
  )

(message "Loaded MC")
```

2.19 2.19 git related

- Note taken on [2019-02-07 Thu 09:30] Started using this today. It is actually very convenient to quickly view the changes made in the document. There is a function to pop up the changes at that location. I need to learn more about using this tool effectively.

```
(use-package git-gutter
  :ensure t
  :config
  (global-git-gutter-mode 't)
  :diminish git-gutter-mode)
```

```
(setq magit-revert-buffers 'silent)
```

```
(message "Loaded git related config")
```

2.20 2.20 Projectile behavior

```
(setq projectile-sort-order 'recently-active)

;; Change cache file location
(setq projectile-cache-file "~/my_org/emacs_meta/.projectile-cache")
```

2.21 2.21 Helm

- Note taken on [2019-07-05 Fri 11:55] Adding `helm-for-files` as this is not being autoloaded for enabling the hotspot feature in Scimax.
- Note taken on [2019-03-06 Wed 17:26] I tried using Ivy for a period. However, Helm's interface is simply a lot more pleasing and there are actually several additional actions that can be performed via helm itself.
- Note taken on [2019-03-04 Mon 15:48] Though I preferred Helm initially for several commands - I realised that scimax has several useful customisations for the ivy and counsel packages. Overall ivy is also lighter than helm and therefore these customisations are being discarded for now.

I prefer using Helm for specific functions like M-x, find files and bookmarks and switching buffers.

```
(global-set-key (kbd "M-x") 'helm-M-x)
;; Enable fuzzy match for helm-M-x
(setq helm-M-x-fuzzy-match t)

(global-set-key (kbd "C-x C-f") #'helm-find-files)
(global-set-key (kbd "C-x b") #'helm-mini)

(require 'helm-config)
(require 'helm-for-files)
(helm-mode 1)
```

The default save location in the .emacs folder is not very convenient. I would rather store this with my org files since I commit them Everyday.

```
(setq bookmark-default-file "~/my_org/emacs_meta/bookmarks")
```

The default bookmarks list `C-x r l` can be accessed using `helm-bookmarks`. The location of the file would be a nice addition. Technically, `helm-filtered-bookmarks` has almost the same functionality as the list in terms of being able to fuzzy-match a bookmark.

```
(global-set-key (kbd "C-x r b") #'helm-filtered-bookmarks)
(global-set-key (kbd "C-x r l") #'helm-bookmarks)
(setq helm-bookmark-show-location t)
```

- Note taken on [2019-07-04 Thu 08:08] Interim issue with bookmarks file becoming corrupted due to a git conflict. The sources work as expected, with helm mini as well as hotspots.
- Note taken on [2019-04-29 Mon 07:43] After a package update, setting the sources explicitly is causing issues with helm-mini and with scimax hotspots.
- Note taken on [2019-03-04 Mon 15:49] The scimax hotspots can be customised with an improved function that only requires commands locations to be separately defined. This resolved the helm-recentf problem.
- Note taken on [2019-02-12 Tue 14:55] This is still causing issues: the recentf list has to be cleared via helm-mini first.
- Note taken on [2019-02-07 Thu 16:28] This was needed as it seems helm was not sourcing from recentf file lists. With this source list defined, it provides options to choose from recent files, bookmarks, open buffers.

As an example: setting these sources enables my bookmarks to be available along with my buffers, enabling a jump to either.

```
(setq helm-mini-default-sources '(helm-source-buffers-list
                                  helm-source-recentf
                                  helm-source-bookmarks
                                  helm-source-bookmark-set
                                  helm-source-buffer-not-found))

(setq helm-buffers-list-default-sources '(helm-source-buffers-list
                                          helm-source-recentf
                                          helm-source-bookmarks
                                          helm-source-bookmark-set
                                          helm-source-buffer-not-found))
```

This needs *a0217652-e01b-4ba0-82e6-7ef2780381f8* enabled, and is a really cool function that enables jumping around variables and functions in a script file with fuzzy matching !

```
(setq helm-semantic-fuzzy-match t
      helm-imenu-fuzzy-match t)
```

- Note taken on [2019-02-07 Thu 07:46] Need to find exactly what this does

```
(custom-set-variables
 '(helm-follow-mode-persistent t))
```

```
(use-package helm-ag
  :ensure t
  :defer nil
  :config
  (require 'helm-ag)
)

(use-package helm-org-rifle
  :ensure t
  :defer nil
  :config
  (require 'helm-org-rifle)
```

(continues on next page)

(continued from previous page)

```
  (global-set-key (kbd "C-c C-w") #'helm-org-rifle--refile)
)
```

- Note taken on [2019-02-07 Thu 16:53] This is an awesome find. Helm swoop changes the search pattern depending on the location of the cursor. Therefore, while placed on an org headline, calling helm-swoop will preset the search pattern to have headings. The same is true for source code blocks! Fantastic.

Source: https://writequit.org/org/#orgheadline92

```
(use-package helm-swoop
  :ensure t
  :bind (("M-i" . helm-swoop)
         ("M-I" . helm-swoop-back-to-last-point)
         ("C-c M-i" . helm-multi-swoop))
  :config
  ;; When doing isearch, hand the word over to helm-swoop
  (define-key isearch-mode-map (kbd "M-i") 'helm-swoop-from-isearch)
  ;; From helm-swoop to helm-multi-swoop-all
  (define-key helm-swoop-map (kbd "M-i") 'helm-multi-swoop-all-from-helm-swoop)
  ;; Save buffer when helm-multi-swoop-edit complete
  (setq helm-multi-swoop-edit-save t
        ;; If this value is t, split window inside the current window
        helm-swoop-split-with-multiple-windows t
        ;; Split direcion. 'split-window-vertically or 'split-window-horizontally
        helm-swoop-split-direction 'split-window-vertically
        ;; If nil, you can slightly boost invoke speed in exchange for text color
        helm-swoop-speed-or-color nil))
```

```
(message "Loaded Helm customisations")
```

2.22 2.22 Org mode related

```
(setq org-complete-tags-always-offer-all-agenda-tags t)
```

```
(setq
 org-directory "~/my_org/"
 org-agenda-files '("~/my_org/")
 )
```

I've been inserting org notes into the body of the text, since I do not make extensive use of the log book in the agenda and prefer active time stamped notes and the org-journal and org-projectile to take down 'linked' log notes. However, I would like the notes to be inserted after any properties drawers.

```
(setq org-log-state-notes-insert-after-drawers t)
(setq org-log-redeadline 'time)
```

- Note taken on [2019-02-07 Thu 08:55] Need to actually get org-capture via external browser protocol working. Not sure if I need to require org-capture in scimax.

Source: http://www.diegoberrocal.com/blog/2015/08/19/org-protocol/

```
(require 'org-capture)
(require 'org-protocol)
```

- Note taken on [2019-02-07 Thu 08:31] check whether the add-to-list function is sufficient.

```
(add-hook 'find-file-hooks
          (lambda ()
            (let ((file (buffer-file-name)))
              (when (and file (equal (file-name-directory file) "~/my_org/archive/"))
                (org-mode)))))

(add-to-list 'auto-mode-alist '("\\.org_archive\\'" . org-mode))
```

Archive organised by Top level headings in the original file and with Tag preservation

```
(defun my-org-inherited-no-file-tags ()
  (let ((tags (org-entry-get nil "ALLTAGS" 'selective))
        (ltags (org-entry-get nil "TAGS")))
    (mapc (lambda (tag)
            (setq tags
                  (replace-regexp-in-string (concat tag ":") "" tags)))
          (append org-file-tags (when ltags (split-string ltags ":" t))))
    (if (string= ":" tags) nil tags)))

(defadvice org-archive-subtree (around my-org-archive-subtree-low-level activate)
  (let ((tags (my-org-inherited-no-file-tags))
        (org-archive-location
         (if (save-excursion (org-back-to-heading)
                             (> (org-outline-level) 1))
             (concat (car (split-string org-archive-location "::"))
                     "::* "
                     (car (org-get-outline-path)))
           org-archive-location)))
    ad-do-it
    (with-current-buffer (find-file-noselect (org-extract-archive-file))
      (save-excursion
        (while (org-up-heading-safe))
        (org-set-tags-to tags)))))
```

2.22.1 2.22.7.1 Installation

```
(let ((url "https://raw.githubusercontent.com/caiorss/org-wiki/master/org-wiki.el"))
      (with-current-buffer (url-retrieve-synchronously url)
    (goto-char (point-min))
    (re-search-forward "^$")
    (delete-region (point) (point-min))
    (kill-whole-line)
    (package-install-from-buffer)))
```

2.22.2 2.22.7.2 Setup

```
(require 'org-wiki)
(setq org-wiki-location "~/my_projects/ds-job-search")
```

Using the org-id for reference to headings ensures that even if the heading changes, the links will still work.

In addition, I would like an org id to be created every time the capture is used. This facilitates using packages like org-brain which rely extensively on org-id's.

```
(require 'org-id)
(setq org-id-link-to-org-use-id t)
(org-link-set-parameters "id" :store #'org-id-store-link)
(org-link-set-parameters "nb" :store nil)
;; Update ID file .org-id-locations on startup
;; This adds too much time to startup
;; (org-id-update-id-locations)

(setq org-id-method (quote uuidgen))
(add-hook 'org-capture-prepare-finalize-hook 'org-id-get-create)
```

- Note taken on [2019-02-12 Tue 12:19] This requires a complete reload of org to come in effect.

```
(setq org-todo-keywords
      '((sequence "TODO(t)" "NEXT(n)" "CANCEL(c)" "POSTPONED(p)" "|" "DONE(d)"
↪"STABLE(s)")
        (sequence "TEST(T)" "BUG(b)" "KNOWNCAUSE(k)" "|" "FIXED(f)")
        (sequence "|" )))
```

- Note taken on [2019-07-06 Sat 13:56] Helm org rifle is mapped to the refile command. See Helm section.

2.22.3 2.22.10.1 Refile target level for search

```
(setq org-refile-targets
      '((nil :maxlevel . 4)
        (org-agenda-files :maxlevel . 4)))
```

2.22.4 2.22.10.2 TODO General refile settings

- Note taken on [2019-02-07 Thu 08:33] Needs further review and optimisation

```
(setq org-refile-use-outline-path 'file)
(setq org-outline-path-complete-in-steps nil)
(setq org-reverse-note-order t)
(setq org-refile-allow-creating-parent-nodes 'confirm)
```

2.22.5 2.22.10.3 a7ceeb6d-2085-4380-909f-78f5ee698ad7

2.22.6 2.22.11.1 Weekday starts on Monday

```
(setq org-agenda-start-on-weekday 1)
```

2.22.7 2.22.11.2 Display heading tags farther to the right

```
(setq org-agenda-tags-column -150)
```

2.22.8 2.22.11.3 TODO Agenda customisation

- Note taken on [2019-02-07 Thu 08:26] Need to clear up the search functions, enabling complete search in journal files. Archive and some external directories are included, since they are explictly in org mode.

```
(setq org-agenda-custom-commands
      '(("c" "Simple agenda view"
         ((tags "recurr"
            ((org-agenda-overriding-header "Recurring Tasks")))
          (agenda "")
          (todo "")))
        ("o" agenda "Office mode" ((org-agenda-tag-filter-preset '("-course" "-habit"
↪"-someday" "-book" "-emacs"))))
        ("qc" tags "+commandment")
    ("e" tags "+org")
    ("w" agenda "Today" ((org-agenda-tag-filter-preset '("+work"))))
    ("W" todo-tree "WAITING")
    ("q" . "Custom queries") ;; gives label to "q"
    ("d" . "ds related")     ;; gives label to "d"
    ("ds" agenda "Datascience" ((org-agenda-tag-filter-preset '("+datascience"))))
    ("qw" agenda "MRPS" ((org-agenda-tag-filter-preset '("+canjs"))))
    ("qa" "Archive tags search" org-tags-view ""
         ((org-agenda-files (file-expand-wildcards "~/my_org/*.org*"))))
        ("j" "Journal Search" search ""
         ''((org-agenda-text-search-extra-files (file-expand-wildcards "~/my_org/
↪journal/"))))
        ("S" search ""
     ((org-agenda-files '("~/my_org/"))
      (org-agenda-text-search-extra-files )))
    )
      )
```

2.22.9 2.22.11.4 Include gpg files in agenda generation

Source: https://emacs.stackexchange.com/questions/36542/include-org-gpg-files-in-org-agenda

```
;; (unless (string-match-p "\\.gpg" org-agenda-file-regexp)
;;   (setq org-agenda-file-regexp
;;         (replace-regexp-in-string "\\\\.org" "\\\\.org\\\\(\\\\.gpg\\\\)?"
;;                                   org-agenda-file-regexp)))

(setq org-agenda-file-regexp "\\`\\\([^.].*\\.org\\\|[0-9]\\\{8\\\}\\\(\\.gpg\\\)?
↪\\\)\\'")
```

2.22.10 2.22.11.5 Expanding search locations

I initially included my journal location to the agenda search. However it is very slow compared to using grep/rgrep/ag. Therefore, the agenda full text search is now limited to the project directory and the org-brain directory. The snippet below enables searching recursively within folders.

```
(setq org-agenda-text-search-extra-files '(agenda-archives))

(setq org-agenda-text-search-extra-files (apply 'append
                                                (mapcar
```

(continues on next page)

(continued from previous page)

```
                                                (lambda (directory)
                                                  (directory-files-recursively
                                                   directory org-agenda-file-regexp))
                                                '("~/my_projects/" "~/my_org/brain/"))))
```

2.22.11 2.22.11.6 TODO Adding org archive for text search. Optimise this

CREATED <2019-02-07 Thu 08:29>

```
(setq org-agenda-text-search-extra-files '(agenda-archives))
```

2.22.12 2.22.11.7 Enable default fuzzy search like in google

```
(setq org-agenda-search-view-always-boolean t)
```

2.22.13 2.22.11.8 Enable sticky agenda

Experimenting with this setting.

```
(setq org-agenda-sticky t)
```

2.22.14 2.22.11.9 DONE org-habit

- Note taken on [2019-02-12 Tue 13:20] Adding a require has brought org-habit back on track.
- Note taken on [2019-02-07 Thu 09:50] Appears the use-package config for org-habit is not correct and there is some issue in downloading it as a package.

I want to shift the org habit graph in the agenda further out right so as to leave enough room for the headings to be visible.

```
(require 'org-habit)
(setq org-habit-graph-column 90)
```

- Note taken on [2019-02-07 Thu 08:24] need to clean this up.

2.22.15 2.22.12.1 Removing timestamp from datetree captures

```
(setq org-datetree-add-timestamp nil)
```

2.22.16 2.22.12.2 Capture templates

```
(setq org-capture-templates
      '(("t" "Task entry")
        ("tt" "Todo - Fast" entry (file+headline "~/my_org/todo-global.org" "@Inbox")
     "** TODO %?")
```

(continues on next page)

(continued from previous page)

```
        ("tj" "Todo -Job journal" entry (file+olp+datetree "~/my_org/ds-jobs.org"
↪"Job Search Journal")
     "** TODO %?")
        ("te" "Todo - Emacs" entry (file+headline "~/my_org/todo-global.org" "@Emacs␣
↪notes and tasks")
         "** TODO %?")
        ("td" "Datascience inbox" entry (file+headline "~/my_org/datascience.org"
↪"@Datascience @Inbox")
         "** TODO %?")
    ("tm" "Mail Link Todo" entry (file+headline "~/my_org/todo-global.org" "@Inbox")
     "** TODO Mail: %a ")
        ("l" "Link/Snippet" entry (file+headline "~/my_org/link_database.org" ".UL␣
↪Unfiled Links")
         "** %? %a ")
        ("e" "Protocol info" entry ;; 'w' for 'org-protocol'
         (file+headline "~/my_org/link_database.org" ".UL Unfiled Links")
         "*** %a, \n %:initial")
        ("n" "Notes")
        ("ne" "Emacs note" entry (file+headline "~/my_org/todo-global.org" "@Emacs␣
↪notes and tasks")
         "** %?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d %a %H:%M>]\n:END:")
        ("nn" "General note" entry (file+headline "~/my_org/notes.org" "@NOTES")
         "** %?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d %a %H:%M>]\n:END:")
        ("nd" "Datascience note" entry (file+headline "~/my_org/datascience.org"
↪"@Datascience @Notes")
         "** %?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d %a %H:%M>]\n:END:")
        ("g" "BGR stuff")
        ("gi" "Inventory project")
        ("gil" "Daily log" entry (file+olp+datetree "~/my_org/bgr.org" "Inventory␣
↪management Project") "** %? %i")
        ("C" "Commandment" entry (file+datetree "~/my_org/lifebook.org" "")
         "** %? %i :commandment:")
        ("J" "Job search" entry (file+headline "~/my_org/mrps_canjs.org" "MRPS #CANJS
↪")
         "** TODO %? %i ")
        ("w" "Website" plain
         (function org-website-clipper)
         "* %a %T\n" :immediate-finish t)
        ("j" "Journal entry" entry (function org-journal-find-location)
         "* %(format-time-string org-journal-time-format) %?")
        ("i" "Whole article capture" entry
         (file+headline "~/my_org/full_article_archive.org" "" :empty-lines 1)
         "** %a, %T\n %:initial" :empty-lines 1)
        ("c" "Clocking capture")
        ("ct" "Clock TODO" entry (clock) "** TODO %?")
        ("cn" "Clock Note" entry (clock) "** %?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d
↪%a %H:%M>]\n:END:")
        ("r" "Review note" entry (file+weektree "~/my_org/lifebook.org" "#Personal
↪#Reviews")
         "** %?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d %a %H:%M>]\n:END:")
         ))
```

2.22.17 2.22.12.3 TEST Closing org-capture frame on abort

- Note taken on [2019-03-13 Wed 07:35] This basically ensures a clean exit in case of aborting a capture.

- Note taken on [2019-02-07 Thu 08:53] Needs further review.

Source: http://stackoverflow.com/questions/23517372/hook-or-advice-when-aborting-org-capture-before-template-selection

```
(defadvice org-capture
    (after make-full-window-frame activate)
  "Advise capture to be the only window when used as a popup"
  (if (equal "emacs-capture" (frame-parameter nil 'name))
      (delete-other-windows)))

(defadvice org-capture-finalize
    (after delete-capture-frame activate)
  "Advise capture-finalize to close the frame"
  (if (equal "emacs-capture" (frame-parameter nil 'name))))
```

2.22.18 2.22.12.4 TODO Controlling org-capture buffers

- Note taken on [2019-03-13 Wed 08:01] This interferes with org-journal's capture format.

I dislike the way org-capture disrupts my current window, and shows me the capture buffer, and the target buffer as well. I would prefer a small pop up window, and then a revert back to the existing windows once the capture is completed or aborted. However this does not seem possible without modifying Org-mode's source code. This is a workaround described at https://stackoverflow.com/questions/54192239/open-org-capture-buffer-in-specific-Window ,which partially resolves the issue by enabling just a single capture buffer.

```
(defun my-org-capture-place-template-dont-delete-windows (oldfun args)
  (cl-letf (((symbol-function 'delete-other-windows) 'ignore))
    (apply oldfun args)))

(with-eval-after-load "org-capture"
  (advice-add 'org-capture-place-template :around 'my-org-capture-place-template-dont-
↪delete-windows))
```

- Note taken on [2019-02-07 Thu 08:15] Need to check out how this works and whether this is still necessary, since I am using Git.

```
(setq delete-old-versions -1)
(setq version-control t)
```

```
Org-noter's purpose is to let you create notes that are kept in sync when you scroll␣
↪through the document, but that are external to it - the notes themselves live in an␣
↪Org-mode file. As such, this leverages the power of Org-mode (the notes may have␣
↪outlines, latex fragments, babel, etc...) while acting like notes that are made␣
↪inside the document. Also, taking notes is very simple: just press i and annotate␣
↪away!

`Gonçalo Santos <https://github.com/weirdNox>`_
```

```
(use-package org-noter
  :ensure t
  :defer t
  :config
  (setq org-noter-set-auto-save-last-location t)
  )
```

- Note taken on [2019-02-07 Thu 08:42] need to optimise further and convert to use-package style. Also need a way to capture Notes from projects, in addition to tasks.

Starting off with the basic configuration posted in org-projectile github repo.

```
(use-package org-projectile
  :ensure t
  :bind (("C-c n p" . org-projectile-project-todo-completing-read)
         ("C-c c" . org-capture))
  :config
  (setq org-projectile-projects-file
        "~/my_org/project-tasks.org")
  ;; (setq org-agenda-files (append org-agenda-files (org-projectile-todo-files))) ;;␣
↪Not necessary as my task projects are a part of the main org folder
  (push (org-projectile-project-todo-entry) org-capture-templates)
  )
```

2.22.19 2.22.17.1 TEST Optimise CREATED and PLANNED property tags

- Note taken on [2019-02-07 Thu 09:10] Needs further review and optimisation.

Using an active date tag on the heading itself makes the org document look ugly, and makes navigation difficult. This is better entered into a property drawer. Two properties should work well - CREATED (inactive date-time tag) and PLANNED (active date-time tag). This will enable me to filter based on property in the future and easily archive older or irrelevant tasks. When the task is shifted or postponed, only the PLANNED property is changed, leaving clear reference of the created date.

The above is implemented only for tasks with a TODO heading. For now, I want to test using Notes with an inactive date-time tag, which can be individually setup via the capture templates. The attempt is to have a clear separation between tasks and notes.

This is a modified version of the snippet at https://emacs.stackexchange.com/questions/35751/how-to-add-a-created-field-to-any-todo-task

```
(defun sr/log-todo-creation-date (&rest ignore)
  "Log TODO creation time in the property drawer under the key 'CREATED'."
  (when (and (org-get-todo-state)
             (not (org-entry-get nil "CREATED")))
    (org-entry-put nil "CREATED" (format-time-string "[%Y-%m-%d %a]"))
    (org-entry-put nil "PLANNED" (format-time-string (cdr org-time-stamp-formats)))
    ))

(advice-add 'org-insert-todo-heading :after #'sr/log-todo-creation-date)
(advice-add 'org-insert-todo-heading-respect-content :after #'sr/log-todo-creation-
↪date)
(advice-add 'org-insert-todo-subheading :after #'sr/log-todo-creation-date)
(advice-add 'org-capture :after #'sr/log-todo-creation-date)
(advice-add 'org-projectile-project-todo-completing-read :after #'sr/log-todo-
↪creation-date)

;; (require 'org-expiry)
;; ;; Configure it a bit to my liking
;; (setq
;;  org-expiry-created-property-name "CREATED" ; Name of property when an item is␣
↪created
;;  org-expiry-inactive-timestamps   nil         ; Don't have everything in the␣
↪agenda view
;;  )

;; (defun mrb/insert-created-timestamp()
```

(continues on next page)

(continued from previous page)

```
;;    "Insert a CREATED property using org-expiry.el for TODO entries"
;;    (org-expiry-insert-created)
;;    (org-back-to-heading)
;;    (org-end-of-line)
;;    (insert " ")
;;    )

;; ;; Whenever a TODO entry is created, I want a timestamp
;; ;; Advice org-insert-todo-heading to insert a created timestamp using org-expiry
;; (defadvice org-insert-todo-heading (after mrb/created-timestamp-advice activate)
;;    "Insert a CREATED property using org-expiry.el for TODO entries"
;;    (mrb/insert-created-timestamp)
;;    )
;; ;; Make it active
;; (ad-activate 'org-insert-todo-heading)

;; (require 'org-capture)

;; (defadvice org-capture (after mrb/created-timestamp-advice activate)
;;    "Insert a CREATED property using org-expiry.el for TODO entries"
;;                                         ; Test if the captured entry is a TODO,␣
↪if so insert the created
;;                                         ; timestamp property, otherwise ignore
;;    (mrb/insert-created-timestamp))
;; ;;  (when (member (org-get-todo-state) org-todo-keywords-1)
;; ;;    (mrb/insert-created-timestamp)))
;;    (ad-activate 'org-capture)
```

2.22.20 2.22.17.2 Enabling adding tags in the capture window

```
;; Add feature to allow easy adding of tags in a capture window
(defun mrb/add-tags-in-capture()
  (interactive)
  "Insert tags in a capture window without losing the point"
  (save-excursion
    (org-back-to-heading)
    (org-set-tags)))
;; Bind this to a reasonable key
(define-key org-capture-mode-map "\C-c\C-t" 'mrb/add-tags-in-capture)
```

- Note taken on [2019-02-07 Thu 09:11] This works fine now. However, it would be nice to find a way to strip the headers and menu columns and other unnecessary information before capture.

Source: http://www.bobnewell.net/publish/35years/webclipper.html

```
;; org-eww and org-w3m should be in your org distribution, but see
;; note below on patch level of org-eww.
(require 'org-eww)
(require 'org-w3m)
(defvar org-website-page-archive-file "~/my_org/full_article_archive.org")
(defun org-website-clipper ()
  "When capturing a website page, go to the right place in capture file,
   but do sneaky things. Because it's a w3m or eww page, we go
   ahead and insert the fixed-up page content, as I don't see a
   good way to do that from an org-capture template alone. Requires
```

(continues on next page)

(continued from previous page)

```
   Emacs 25 and the 2017-02-12 or later patched version of org-eww.el."
  (interactive)

  ;; Check for acceptable major mode (w3m or eww) and set up a couple of
  ;; browser specific values. Error if unknown mode.

  (cond
   ((eq major-mode 'w3m-mode)
    (org-w3m-copy-for-org-mode))
   ((eq major-mode 'eww-mode)
    (org-eww-copy-for-org-mode))
   (t
    (error "Not valid -- must be in w3m or eww mode")))

  ;; Check if we have a full path to the archive file.
  ;; Create any missing directories.

  (unless (file-exists-p org-website-page-archive-file)
    (let ((dir (file-name-directory org-website-page-archive-file)))
      (unless (file-exists-p dir)
        (make-directory dir))))

  ;; Open the archive file and yank in the content.
  ;; Headers are fixed up later by org-capture.

  (find-file org-website-page-archive-file)
  (goto-char (point-max))
  ;; Leave a blank line for org-capture to fill in
  ;; with a timestamp, URL, etc.
  (insert "\n\n")
  ;; Insert the web content but keep our place.
  (save-excursion (yank))
  ;; Don't keep the page info on the kill ring.
  ;; Also fix the yank pointer.
  (setq kill-ring (cdr kill-ring))
  (setq kill-ring-yank-pointer kill-ring)
  ;; Final repositioning.
  (forward-line -1)
  )
```

2.22.21 2.22.19.1 Loading language base

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((clojure . t)
   (scheme . t)
   (sqlite . t)
   (R . t)
   (lisp . t)
   (sql .  t)
   ;(jupyter . t)
   )
 )
```

2.22.22 2.22.19.2 Clojure and cider

```
(require 'cider)
(setq org-babel-clojure-backend 'cider)
```

2.22.23 2.22.21.1 Continuous clocking + punch in/out approach

This approach and code snippets are adapted (and shamelessly borrowed) from Bernt Hansen's approach. While Bernt follows a complex approach of clocking into parent tasks - my current workflow favors clocking in directly to set clocking headlines within projects, which are placed in my org-projectile todo task file.

I have a default continuous clock after punching in (defined by org-id) which will cater to general re-organisation, including capturing notes, refiling , email etc. Other tasks or even mini projects can be directly clocked into when required. These mini-projets are often just located within my org-agenda files and not as a separate git repositoy. Every time I am on my computer, whether on Emacs or not, I would like the automatic clock to capture time, unless it is being clocked to a specific project.

2.22.21.1.1 Defining default Task

```
(defvar sr/organization-task-id "a8712a47-a648-477f-bdbf-d6004a0cc70b")

(defun sr/clock-in-organization-task-as-default ()
  (interactive)
  (org-with-point-at (org-id-find sr/organization-task-id 'marker)
    (org-clock-in '(16))))
```

2.22.21.1.2 Punch in

Bernt Hansen shares that he has a default punch in and punch out task that keeps the clock on all day. I think this will work for me as well. Other than work and projects, most of the time I am tinkering with Emacs, or writing a journal note or trying to re-organise my stuff. By using a punch in and out, I can track how much time I am engaged with a computer, other than specific projects.

```
(defun sr/punch-in (arg)
    (interactive "p")
  (setq sr/keep-clock-running t)
  (sr/clock-in-organization-task-as-default))
```

2.22.21.1.3 Punch Out

```
(defun sr/punch-out ()
  (interactive)
  (setq sr/keep-clock-running nil)
  (when (org-clock-is-active)
    (org-clock-out))
  )
```

2.22.21.1.4 Advising clock Out

```
(defun sr/clock-out-maybe ()
  (when (and sr/keep-clock-running
             (not org-clock-clocking-in)
             (marker-buffer org-clock-default-task)
             (not org-clock-resolving-clocks-due-to-idleness))
    (sr/clock-in-organization-task-as-default)))

(add-hook 'org-clock-out-hook 'sr/clock-out-maybe 'append)
```

2.22.24 2.22.21.2 TEST org-mru-clock

- Note taken on [2019-03-14 Thu 10:16] Issue is with the org-mru-select-recent-task command - it doesn't jump to the specified task and always pesudo messes up the format of the headings.

This is a handy package to quickly select past tasks which have been clocked in.

```
(use-package org-mru-clock
  :ensure t
  :bind (("M-s 1" . org-mru-clock-in)
         ("C-c C-x C-j" . org-mru-clock-select-recent-task))
  :init
  (setq org-mru-clock-how-many 100
        org-mru-clock-completing-read #'ivy-completing-read))
```

2.22.25 2.22.21.3 Do not log or consider 0 Clocks

```
(setq org-clock-out-remove-zero-time-clocks t)
```

2.22.26 2.22.21.4 set idle timer for clocked task

```
;; setting idle timer to 15 minutes
(setq org-clock-idle-time 15)
```

2.22.27 2.22.21.5 Show clocked task history and enable re-clocking

Source: link

This should enable me to quickly clock back into specific tasks.

```
;; Show lot of clocking history so it's easy to pick items off the `C-c I` list
(setq org-clock-history-length 23)

(defun eos/org-clock-in ()
  (interactive)
  (org-clock-in '(4)))

(global-set-key (kbd "C-c I") #'eos/org-clock-in)
(global-set-key (kbd "C-c O") #'org-clock-out)
```

```
org-brain implements a variant of concept mapping in Emacs, using org-mode.

You can think of org-brain as a combination of a wiki and a mind map, where each wiki␣
↪page / mind map node is an org-mode file which resides in your org-brain-path, or a␣
↪headline with an ID property in one of those files. These are called entries.␣
↪Entries can be linked together, and you can then view the network of links as a␣
↪mind map, using M-x org-brain-visualize

`org-brain on github <https://github.com/Kungsgeten/org-brain>`_
```

2.22.28 2.22.22.1 STABLE Basic setup along with org-id

Since org-brain requires the org id for a heading to be recognized and displayed, it is convenient to have capture and refile mechanisms that create the org-id if the heading does not have it.

Further streamlining is necessary as such.

```
(use-package org-brain
  :ensure t
  :init
  (setq org-brain-path "~/my_org/brain/")
  ;; ;; For Evil users
  ;; (with-eval-after-load 'evil
  ;;   (evil-set-initial-state 'org-brain-visualize-mode 'emacs))
  :config
  (setq org-id-track-globally t)
  (setq org-id-locations-file "~/my_org/emacs_meta/.org-id-locations")
  (push '("b" "Brain" plain (function org-brain-goto-end)
          "* %i%?\n:PROPERTIES:\n:CREATED: [%<%Y-%m-%d %a %H:%M>]\n:END:" :empty-
↪lines 1)
        org-capture-templates)
  (setq org-brain-visualize-default-choices 'all)
  (setq org-brain-title-max-length 12)
  (add-hook 'org-brain-refile 'org-id-get-create)
  (global-set-key (kbd "M-s v") #'org-brain-visualize)
  )
```

2.22.29 2.22.23.1 Base config

```
(use-package org-journal
  :ensure t
  :defer t
  :custom
  (org-journal-dir "~/my_org/journal/")
  (org-journal-file-format "%Y%m%d")
  (org-journal-enable-agenda-integration t)
  )
```

2.22.30 2.22.23.2 setting org-capture template for Journal

```
(defun org-journal-find-location ()
  ;; Open today's journal, but specify a non-nil prefix argument in order to
```

(continues on next page)

(continued from previous page)

```
  ;; inhibit inserting the heading; org-capture will insert the heading.
  (org-journal-new-entry t)
  ;; Position point on the journal's top-level heading so that org-capture
  ;; will add the new entry as a child entry.
  (goto-char (point-min)))
```

2.22.31 2.22.23.3 TODO Figure out easy encryption approach for org journal

- Explore further options : example full path or customised path to be shown

```
This package displays in the header-line the Org heading for the node that's at the␣
↪top of the window. This way, if the heading for the text at the top of the window␣
↪is beyond the top of the window, you don't forget which heading the text belongs to.
↪ The display can be customized to show just the heading, the full outline path, or␣
↪the full outline path in reverse.

`org-sticky-header <https://github.com/alphapapa/org-sticky-header/blob/master/README.
↪org>`_
```

This is especially useful for free form longer Documentation.

```
(use-package org-sticky-header
  :ensure t
  :config
  (org-sticky-header-mode)
  )
```

- Note taken on [2019-03-28 Thu 13:48] This seems to be able to do exactly what I am looking for. However there are unexplained errors while starting up the package.

```
(use-package org-wild-notifier
  :ensure t
  :custom
  (require 'org-wild-notifier)
  (org-wild-notifier-mode)
  )
```

It is useful to be able to export to the RST format to develop documentation for projects and host with the sphinx or readthedocs platform.

This platform is actually pleasant to browse through documentation and has good search facilities as well.

```
(use-package ox-rst
  :ensure t
  :defer t
  :config
  (require 'ox-rst)
  )
```

This should prove handy as I write almost all my responses within Org mode and copy this into Slack.

```
(use-package ox-slack
  :ensure t
  :defer
  :config
```

(continues on next page)

(continued from previous page)

```
  (require 'ox-slack)
  )
```

```
(use-package ox-pandoc
  :ensure t
  :defer
  :config
  (require 'ox-pandoc)
  )
```

```
(setq org-columns-default-format "%50ITEM %TODO %3PRIORITY %10TAGS
↪%17Effort(Estimated Effort){:} %12CLOCKSUM")
```

```
(use-package org-sidebar
  :ensure t
  :defer nil
  )
```

CHAPTER 3

3 Stable Extras

These are packages and code snippets that I know to be working fine. Some still have rough edges and these are marked test. However not having these will not break my experience. Over time, some of these may make it to the core stable group.

3.1 3.1 TODO Time machine for git

- Note taken on [2019-02-08 Fri 13:21] Launched by `M-x git-timemachine`, this lets you navigate through the commit history with a single key press! This is especially awesome for tracking changes to a particular snippet of code.
- Note taken on [2019-02-07 Thu 09:30] Need to evaluate this. The purpose is for stepping through the history of a file recorded in git. This should be very interesting.

```
(use-package git-timemachine
  :ensure t)
```

3.2 3.2 Loading External Packages

Loading external packages: there are some packages which are not avaialble on MELPA and have to be loaded via cloning their git Repositories. This is especially relevant to new packages.

```
(let ((default-directory  "~/scimax/user/external_packages/"))
  (normal-top-level-add-subdirs-to-load-path))
```

3.3 3.3 memento mori

- Note taken on [2019-07-15 Mon 09:37] This package requires the birthdate to be specified, therefore it is included in the encrypted section of my personal config. A sample configuration is provided for reference.

This is a cool little package that displays your age with double decimal digits. A good reminder to get productive.

```
(use-package memento-mori
  :ensure t
  :defer nil
  :config
  (setq memento-mori-birth-date "2018-12-31")
  (memento-mori-mode 1)
  )
```

#+END_SRC

3.4 3.4 TEST Treemacs Setup

```
(use-package treemacs
  :ensure t
  :defer t
  :init
  (with-eval-after-load 'winum
    (define-key winum-keymap (kbd "M-0") #'treemacs-select-window))
  :config
  (progn
    (setq treemacs-collapse-dirs
          (if (executable-find "python3") 3 0)
          treemacs-deferred-git-apply-delay      0.5
          treemacs-display-in-side-window        t
          treemacs-eldoc-display                 t
          treemacs-file-event-delay              5000
          treemacs-file-follow-delay             0.2
          treemacs-follow-after-init             t
          treemacs-git-command-pipe              ""
          treemacs-goto-tag-strategy             'refetch-index
          treemacs-indentation                   2
          treemacs-indentation-string            " "
          treemacs-is-never-other-window         nil
          treemacs-max-git-entries               5000
          ttreemacs-no-png-images                 nil
          treemacs-no-delete-other-windows       t
          treemacs-project-follow-cleanup        nil
          treemacs-persist-file                  "~/my_org/emacs_meta/.treemacs-
↪persist"
          treemacs-recenter-distance             0.1
          treemacs-recenter-after-file-follow    nil
          treemacs-recenter-after-tag-follow     nil
          treemacs-recenter-after-project-jump   'always
          treemacs-recenter-after-project-expand 'on-distance
          treemacs-show-cursor                   nil
          treemacs-show-hidden-files             t
          treemacs-silent-filewatch              nil
          treemacs-silent-refresh                nil
          treemacs-sorting                       'alphabetic-desc
          treemacs-space-between-root-nodes      t
          treemacs-tag-follow-cleanup            t
          treemacs-tag-follow-delay              1.5
          treemacs-width                         35)
```

(continues on next page)

(continued from previous page)

```
    ;; The default width and height of the icons is 22 pixels. If you are
    ;; using a Hi-DPI display, uncomment this to double the icon size.
    ;;(treemacs-resize-icons 44)

    ;;(treemacs-follow-mode t)
    (treemacs-filewatch-mode t)
    (treemacs-fringe-indicator-mode t)
    (pcase (cons (not (null (executable-find "git")))
                 (not (null (executable-find "python3"))))
      (`(t . t)
       (treemacs-git-mode 'deferred))
      (`(t . _)
       (treemacs-git-mode 'simple))))
  :bind
  (:map global-map
        ("M-0"       . treemacs-select-window)
        ("M-s t t" . treemacs)
        ("M-s t w" . treemacs-switch-workspace)
        ;; ("C-x t 1"   . treemacs-delete-other-windows)
        ;; ("C-x t t"   . treemacs)
        ;; ("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-evil
;;   :after treemacs evil
;;   :ensure t)

(use-package treemacs-projectile
  :after treemacs projectile
  :ensure t)

(use-package treemacs-icons-dired
  :after treemacs dired
  :ensure t
  :config (treemacs-icons-dired-mode))

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

3.5 3.5 Scimax customisations

These are settings which custmise scimax specific variables. These are separated out here so that it becomes easier to try out Emacs configurations that are outside scimax.

Reference: https://oremacs.com/2019/05/18/hydra-0.15.0/

```
;; Adding helm ag to the search commands
(defhydra scimax-search (:color blue :inherit (scimax-base/heads) :columns 3)
  "search"
  ("a" counsel-ag "ag")
```

(continues on next page)

(continued from previous page)

```
  ("g" counsel-git-grep "grep")
  ("m" multi-moccur "moccur")
  ("o" occur "occur")
  ("p" projectile-grep "project grep")
  ("r" isearch-backward "search back")
  ("s" counsel-grep-or-swiper "search")
  ("h" helm-ag "helm-ag")
  ("t" counsel-pt "pt"))
```

The correct way of adding to a hydra is something like this. However this appears to work only after the original hydra has been executed atleast once.

```
(defhydra+ scimax-search ()
  ("h" helm-ag "helm-ag"))
```

3.5.1 3.5.2.1 TEST explorer

```
(defun explorer (&optional path)
  "Open Finder or Windows Explorer in the current directory."
  (interactive (list (if (buffer-file-name)
                         (file-name-directory (buffer-file-name))
                       (expand-file-name default-directory))))
  (cond
   ((string= system-type "gnu/linux")
    (shell-command "nautilus"))
   ((string= system-type "darwin")
    (shell-command (format "open -b com.apple.finder%s"
                           (if path (format " \"%s\""
                                            (file-name-directory
                                             (expand-file-name path))) ""))))
   ((string= system-type "windows-nt")
    (shell-command (format "explorer %s"
                           (replace-regexp-in-string
                            "/" "\\\\"
                            path))))))
```

3.5.2 3.5.2.2 bash

```
;; (defun sr/bash (args) ...)

(defun sr/bash (&optional path)
  "Open a bash window.
PATH is optional, and defaults to the current directory.
commands (`scimax-user-bash-app')
"
  (interactive (list (if (buffer-file-name)
                         (file-name-directory (buffer-file-name))
                       (expand-file-name default-directory))))
  (cond
   ((string= system-type "gnu/linux")
    (shell-command "xfce4-terminal"))
   ((string= system-type "darwin")
```

(continues on next page)

(continued from previous page)

```
    (shell-command
     (format "open -b com.apple.iterm2"
         (if path (format " \"%s\"" (expand-file-name path)) ""))))
   ((string= system-type "windows-nt")
    (shell-command "start \"\" \"%SYSTEMDRIVE%\\Program Files\\Git\\bin\\bash.exe\" --
↪login &"))))

(advice-add 'bash :override #'sr/bash)
```

```
(advice-remove 'bash #'scimax-user-bash-app)
```

- Note taken on [2019-03-07 Thu 16:24] Changing keyboard shortcut for equation insertion as this interferes with i3wm functioning.

Note: any expansion can be undone with C-/

```
(setq scimax-user-hotspot-commands
      '(("Agenda All" . (lambda () (org-agenda "" "a")))
        ("Agenda Office" . (lambda () (org-agenda "" "o")))
    ("Mail" . (lambda ()
                    (if (get-buffer "*mu4e-headers*")
                        (progn
                          (switch-to-buffer "*mu4e-headers*")
                          (delete-other-windows))
                      (mu4e))))
        ("Bookmarks" . (lambda () (helm-source-bookmarks)))
        ("Reload custom config - org babel" . (lambda () (org-babel-load-file (expand-
↪file-name "sr-config.org" user-emacs-directory))))
        )
      )

(setq scimax-user-hotspot-locations
      '(
        ("CV Org" . "~/Desktop/Resume_Shreyas_Ragavan.pdf")
        ("tmrs"  .  "~/my_org/tmsr.org")
        ("scd - scimax dir" . "~/scimax/" )
        ("scu - scimax user dir" . "~/scimax/user/")
        ( "sco - scimax org conf" . "~/scimax/user/sr-config.org")
        ("blog" . "~/my_org/blog-book.org")
    ("github" . "~/my_gits/")
        ("project" . "~/my_projects/")
        ("cheatsheet" . "~/my_projects/ds_cheatsheets/")
        ("passwords" . "~/my_org/secrets.org.gpg")
        ("references" . "~/Dropbox/bibliography/references.bib")
        ("R Lib source src folder" . "/Library/Frameworks/R.framework/Resources/
↪library")
        )
      )
```

```
(require 'scimax-elfeed)
```

```
(setq nb-notebook-directory "~/my_projects/")
```

```
(global-set-key (kbd "M-s n") 'nb-open)
```

```
(global-set-key (kbd "C-\\") 'scimax/body)
```

```
(require 'scimax-org-babel-python)
(require 'ob-ipython)
(require 'scimax-ob)
(require 'scimax-org-babel-ipython-upstream)
(setq ob-ipython-exception-results nil)
(scimax-ob-ipython-turn-on-eldoc)
```

```
(message "Loaded scimax customisations")
```

3.6 3.6 Swiper

- Note taken on [2019-09-24 Tue 10:31] Swiper and below keybinding is included by default in scimax.
- Note taken on [2019-02-07 Thu 16:50] I use swiper for a general search. However *3F1BAD63-98A3-4BF0-B5DD-67ED63D0AFEB* is awesome.

```
(global-set-key (kbd "C-s") 'swiper)
(setq ivy-display-style 'fancy)

;; advise swiper to recenter on exit
(defun bjm-swiper-recenter (&rest args)
  "recenter display after swiper"
  (recenter)
  )
(advice-add 'swiper :after #'bjm-swiper-recenter)

(message "Loaded Swiper customisation")
```

3.7 3.7 Writeroom customisations

The goal is to enable a customised zen writing mode, especially facilitating blog posts and other longer forms of writing. As of now, there are customisations for the width, and calling the art-bollocks mode when writeroom mode is enabled.

```
(with-eval-after-load 'writeroom-mode
  (define-key writeroom-mode-map (kbd "C-s-,") #'writeroom-decrease-width)
  (define-key writeroom-mode-map (kbd "C-s-.") #'writeroom-increase-width)
  (define-key writeroom-mode-map (kbd "C-s-=") #'writeroom-adjust-width))

(advice-add 'text-scale-adjust :after
        #'visual-fill-column-adjust)

;;  loading artbollocks whenever the writeroom mode is called in particular.
(autoload 'artbollocks-mode "artbollocks-mode")
(add-hook 'writeroom-mode-hook 'artbollocks-mode)

(message "Loaded writeroom customisations")
```

3.8 3.8 TEST ESS configuration

- Note taken on [2019-02-19 Tue 10:14] Using the tabviewer application for Antergos.‘link <https://bbs.archlinux.org/viewtopic.php?id=156295>‘_
- Note taken on [2019-02-09 Sat 12:36] Set this up with use-package and explore further customisations. As of now, I use yasnippet to insert commonly used operators like the assign and pipe operators.

Note: I use the TAD application to view CSV files. It is a cross platform application that is a lot faster than launching a spreadsheet based program.

```
(use-package ess
  :ensure t
  :config
  (require 'ess)
  (use-package ess-R-data-view)
  (use-package polymode)
  (setq ess-describe-at-point-method nil)
  (setq ess-switch-to-end-of-proc-buffer t)
  (setq ess-rutils-keys +1)
  (setq ess-eval-visibly 'nil)
  (setq ess-use-flymake +1)
  (setq ess-use-company t)
  (setq ess-history-file "~/.Rhistory")
  (setq ess-use-ido t)
  (setq ess-roxy-hide-show-p t)
  ;;(speedbar-add-supported-extension ".R")
  (setq comint-scroll-to-bottom-on-input t)
  (setq comint-scroll-to-bottom-on-output t)
  (setq comint-move-point-for-output t)
  )

;; The following chunk is taken from: https://github.com/syl20bnr/spacemacs/blob/
↪master/layers/%2Blang/ess/packages.el
;;; Follow Hadley Wickham's R style guide
(setq ess-first-continued-statement-offset 2
      ess-continued-statement-offset 0
      ess-expression-offset 2
      ess-nuke-trailing-whitespace-p t
      ess-default-style 'DEFAULT)


;; Adding Poly-R package

(use-package poly-R
  :ensure t
  )
;; The following chunk is taken from antonio's answer from https://stackoverflow.com/
↪questions/16172345/how-can-i-use-emacs-ess-mode-with-r-markdown
(defun rmd-mode ()
  "ESS Markdown mode for rmd files."
  (interactive)
  (require 'poly-R)
  (require 'poly-markdown)
  (poly-markdown+r-mode))

(use-package ess-view
  :ensure t
```

(continues on next page)

(continued from previous page)

```
  :config
  (require 'ess-view)
  (if (system-type-is-darwin)
      (setq ess-view--spreadsheet-program
            "/Applications/Tad.app/Contents/MacOS/Tad"
            )
    )
  (if (system-type-is-gnu)
      (setq ess-view--spreadsheet-program
            "tad"
            )
    )
  )

;; This is taken and slightly modified from the ESS manual
;; The display config is similar to that of Rstudio

(setq display-buffer-alist
      `(("*R Dired"
         (display-buffer-reuse-window display-buffer-in-side-window)
         (side . right)
         (slot . -1)
         (window-width . 0.33)
         (reusable-frames . nil))
        ("*R"
         (display-buffer-reuse-window display-buffer-at-bottom)
         (window-width . 0.35)
         (reusable-frames . nil))
        ("*Help"
         (display-buffer-reuse-window display-buffer-in-side-window)
         (side . right)
         (slot . 1)
         (window-width . 0.33)
         (reusable-frames . nil))))

(message "Loaded ESS configuration")
```

```
;; (use-package ox-ravel
;; :ensure t
;; :defer nil
;; (require 'ox-ravel)
;; )
```

3.9 3.9 ox-reveal - presentations

```
(use-package ox-reveal
  :ensure ox-reveal
  :defer nil
  :config
  (setq org-reveal-root "http://cdn.jsdelivr.net/reveal.js/3.0.0/")
  (setq org-reveal-mathjax t)
  )
```

(continues on next page)

(continued from previous page)

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

(message "Loaded ox-reveal cust")
```

3.10 3.10 Deft

```
Deft is an Emacs mode for quickly browsing, filtering, and editing directories of␣
↪plain text notes, inspired by Notational Velocity. It was designed for increased␣
↪productivity when writing and taking notes by making it fast and simple to find the␣
↪right file at the right time and by automating many of the usual tasks such as␣
↪creating new files and saving files.

`Deft project <https://jblevins.org/projects/deft/>`_
```

```
(use-package deft
  :bind ("<f8> d" . deft)
  :commands (deft)
  :config (setq deft-directory "~/my_org/brain/"
                deft-extensions '("md" "org" "txt")
                deft-recursive t
                ))
```

3.11 3.11 w3m customisation

w3m is a suprisingly able browser that is able to cater to most websites, except those that are a little too rich with java and etc. Being within Emacs, and launching almost instantly with significantly less overhead in terms of RAM no matter how many tabs are open - it is also easy to customise the behavior as needed and is an excellent method of distraction free browsing.

However, it pays to have handy shortcuts to open a link in the default browser of the OS. This is especially to cater to heavier websites. The w3m package would need to be installed using the package manager of the OS to use w3m.

A few snippets were sourced from: http://beatofthegeek.com/2014/02/my-setup-for-using-emacs-as-web-browser.html

```
;;(setq browse-url-browser-function 'browse-url-default-browser)
(setq browse-url-browser-function 'w3m-goto-url-new-session)
(setq w3m-default-display-inline-images t)
```

- Note taken on [2019-02-07 Thu 07:40] Check whether this is necessary

```
;;when I want to enter the web address all by hand
(defun w3m-open-site (site)
  "Opens site in new w3m session with 'http://' appended"
  (interactive
   (list (read-string "Enter website address(default: w3m-home):" nil nil w3m-home-
↪page nil )))
  (w3m-goto-url-new-session
   (concat "http://" site)))
```

Source: Sacha Chua : http://sachachua.com/blog/2008/09/emacs-and-w3m-making-tabbed-browsing-easier/

```
(eval-after-load 'w3m
  '(progn
     (define-key w3m-mode-map "q" 'w3m-previous-buffer)
     (define-key w3m-mode-map "w" 'w3m-next-buffer)
     (define-key w3m-mode-map "x" 'w3m-close-window)))
```

- Note taken on [2019-02-07 Thu 07:37] Need to have this change depending whether the OS is Linux or Mac OS

```
(defun wicked/w3m-open-current-page-in-default-browser ()
  "Open the current URL in Mozilla Firefox."
  (interactive)
  (browse-url-default-browser w3m-current-url)) ;; (1)

(defun wicked/w3m-open-link-or-image-in-default-browser ()
  "Open the current link or image in Firefox."
  (interactive)
  (browse-url-default-browser (or (w3m-anchor) ;; (2)
                                  (w3m-image)))) ;; (3)

(eval-after-load 'w3m
  '(progn
     (define-key w3m-mode-map "o" 'wicked/w3m-open-current-page-in-default-browser)
     (define-key w3m-mode-map "O" 'wicked/w3m-open-link-or-image-in-default-browser)))
```

```
(defun wikipedia-search (search-term)
  "Search for SEARCH-TERM on wikipedia"
  (interactive
   (let ((term (if mark-active
                   (buffer-substring (region-beginning) (region-end))
                 (word-at-point))))
     (list
      (read-string
       (format "Wikipedia (%s):" term) nil nil term)))
   )
  (browse-url
   (concat
    "http://en.m.wikipedia.org/w/index.php?search="
    search-term
    ))
  )
```

```
(defun hn ()
  (interactive)
  (browse-url "http://news.ycombinator.com"))
```

- Note taken on [2019-03-07 Thu 11:59] This is worth setting up. It would be convenient for frequently visited websites like reddit and others, to open in the external browser, especially as they do not render well within w3m.

Source : http://ergoemacs.org/emacs/emacs_set_default_browser.Html

```
;; use browser depending on url
(setq
 browse-url-browser-function
 '(
  ("wikipedia\\.org" . browse-url-firefox)
  ("github" . browse-url-chromium)
```

(continues on next page)

(continued from previous page)

```
  ("thefreedictionary\\.com" . eww-browse-url)
  ("." . browse-url-default-browser)
  ))
```

3.12 3.12 frog jump buffer

This package provides a nifty little pop up containing a list of buffers (that can be filtered), and enables jumping to the specified buffer with just a single key press.

```
(use-package frog-jump-buffer
  :ensure t
  :defer nil
  :config
)
```

3.13 3.13 easy-kill and easy mark

- Note taken on [2019-04-25 Thu 07:48] The line selection, 'e', does not pick up lines separated with a full stop. Instead the entire paragraph is being selected.

```
Provide commands easy-kill and easy-mark to let users kill or mark things easily.

`Github <https://github.com/leoliu/easy-kill>`_
```

This looks like a very handy package. The easiest way to get started is to cycle through the selections and use the help. Activate the command with `M-w` and `?` for help which provides the list of key bindings. Alternately, use `SPC` to cycle through the options available.

```
(use-package easy-kill
  :config
  (global-set-key [remap kill-ring-save] 'easy-kill)
  (global-set-key [remap mark-sexp] 'easy-mark)
  )
```

3.14 3.14 TEST eyebrowse

This has to be combined with desktop.el or some other method to enable persistence across sessions. However, this does work well for a single session.

```
(use-package eyebrowse
  :ensure t
  :defer nil
  :config
  (setq eyebrowse-mode-line-separator " "
        eyebrowse-new-workspace t)
  (eyebrowse-mode 1)
  )
```

3.15 3.15 Hugo

Modified this function from:

3.15.1 3.15.1.1 TODO Defining content directory

- Note taken on [2019-02-07 Thu 08:06] Need to check if this is still required since I have switche to ox-hugo

```
(defvar hugo-content-dir "~/my_gits/hugo-sr/content/post/"
  "Path to Hugo's content directory")
```

3.15.2 3.15.1.2 Ensuring properties exist and creating if they dont exist

```
(defun hugo-ensure-property (property)
  "Make sure that a property exists. If not, it will be created.
Returns the property name if the property has been created, otherwise nil."
  (org-id-get-create)
  (if (org-entry-get nil property)
      nil
    (progn (org-entry-put nil property "")
           property)))

(defun hugo-ensure-properties ()

  (require 'dash)
  (let ((current-time (format-time-string
                       (org-time-stamp-format t t) (org-current-time)))
        first)
    (save-excursion
      (setq first (--first it (mapcar #'hugo-ensure-property
                                      '("HUGO_TAGS" "HUGO_CATEGORIES"))))
      (unless (org-entry-get nil "HUGO_DATE")
        (org-entry-put nil "EXPORT_DATE" current-time)))
    (org-entry-put nil "EXPORT_FILE_NAME" (org-id-get-create))
    (org-entry-put nil "EXPORT_HUGO_CUSTOM_FRONT_MATTER" ":profile false")
    (when first
      (goto-char (org-entry-beginning-position))
      ;; The following opens the drawer
      (forward-line 1)
      (beginning-of-line 1)
      (when (looking-at org-drawer-regexp)
        (org-flag-drawer nil))
      ;; And now move to the drawer property
      (search-forward (concat ":" first ":"))
      (end-of-line))
    first))
```

3.15.3 3.15.1.3 Hugo function calling the above

```
(defun hugo ()
  (interactive)
  (unless (hugo-ensure-properties)
```

(continues on next page)

(continued from previous page)

```
    (let* ((type     (concat "type = \"" (org-entry-get nil "HUGO_TYPE") "\"\n"))
           (date     (concat "date = \""
                             (format-time-string "%Y-%m-%d"
                                                 (apply 'encode-time
                                                        (org-parse-time-string
                                                         (org-entry-get nil "HUGO_DATE
↪"))) t) "\"\n"))
           (tags     (concat "tags = [ \""
                             (mapconcat 'identity
                                        (split-string
                                         (org-entry-get nil "HUGO_TAGS")
                                         "\\( *, *\\)" t) "\", \"") "\" ]\n"))
           (fm (concat "+++\n"
                       title
                   type
                       date
                       tags
                       topics
                       "+++\n\n"))
           (coding-system-for-write buffer-file-coding-system)
           (backend  'md)
           (blog))
      ;; try to load org-mode/contrib/lisp/ox-gfm.el and use it as backend
      (if (require 'ox-gfm nil t)
          (setq backend 'gfm)
        (require 'ox-md))
      (setq blog (org-export-as backend t))
      ;; Normalize save file path
      (unless (string-match "^[/~]" file)
        (setq file (concat hugo-content-dir file))
        (unless (string-match "\\.md$" file)
          (setq file (concat file ".md")))
        ;; save markdown
        (with-temp-buffer
          (insert fm)
          (insert blog)
          (untabify (point-min) (point-max))
          (write-file file)
          (message "Exported to %s" file))))))
```

```
(use-package ox-hugo
  :ensure t
  :defer t
  :after ox
  :custom
  (org-hugo--tag-processing-fn-replace-with-hyphens-maybe t)
  )
```

3.16 3.16 STABLE PDF Tools

- Note taken on [2019-10-23 Wed 09:26] This appears to be setup via scimax already. Disabling for now.
- Note taken on [2019-02-18 Mon 14:30] Install epdfinfo via 'brew install pdf-tools' and then install the pdf-tools elisp via the use-package below. To upgrade the epdfinfo server, use 'brew upgrade pdf-tools' prior to upgrading

to newest pdf-tools package using Emacs package system. If things get messed up, just do 'brew uninstall pdf-tools', wipe out the elpa pdf-tools package and reinstall both as at the start. source: https://emacs.stackexchange.com/questions/13314/install-pdf-tools-on-emacs-macosx

```
(use-package pdf-tools
  :ensure t
  :config
  (custom-set-variables
   '(pdf-tools-handle-upgrades nil)) ; Use brew upgrade pdf-tools instead in the mac
  (setq pdf-info-epdfinfo-program "/usr/local/bin/epdfinfo")
  (pdf-tools-install)
)
```

CHAPTER 4

4 Testing

These are packages and functions that I am exploring, and also ones that I can live without. i.e not having these packages functioning, will not make Emacs useless for me, however, having these snippets working could potentially improve my workflow in general.

Some of these snippets or packages are enabled and some are not. In the case of troubleshooting, I will disable

4.1 4.1 Oddmuse curl

```
(add-to-list 'load-path "~/scimax-personal/external_packages/oddmuse-curl/")
(setq oddmuse-username "shrysr")
(setq oddmuse-directory "~/my_org/01_wiki/oddmuse/")
(add-to-list 'auto-mode-alist '("~/my_org/01_wiki/oddmuse" . oddmuse-mode))
(autoload 'oddmuse-edit "oddmuse-curl"
  "Edit a page on an Oddmuse wiki." t)
(add-to-list 'vc-handled-backends 'oddmuse)
(defun vc-oddmuse-registered (file)
  "Handle files in `oddmuse-directory'."
  (string-match (concat "^" (expand-file-name oddmuse-directory))
                (file-name-directory file)))

(require 'oddmuse)
 ;; (oddmuse-mode-initialize)
```

4.2 4.2 Scimax cusomisations

- Note taken on [2019-09-25 Wed 13:15] Adding the shortcut to new entry does not work. This has to be refined.

```
;; (setq scimax-journal-root-dir "~/my_org/journal")
;;(require 'calendar)
;;(global-set-key (kbd "M-s j") 'scimax-journal-new-entry)
```

This was setup a long time ago to convert past technical repots into org mode, with references made in correct technical style. This project is on hold.

```
(require 'doi-utils)
(require 'org-ref-wos)
(require 'org-ref-pubmed)
(require 'org-ref-arxiv)
(require 'org-ref-bibtex)
(require 'org-ref-pdf)
(require 'org-ref-url-utils)
(require 'org-ref-helm)

;; note and bib location

(setq org-ref-bibliography-notes "~/my_org/references/references.org"
      org-ref-bibliography-notes "~/my_org/references/research_notes.org"
      org-ref-default-bibliography '("~/my_org/references/references.bib")
      org-ref-pdf-directory "~/my_org/references/pdfs/")

;; setting up helm-bibtex
(setq helm-bibtex-bibliography "~/my_org/references/references.bib"
      helm-bibtex-library-path "~/my_org/org/references/pdfs"
      helm-bibtex-notes-path "~/my_org/references/research_notes.org")
```

```
(ivy-add-actions
 'projectile-completing-read
 '(("b" (lambda (x)
      (bash x))  "Open bash here.")
   ("f" (lambda (x)
      (finder x))  "Open Finder here.")
   ("a" (lambda (x)
      (helm-do-ag x) "helm do ag here."))))
```

```
(if (system-name-is-darwin)
    (setq mac-)
  )
```

```
\`org-db' is an org-mode database. When it is active every org-mode file you visit␣
↪will be indexed into a sqlite database. In each file, each headline with its title,␣
↪tags and properties are stored, and every link in each file is stored.

This becomes useful because you can then search all of your org-files and jump to␣
↪different locations.

Scimax help documentation
```

```
(use-package emacsql-sqlite
  :ensure t
  :config
  (require 'org-db)
)
```

4.3 4.3 TODO Tangle org mode config on save

- Note taken on [2019-08-28 Wed 13:44] Though this is an after-save hook : there is something wrong with the method of tangling. Perhaps an older version of the file is being used. I have changed this to a tangle during the loading of Scimax.
- Note taken on [2019-02-14 Thu 13:14] Need to add a condition of check: tangle if the file does not exist.

Source: https://thewanderingcoder.com/2015/02/literate-emacs-configuration/

This is a nice code snippet to automate the tangling on saving the config. This saves time while starting up Emacs...

```
(defun sr/tangle-on-save-emacs-config-org-file()
  (interactive)
  (if (string= buffer-file-name (file-truename "~/scimax/user/sr-config.org"))
      (org-babel-tangle-file "~/scimax/user/sr-config.org" "~/scimax/user/sr-config.el
↪")
    )
  )

(defun sr/tangle-if-file-absent ()
  (interactive)
  (if nil  (file-exists-p "~/scimax/user/sr-config.el")
    (org-babel-tangle-file  "~/scimax/user/sr-config.org" "~/scimax/user/sr-config.el
↪")
    )
  )
;; (add-hook 'after-save-hook 'sr/dotemacs-export)
(add-hook 'after-save-hook
          'sr/tangle-on-save-emacs-config-org-file)
```

4.4 4.4 TEST Visual line and visual fill column

```
(use-package visual-fill-column
  :ensure t
  :config
  (setq visual-fill-column-width 80)
  (setq global-visual-fill-column-mode 1)
  (setq global-visual-line-mode 1)
)

;;(add-hook 'visual-line-mode-hook #'visual-fill-column-mode)
```

4.5 4.5 TEST Marking

I want a way to efficiently mark a location in a long script and jump around these locations (forward and backwards). The `transient-mark-mode` and the different mark-rings need to be leveraged to do accomplish this. First step is to set a mark using `C-spc C-spc`.

Adopting the approach described at Mastering Emacs. This enables a single key for a mark to activate and then deactivate, thus creating a mark.

```
(defun push-mark-no-activate ()
  "Pushes `point' to `mark-ring' and does not activate the region
   Equivalent to \\[set-mark-command] when \\[transient-mark-mode] is disabled"
  (interactive)
  (push-mark (point) t nil)
  (message "Pushed mark to ring"))

(global-set-key (kbd "C-`") 'push-mark-no-activate)
```

The `tmm-menu` command's shortcut `M-`` is much better served by `M-x counsel-tmm` where search is possible.

```
(defun jump-to-mark ()
  "Jumps to the local mark, respecting the `mark-ring' order.
  This is the same as using \\[set-mark-command] with the prefix argument."
  (interactive)
  (set-mark-command 1))
(global-set-key (kbd "M-`") 'jump-to-mark)
```

4.6 4.6 TEST Semantic Mode

```
Semantic is a package that provides language-aware editing commands based on 'source-
↪code parsers'. When enabled, each file you visit is automatically parsed.

`https://tuhdo.github.io/helm-intro.html <https://tuhdo.github.io/helm-intro.html>`_
```

```
(semantic-mode 1)
```

4.7 4.7 TEST Sauron

```
(use-package sauron
  :ensure t
  :config
  (require 'sauron)
  (setq sauron-modules '(sauron-org sauron-notifications))
  )
```

4.8 4.8 emacs url shortener

Note that to use one of the url shortening services, an API access token will be needed. Currently, I am using bitly.

```
(use-package url-shortener
  :ensure t
  :defer t
  )
```

4.9 4.9 free-keys

This seems to be a nifty little package that lists the key combinations that are not bound.

```
(use-package free-keys
  :defer t
  :ensure t
)
```

4.10 4.10 Zenburn theme exploration

For a long period, I was using the zenburn theme and had started customising it for my needs. However, I think leuven with a greyish background is really quite suitable. Even so, it's nice to have a dark theme available when required. I'm keeping this around for tinkering down the line.

```
(disable-theme 'leuven)
;;(load-theme 'spacemacs-dark t)
(load-theme 'zenburn t)
```

This is necessary due to the customisation in scimax

```
(set-face-background 'org-block-emacs-lisp "black")
(set-face-background 'org-block-python "black")
(set-face-background 'org-block-ipython "black")
(set-face-background 'org-block "black")
;; (set-face-background 'org-block-quote "black")
```

```
;; use variable-pitch fonts for some headings and titles
(setq zenburn-use-variable-pitch t)

;; scale headings in org-mode
(setq zenburn-scale-org-headlines t)

;; scale headings in outline-mode
(setq zenburn-scale-outline-headlines t)
```

Source: https://github.com/m-parashar/emax64/issues/5

```
(use-package zenburn-theme
  :demand t
  :config
  (load-theme 'zenburn t)
  (set-face-attribute 'font-lock-comment-face nil :italic t)
  (set-face-attribute 'font-lock-doc-face nil :italic t)
  (zenburn-with-color-variables
    (set-face-attribute 'button nil :foreground zenburn-yellow-2)
    (set-face-attribute 'default nil
                        :background zenburn-bg-05
                        :height mp/font-size-default
                        :font mp/font-family)
    (set-face-attribute 'help-argument-name nil :foreground zenburn-orange :italic␣
↪nil)
    (set-face-attribute 'hl-line nil :background zenburn-bg+1)
    (set-face-attribute 'header-line nil
```

(continues on next page)

(continued from previous page)

```
                        :background zenburn-bg-1
                        :box `(:line-width 2 :color ,zenburn-bg-1)
                        :height mp/font-size-header-line)
    (set-face-attribute 'mode-line nil
                        :box `(:line-width 2 :color ,zenburn-bg-1)
                        :foreground zenburn-bg+3
                        :height mp/font-size-mode-line)
    (set-face-attribute 'mode-line-inactive nil
                        :box `(:line-width 2 :color ,zenburn-bg-05)
                        :foreground zenburn-bg+3
                        :height mp/font-size-mode-line)
    (set-face-attribute 'region nil
                        :background zenburn-fg-1
                        :distant-foreground 'unspecified)
    (set-face-attribute 'vertical-border nil :foreground zenburn-bg))

  ;; NOTE: See https://github.com/bbatsov/zenburn-emacs/issues/278.
  (zenburn-with-color-variables
    (mapc
     (lambda (face)
       (when (eq (face-attribute face :background) zenburn-bg)
         (set-face-attribute face nil :background 'unspecified)))
     (face-list))))
```

- Note taken on [2019-03-28 Thu 07:09] This is available as in-built settings for the zenburn theme. However, once the font is changed, the

```
(custom-set-faces
 '(org-level-1 ((t (:inherit outline-1 :height 1.2))))
 '(org-level-2 ((t (:inherit outline-2 :height 1.1))))
 '(org-level-3 ((t (:inherit outline-3 :height 1.05))))
 '(org-level-4 ((t (:inherit outline-4 :height 1.00))))
 '(org-level-5 ((t (:inherit outline-5 :height .95))))
 )
```

4.11 4.11 TEST Alfred Integration

Source: https://github.com/jjasghar/alfred-org-capture

```
(if (system-type-is-darwin)
    (progn
      ;;; Code:
      (defun make-orgcapture-frame ()
        "Create a new frame and run org-capture."
        (interactive)
        (make-frame '((name . "remember") (width . 80) (height . 16)
                      (top . 400) (left . 300)
                      (font . "-apple-Monaco-medium-normal-normal-*-13-*-*-*-m-0-
↪iso10646-1")
                      ))
        (select-frame-by-name "remember")
        (org-capture))
      )
  )
```

4.12 4.12 TODO Project publishing setup [0/3]

This is under construction and was initially started with the idea of having custom publishing settings for different projects. I was initially looking at this for publishing my hugo blog. However, the need has been negated with the excellent ox-hugo package.

- Note taken on [2019-07-04 Thu 11:20] Minor experiments are completed with this package. However, detailed exploration is required to incorporate intoa workflow.

```
This is an export backend for Org-mode that exports buffers to HTML that is␣
↪compatible with Tufte CSS out of the box (meaning no CSS modifications needed).

It's still a work-in-progress, but it works well right now.

`Github <https://github.com/dakrone/ox-tufte>`_
```

```
(use-package ox-tufte
  :defer t
  :config
  (require 'ox-tufte)
  )
```

```
(
 setq org-publish-project-alist
 '(
   ("org-repo"
    :base-directory "./"
    :base-extension "org"
    :publishing-directory "/Users/shreyas/my_projects/dotemacs"
    :EXPORT_FILE_NAME "README.org"
    :recursive f
    :publishing-function org-html-publish-to-html
    ;; :html-head "<link rel="stylesheet" href="http://dakrone.github.io/org2.css"␣
↪type="text/css" />"
    )

   ("md"
    :base-directory "./"
    :base-extension "org"
    :publishing-directory "./export/"
    :recursive t
    :publishing-function org-md-export-to-markdown
    )

   ("Documentation - html + md"
    :components ("html-static" "md" )
    )))
```

- Note taken on [2019-02-14 Thu 14:05] Save the filename as variables.
- Note taken on [2019-02-14 Thu 13:30] Add a condition to check if the directory exists.
- Note taken on [2019-02-10 Sun 07:16] Add a line to revert target export files if they are open. Prefer exporting the org file rather than copying.

This is the beginning of a function to perform 3 exports:

1. Export to my hugo website as a part of my documentation (ox-hugo)

2. Copy the org file to my github repo.
3. Tangle the copied org file to the above github repository to have the script ready.

Maintaining the documentation on my website does not make it easy to others to view the changes in the configuration and fork or download the same as an org file or emacs-lisp script. Therefore the config that I publish should be maintained in it's own repository.

As of now, I'm calling this function from my Emacs config file, and need to improve the above workflow.

```
(defun sr/dotemacs-export()
  (interactive)
  "If directories exist - exporting Org config to Hugo blog, and to Github repository␣
↪org file and lisp"

  (if (file-directory-p "~/my_projects/dotemacs")
      (progn
        (copy-file "~/scimax/user/sr-config.org" "~/my_projects/dotemacs/README.org"
↪"OK-IF-ALREADY-EXISTS")
        (copy-file "~/scimax/user/sr-config.el" "~/my_projects/dotemacs/config.el"
↪"OK-IF-ALREADY-EXISTS")
        ;; (org-babel-tangle-file  "~/my_projects/dotemacs/README.org" "~/my_projects/
↪dotemacs/config.el")
        )
    )

  )
```

4.13 4.13 TEST Docker

I've started playing with Docker, and need Emacs to take care of my workflows! :).

The `docker` package enables several commands, but does not seem to include syntax highlighting for docker files.

```
(use-package docker
  :ensure t
  :defer t
  :config
  (require 'docker)
)
```

```
(use-package docker-compose-mode
  :ensure t
  :defer t
  :config
  (require 'docker-compose-mode)
  )
```

This is required for syntax highlighting in dockerfiles.

```
(use-package dockerfile-mode
  :ensure t
  :defer t
  :config
  (require 'dockerfile-mode)
  (add-to-list 'auto-mode-alist
```

(continues on next page)

(continued from previous page)

```
                '("Dockerfile\\" . dockerfile-mode)
                )
  )
```

4.14 4.14 org-bookmark-heading

For some reason, the default bookmark behavior in org mode is that the bookmark is not linked to the org-id. This means that if the heading is shifted somewhere, the bookmark becomes useless! The remedy seems to be using the package org-bookmark-Heading

```
(use-package org-bookmark-heading
  :ensure t
  :config
  (require 'org-bookmark-heading)
)
```

4.15 4.15 TODO Crux - basic movement

Source: https://jamiecollinson.com/blog/my-emacs-config/ Contains functions from Prelude. I should check this out in more detail.

Set C-a to move to the first non-whitespace character on a line, and then to toggle between that and the beginning of the line.

```
(use-package crux
  :ensure t
  :bind (("C-a" . crux-move-beginning-of-line)))
```

4.16 4.16 Crypto setup

```
(setq epa-file-encrypt-to "shreyas@fastmail.com")
(require 'org-crypt)
(add-to-list 'org-modules 'org-crypt)
                                        ; Encrypt all entries before saving
(org-crypt-use-before-save-magic)
;;(setq org-tags-exclude-from-inheritance (quote ("crypt")))
                                        ; GPG key to use for encryption. nil for␣
↪symmetric encryption
;;(setq org-crypt-key nil)
(setq org-crypt-disable-auto-save t)
;;(setq org-crypt-tag-matcher "locked")

(message "Loaded crypto setup")
```

This was prompted by this discussion https://emacs.stackexchange.com/questions/10207/how-to-get-org2blog-to-use-authinfo-gpg

I have modified it to my own file names.

```
(require 'auth-source)

(setq
 auth-sources '(default
                "secrets:session"
                "secrets:Login"
                "~/.gh.authinfo.gpg"
                "~/.netrc.gpg"
                "~/.bitly-access.token.gpg"
                )
 epa-file-cache-passphrase-for-symmetric-encryption t)
```

Source: https://emacs.stackexchange.com/questions/40994/using-auth-source-with-magit-and-bitbucket

Fill the out the following details before executing the script. Machine can be found be executing 'hostname' in shell.

```
cat > ~/.gh.authinfo << EOF
machine shrysr@github.com password ABCD
EOF
```

`M-x epa-encrypt-file` and point towards the above file and choose your key for encryptions. This will generate the .gpg file.

Remove the original file when done.

```
rm -f ~/.gh.authinfo
```

```
(setq magit-process-find-password-functions '(magit-process-password-auth-source))
```

4.17 4.17 TODO Persp-projectile

Refer Howard's config snippet to setup a test.

4.18 4.18 TODO LOB

- Note taken on [2019-04-25 Thu 07:39] Since shifting to using org-brain for permanent notes and snippets, I need to review this ingest.

There are a bunch of scripts that I would like ingested into the Library of Babel to be available for ready use. In some cases, with specific and relatively simple actions these are useful, and generally easier to define that Emacs Functions.

```
(org-babel-lob-ingest "~/my_projects/sr-snip-lob/README.org")
```

4.19 4.19 Hydras and some custom functions

Adapted from https://emacs.stackexchange.com/questions/8045/org-refile-to-a-known-fixed-location

source: https://gist.github.com/mm–/60e0790bcbf8447160cc87a66dc949ab

Also see

```
(defun my/refile (file headline &optional arg)
  "Refile to a specific location.

With a 'C-u' ARG argument, we jump to that location (see
`org-refile').

Use `org-agenda-refile' in `org-agenda' mode."
  (let* ((pos (with-current-buffer (or (get-buffer file) ;Is the file open in a␣
↪buffer already?
                                       (find-file-noselect file)) ;Otherwise, try to find␣
↪the file by name (Note, default-directory matters here if it isn't absolute)
            (or (org-find-exact-headline-in-buffer headline)
                (error "Can't find headline `%s'" headline))))
     (filepath (buffer-file-name (marker-buffer pos))) ;If we're given a relative␣
↪name, find absolute path
     (rfloc (list headline filepath nil pos)))
    (if (and (eq major-mode 'org-agenda-mode) (not (and arg (listp arg)))) ;Don't use␣
↪org-agenda-refile if we're just jumping
    (org-agenda-refile nil rfloc)
      (org-refile arg nil rfloc))))

(defun josh/refile (file headline &optional arg)
  "Refile to HEADLINE in FILE. Clean up org-capture if it's activated.

With a `C-u` ARG, just jump to the headline."
  (interactive "P")
  (let ((is-capturing (and (boundp 'org-capture-mode) org-capture-mode)))
    (cond
     ((and arg (listp arg))     ;Are we jumping?
      (my/refile file headline arg))
     ;; Are we in org-capture-mode?
     (is-capturing          ;Minor mode variable that's defined when capturing
      (josh/org-capture-refile-but-with-args file headline arg))
     (t
      (my/refile file headline arg)))
    (when (or arg is-capturing)
      (setq hydra-deactivate t))))

(defun josh/org-capture-refile-but-with-args (file headline &optional arg)
  "Copied from `org-capture-refile' since it doesn't allow passing arguments. This␣
↪does."
  (unless (eq (org-capture-get :type 'local) 'entry)
    (error
     "Refiling from a capture buffer makes only sense for `entry'-type templates"))
  (let ((pos (point))
    (base (buffer-base-buffer (current-buffer)))
    (org-capture-is-refiling t)
    (kill-buffer (org-capture-get :kill-buffer 'local)))
    (org-capture-put :kill-buffer nil)
    (org-capture-finalize)
    (save-window-excursion
      (with-current-buffer (or base (current-buffer))
    (org-with-wide-buffer
     (goto-char pos)
     (my/refile file headline arg))))
    (when kill-buffer (kill-buffer base))))
```

(continues on next page)

(continued from previous page)

```
(defmacro josh/make-org-refile-hydra (hydraname file keyandheadline)
  "Make a hydra named HYDRANAME with refile targets to FILE.
KEYANDHEADLINE should be a list of cons cells of the form (\"key\" . \"headline\")"
  `(defhydra ,hydraname (:color blue :after-exit (unless (or hydra-deactivate
                                                             current-prefix-arg) ;If we
↪'re just jumping to a location, quit the hydra
                                                         (josh/org-refile-hydra/body)))
     ,file
     ,@(cl-loop for kv in keyandheadline
            collect (list (car kv) (list 'josh/refile file (cdr kv) 'current-prefix-
↪arg) (cdr kv)))
     ("q" nil "cancel")))

;;;;;;;;;;
;; Here we'll define our refile headlines
;;;;;;;;;;

(josh/make-org-refile-hydra josh/org-refile-hydra-file-ds
                        "~/my_org/datascience.org"
                        (("1" . "@Datascience @Inbox")
                         ("2" . "@Datascience @Notes")))

(josh/make-org-refile-hydra josh/org-refile-hydra-file-bgr
                        "~/my_org/bgr.org"
                        (("1" . "#BGR #Inbox")
                         ("2" . "#questions @ BGR")
                             ("3" . "Inventory management Project")))

(josh/make-org-refile-hydra josh/org-refile-hydra-file-todoglobal
                        "todo-global.org"
                        (("1" . ";Emacs Stuff")
                         ("2" . ";someday")))

(defhydra josh/org-refile-hydra (:foreign-keys run)
  "Refile"
  ("a" josh/org-refile-hydra-file-ds/body "File A" :exit t)
  ("b" josh/org-refile-hydra-file-bgr/body "File B" :exit t)
  ("c" josh/org-refile-hydra-file-todoglobal/body "File C" :exit t)
  ("j" org-refile-goto-last-stored "Jump to last refile" :exit t)
  ("q" nil "cancel"))

(global-set-key (kbd "<f8> r") 'josh/org-refile-hydra/body)
```

Source : Hydra documentation

```
;;  Hydras for window configuration. Using the deluxe
(defhydra hydra-window ()
  "
Movement^^        ^Split^         ^Switch^          ^Resize^
----------------------------------------------------------------
_h_ ←               _v_ertical      _b_uffer                _q_ X←
_j_ ↓               _x_ horizontal  _f_ind files    _w_ X↓
_k_ ↑               _z_ undo        _a_ce 1         _e_ X↑
_l_ →               _Z_ reset       _s_wap          _r_ X→
_F_ollow            _D_lt Other     _S_ave          max_i_mize
_SPC_ cancel        _o_nly this     _d_elete
"
```

(continues on next page)

(continued from previous page)

```
  ("h" windmove-left )
  ("j" windmove-down )
  ("k" windmove-up )
  ("l" windmove-right )
  ("q" hydra-move-splitter-left)
  ("w" hydra-move-splitter-down)
  ("e" hydra-move-splitter-up)
  ("r" hydra-move-splitter-right)
  ("b" helm-mini)
  ("f" helm-find-files)
  ("F" follow-mode)
  ("a" (lambda ()
         (interactive)
         (ace-window 1)
         (add-hook 'ace-window-end-once-hook
                   'hydra-window/body))
   )
  ("v" (lambda ()
         (interactive)
         (split-window-right)
         (windmove-right))
   )
  ("x" (lambda ()
         (interactive)
         (split-window-below)
         (windmove-down))
   )
  ("s" (lambda ()
         (interactive)
         (ace-window 4)
         (add-hook 'ace-window-end-once-hook
                   'hydra-window/body)))
  ("S" save-buffer)
  ("d" delete-window)
  ("D" (lambda ()
         (interactive)
         (ace-window 16)
         (add-hook 'ace-window-end-once-hook
                   'hydra-window/body))
   )
  ("o" delete-other-windows)
  ("i" ace-maximize-window)
  ("z" (progn
         (winner-undo)
         (setq this-command 'winner-undo))
   )
  ("Z" winner-redo)
  ("SPC" nil)
  )

(global-set-key (kbd "<f8> w") 'hydra-window/body)
```

Reference: https://emacs.stackexchange.com/questions/44128/function-to-do-helm-do-ag-for-a-specific-project

4.19.1 4.19.3.1 In project directory

```
(defun helm-do-ag-projects ()
  "Grep string in Project directory" (interactive)
  (let ((rootdir (concat "~/my_projects/")))
    (let ((helm-ag-command-option (concat helm-ag-command-option "")))
      (helm-do-ag rootdir))))
```

4.19.2 4.19.3.2 Scimax config directory

```
(defun helm-do-ag-emacs-config ()
  "Grep string in Emacs custom code"
  (interactive)
  (let ((rootdir (concat "~/scimax/user/sr-cust/")))
    (let ((helm-ag-command-option (concat helm-ag-command-option "")))
      (helm-do-ag rootdir))))
```

4.19.3 4.19.3.3 Journal directory

```
(defun helm-do-ag-journal ()
  "Grep string in journal directory"
  (interactive)
  (let ((specfile (concat "~/my_org/journal/")))
    (let ((helm-ag-command-option (concat helm-ag-command-option "")))
      (helm-ag-this-file specfile))))
```

4.19.4 4.19.3.4 BGR file

```
(defun helm-do-ag-bgr ()
  "Grep string in BGR file"
  (interactive)
  (let ((specfile (concat "~/my_org/bgr.org")))
    (let ((helm-ag-command-option (concat helm-ag-command-option "")))
      (helm-do-ag-this-file specfile))))
```

4.19.5 4.19.3.5 Defining hydra

```
(defhydra shrysr/hydra-helm-ag-do-menu ()
  "
Helm-do-ag in specified locations
^location^  ^command^
-------------------------------------------------------------
e:        emacs custom config
b:        bgr file
o:        org files
j:        journal search
"
  ("e" helm-do-ag-emacs-config)
  ("j" helm-do-ag-journal :color blue)
```

(continues on next page)

(continued from previous page)

```
  ("p" helm-do-ag-projects)
  ("o" helm-do-ag-org)
  ("q" quit-window "quit" :color red))

(global-set-key (kbd "<f8> h") 'shrysr/hydra-helm-ag-do-menu/body)
```

4.19.6 4.19.4.1 Scimax - magit and windows

```
;; scimax directory magit status
(defun sr/windows-magit-scimax ()
  (interactive)
  (ace-delete-other-windows)
  (dired "~/scimax/user/")
  (switch-window-then-split-right nil)
  (magit-status "~/scimax/")
  (switch-window)
  (split-window-vertically)
  (dired-up-directory)
  (windmove-right)
  )
```

4.19.7 4.19.4.2 Org files - magit and windows

```
;; my_org magit status
(defun sr/windows-magit-org ()
  (interactive)
  (ace-delete-other-windows)
  (magit-status "~/my_org/")
  )
```

4.19.8 4.19.4.3 Project directory - magit and windows

```
;; magit status
(defun sr/windows-magit-projects ()
  (interactive)
  (ace-delete-other-windows)
  (switch-window-then-split-right nil)
  (magit-status "~/my_projects/")
  (switch-window)
  (dired "~/my_projects/")
  (switch-window)
  )
```

4.19.9 4.19.4.4 TODO Project: Switch and windows

- Note taken on [2019-02-10 Sun 07:09] Experiment with helm-swoop functions to target only top level headings

```
(defun sr/windows-projects ()
  (interactive)
  (ace-delete-other-windows)
  (switch-window-then-split-right nil)
  (projectile-switch-project)
  (switch-window)
  (find-file "~/my_org/project-tasks.org")
  (widen)
  (helm-org-rifle-current-buffer)
  (org-narrow-to-subtree)
  (outline-show-children)
  )
```

4.19.10 4.19.4.5 Defining Hydra

```
(defhydra sr/process-window-keys ()
  "
Key^^    ^Workflow^
--------------------
o        org magit
s        scimax magit
p        projects magit
w        select project and set window config
SPC      exit
"
  ("o" sr/windows-magit-org )
  ("p" sr/windows-magit-projects )
  ("s" sr/windows-magit-scimax )
  ("w" sr/windows-projects)
  ("SPC" nil)
  )

(global-set-key (kbd "<f8> m") 'sr/process-window-keys/body)
```

```
(message "Loaded Hydras")
```

4.20 4.20 Python [0/4]

I have tried using the conda.el package, but for some reason, it will not recognise the virtual environments in the paths specified. An excellent guide to using virtual environments with python is available on Rakan's blog, and I've borrowed some bits of it.

I prefer to use miniconda for my python programming, and therefore the environment path is set appropriately below.

Switching an environment is as simple as `M-x pyenv-workon`. This is quite excellent.

```
(use-package pyenv-mode
  :init
  ;;(add-to-list 'exec-path "~/.pyenv/shims")
  (setenv "WORKON_HOME" "~/miniconda3/envs/")
  :config
  (pyenv-mode)
;; :bind
```

(continues on next page)

(continued from previous page)

```
;; ("C-x p e" . pyenv-activate-current-project)
  )
```

```
(use-package conda
  :ensure t
  :config
  (require 'conda)
  ;;(conda-env-initialize-interactive-shells)

  ;;  Yes I want conda active for eshell as well
  ;;(conda-env-initialize-eshell)

  ;; Setting home path for  miniconda. I use the same path everywhee.
  (custom-set-variables
   '(conda-anaconda-home "~/miniconda3/"))
  )
```

- Note taken on [2019-02-12 Tue 14:52] This is to take care of the annoying indentation message that always pops up.

```
(setq python-indent-guess-indent-offset nil)
```

```
(add-to-list 'company-backends 'company-ob-ipython)
(company-mode)
```

- Note taken on [2019-02-12 Tue 14:48] Since I am more familiar with ob-ipython and there are a bunch of interesting features already implemented in it like the automatic setting of a kernel and file names for graphic outputs and so on - I will explore jupyter-emacs at a later date.

```
(use-package jupyter
  :ensure t
  :defer t
  :config
  ;(org-babel-load-languages '(jupyter .t))
  (setq org-babel-default-header-args:jupyter-python '((:async . "yes")
                                                       (:session . "jipython")
                                                       (:kernel . "python3")))
  )
```

CHAPTER 5

5 Disabled

These are packages and snippets that disabled from tangling and execution. Some of them have been explored and discarded for various reasons. The rest are packages / features I could not get working satisfactorily.

5.1 5.1 TODO org2blog : publishing to wordpress [0/1]

- Note taken on [2019-07-21 Sun 16:17] I have a suspicion that the mp-wp implementation is different from the usual wordpress installation and therefore, the org2blog function may not work with it.

Fueled by discussions in #ossasepia, I have decided to shift direction towards Wordpress and a custom VPS server to host and secure my own data.

While there are disadvantages in using a behemoth like wordpress over a simple static hugo site - one key driver is the ability to create comment systems with pingbacks, and control the data. Well to the extent possible on a rented VPS server, which is still more control than using services like Bitbucket / Github.

Why bother with comment systems? I've realised from my discussions with diana_coman that despite the inconvenience, the whole point of the web is sharing and hopefully creating a seed for intellectual discussions that could lead somewhere meaningful. A good commenting system is obviously needed, for the little that takes place.

Eventually, it would be interesting to run V to host my own repository and files, and perhaps employing git annex, for managing an index of other files in different locations.

The good thing that I've become more comfortable with Emacs and getting packages to work, and even exploring the source code to understand the arguments better.

```
This was intimidating in the past, but reading the source code may be the best way to␣
↪understand a language.
```

Samuel Zhao's post is a good introduction to setting up org2blog, and I have replicated the steps with some additional twists:

1. Encryption of the login details with my personal key, and setup to ingest a list of authsources.
2. Setting up the org2blog package to enable publishing and control of blog posts right within Emacs.

3. It would be nice to have a link to the source file in each wordpress post. Perhaps the file could be signed with gpg as well?

```
(use-package org2blog
  :ensure t
  :defer nil
  :config
  (require 'xml-rpc)
  (require 'org2blog-autoloads)
  ;; (require 'netrc)
  (setq org2blog/wp-blog-alist
        `(("young-hands-club"
           :url "http://younghands.club/xmlrc.php"
           :default-title ""
           :default-categories ("daily-log")
           :username ,(auth-source-user-or-password 'young-hands-club "login")
           :password ,(auth-source-user-or-password young-hands-club "password")
           :tags-as-categories true)))
  (setq wpcredentials (auth-source-user-and-password "young-hands-club"))
  (setq org2blog/wp-blog-alist
        `(("young-hands-club"
           :url "http://younghands.club/xmlrc.php"
           :username ,(car wpcredentials)
           :password ,(cadr wpcredentials))))

  ;; implemented as HTML styling. Your pick!
  (setq org2blog/wp-use-sourcecode-shortcode 't)

  ;; removed light="true"
  (setq org2blog/wp-sourcecode-default-params nil)

  ;; target language needs to be in here
  (setq org2blog/wp-sourcecode-langs
        '("actionscript3" "bash" "coldfusion" "cpp" "csharp" "css" "delphi"
          "erlang" "fsharp" "diff" "groovy" "html" "javascript" "java" "javafx"
↪"matlab"
          "objc" "perl" "php" "text" "powershell" "python" "ruby" "scala" "sql"
          "vb" "xml"
          "sh" "emacs-lisp" "lisp" "lua" "R"))

  (setq org-src-fontify-natively t)

  ;; ;; You want set the keymode map you can use these.
  ;; (global-set-key (kbd " l") 'org2blog/wp-login)
  ;; (global-set-key (kbd " p") 'org2blog/wp-post-buffer-and-publish)
  ;; )))
  )
```

5.2 5.2 TEST helm-ext

- Note taken on [2019-04-29 Mon 08:01] Disabling excecution for the time being.

```
Extensions to helm, which I find useful but are unlikely to be accepted in the␣
↪upstream. A collection of dirty hacks for helm!

`https://github.com/cute-jumper/helm-ext <https://github.com/cute-jumper/helm-ext>`_
```

```
(use-package helm-ext
  :ensure t
  :config
  (helm-ext-ff-enable-skipping-dots t)
  ;; Testing the auto path expansion
  ;;(helm-ff-ext-enable-auto-path-expansion t)
  )
```

5.3 5.3 Scimax customisations

- Note taken on [2019-10-07 Mon 13:17] The default config is insufficient and in particular, uses the ess-smart-underscore package which is not useful to my general workflow.

```
(require 'scimax-statistics)
```

5.4 5.4 Dired

Source: https://github.com/angrybacon/dotemacs/blob/master/dotemacs.org

```
(use-package dired
  :ensure nil
  :delight dired-mode "Dired"
  :preface
  (defun me/dired-directories-first ()
    "Sort dired listings with directories first before adding marks."
    (save-excursion
      (let (buffer-read-only)
        (forward-line 2)
        (sort-regexp-fields t "^.*$" "[ ]*." (point) (point-max)))
      (set-buffer-modified-p nil)))
  ;:hook ;(dired-mode . dired-hide-details-mode)
  :config
  (advice-add 'dired-readin :after #'me/dired-directories-first)
  (setq-default
   dired-auto-revert-buffer t
   dired-dwim-target t
   dired-hide-details-hide-symlink-targets nil
   dired-listing-switches "-alh"
   dired-ls-F-marks-symlinks nil
   dired-recursive-copies 'always))

(use-package dired-x
  :ensure nil
  :preface
  (defun me/dired-revert-after-command (command &optional output error)
    (revert-buffer))
  :config
  (advice-add 'dired-smart-shell-command :after #'me/dired-revert-after-command))

(message "Loaded Dired customisation")
```

5.5 5.5 TEST Activating windmove to facilitate Hydras

Super would actually be a good option. However, this interferes with default configurations in MS Windows, especially while using virtualbox. Using Meta for now.

```
(windmove-default-keybindings 'meta)
```

5.6 5.6 TEST Export async

- Note taken on [2019-02-14 Thu 16:03] This requires a separate init file to be setup that enables Emacs to launch a separate process to export large files. It would be better as a vanilla emacs file.

```
(setq org-export-async-init-file
      (expand-file-name "async-export.el" user-emacs-directory)
      )
```

5.7 5.7 TEST Ob-async

- Note taken on [2019-02-14 Thu 16:02] This should enable evaluating code in org babel source blocks asynchronously. The header in the source block should have the async enabled.

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

5.8 5.8 TEST Auto saving all org files by the hour

- Note taken on [2019-07-05 Fri 11:49] On the mac, this seems to be saving for each house since the time specified ? This behavior needs to be checked out.

This is adopted from Bernt Hansen's configuration. Essentially, all the org buffers are saved 1 minute before the hour, every hour.

```
(run-at-time "00:59" 3600 'org-save-all-org-buffers)
```

5.9 5.9 TEST Tags setup

Borrowing Kaushal modi's setup for tags. I will start with using gtags and expand later to ctags.

```
;;; gtags, GNU global

(when (executable-find "global")
;;;; ggtags
  ;; https://github.com/leoliu/ggtags
  (use-package ggtags
    :config
    (progn
```

(continues on next page)

(continued from previous page)

```
      (setq ggtags-update-on-save nil) ;Don't try to update GTAGS on each save; makes␣
↪the system sluggish for huge projects.
      (setq ggtags-highlight-tag nil)  ;Don't auto-highlight tag at point.. makes the␣
↪system really sluggish!
      (setq ggtags-sort-by-nearness nil) ; Enabling nearness requires global 6.5+
      (setq ggtags-navigation-mode-lighter nil)
      (setq ggtags-mode-line-project-name nil)
      (setq ggtags-oversize-limit (* 30 1024 1024)) ; 30 MB

      (dolist (hook '(verilog-mode-hook
                      c-mode-hook))
        (add-hook hook #'ggtags-mode))

      ;; Don't consider ` (back quote) as part of `tag' when looking for a
      ;; Verilog macro definition
      (defun ggtags-tag-at-point ()
        (pcase (funcall ggtags-bounds-of-tag-function)
          (`(,beg . ,end)
           (if (eq ?` (string-to-char (buffer-substring beg end)))
               ;; If `(buffer-substring beg end)' returns "`uvm_info" (for example),
               ;; discard the ` and return just "uvm_info"
               (buffer-substring (1+ beg) end)
             ;; else return the whole `(buffer-substring beg end)'
             (buffer-substring beg end)))))

      ;; Remove the default binding for `M-.' in `ggtags-mode-map'
      (bind-key "M-." nil ggtags-mode-map)
      ;; Remove the default binding for `M-o' in `ggtags-navigation-map'
      (bind-key "M-o" nil ggtags-navigation-map)

      (key-chord-define-global "??" #'ggtags-show-definition))))
```

5.10 5.10 TEST Icicles

- Note taken on [2019-02-28 Thu 16:01] The default key bindings of icicles changes the org source block edit shortcut. However, the package appears very interesting so far, if not a bit slow to respond. Switching over to icicles will need some research for making sure none of the existing keybindings and workflows are crippled. This package cannot be installed via Melpa. The easiest method appears to be to download the files as a zip folder from the icicle git repository. The automatic install script draws files from the Emacs wiki, which at times may be down. As such icicles can be switched off by using `M-x icy-mode`.

```
(load "~/scimax/user/external_packages/icicles-install.el")
(setq icicle-download-dir "~/scimax/user/external_packages/icicle_packages/")
(add-to-list 'load-path "~/scimax/user/external_packages/icicle_packages/")
(require 'icicles)
(icy-mode 1)
```

5.11 5.11 erc

- Note taken on [2019-10-23 Wed 09:15] I'm currently using weechat over tmux, but there are great conveniences in using erc. It is possible then to capture conversation and make notes more easily.

Base config setting the channels that I would like frequent. I have actually had enjoyable experiences chatting with the Emacs and coding whizzes in the Emacs channel.

```
(use-package erc
  :config
  (setq erc-hide-list '("PART" "QUIT" "JOIN"))
  (setq erc-autojoin-channels-alist '(("freenode.net"
                                       "#org-mode"
                                       "#emacs"
                                       "#emacs-beginners"
                                       "#docker"))
        erc-server "irc.freenode.net"
        erc-nick "shrysr")
  (setq erc-fill-static-center t)

  )
```

Loading ERC without password (temporarily). The password has to be set in the .authconfig file and encrypted.

```
(erc
  :server "irc.freenode.net"
  :port 6667
  :nick "shrysr")
```

5.12 5.12 Scheme setup

- Note taken on [2019-10-23 Wed 09:21] I used this briefly during a foray into the SICP course.
- References
  - http://praveen.kumar.in/2011/03/06/gnu-emacs-and-mit-scheme-on-mac-os-x/

```
(setq scheme-program-name "/Applications/MIT-GNU-Scheme.app/Contents/Resources/mit-
↪scheme")
(require 'xscheme)

(message "Loaded scheme setup")
```

5.13 5.13 TODO lintr

- Note taken on [2019-02-11 Mon 07:21] It appears there is no package called lintr. This needs further investigation.

This package is deemed necessary to enable flymake in ESS. Without it, there is significantly more lag while the suggestions / corrections are generated in ESS modes.

```
(use-package lintr
  :ensure nil
)
```

5.14 5.14 Better defaults

- Note taken on [2019-08-28 Wed 13:45] Disabling this package until it is explored better.

I need to explore the changed made by this package. For now, it is loaded right in the beginning so that it does not overwrite other customisations down the line.

```
(use-package better-defaults
  :ensure t
)

(message "Loaded better-defaults package")
```

5.15 5.15 Elfeed customisation

- Note taken on [2019-09-25 Wed 14:09] Scimax's elfeed is enabled, along with elfeed-org and elfeeed-goodies
- Note taken on [2019-07-08 Mon 08:10] Disabling elfeed for now.

Source: http://heikkil.github.io/blog/2015/05/09/notes-from-elfeed-entries/

```
;; Elfeed configuration source :
(use-package elfeed
  :bind (:map elfeed-search-mode-map
              ("A" . bjm/elfeed-show-all)
              ("E" . bjm/elfeed-show-emacs)
              ("D" . bjm/elfeed-show-daily)
              ("q" . bjm/elfeed-save-db-and-bury))
  :init
  (setq my/default-elfeed-search-filter "@1-month-ago +unread !sport ")
  (setq-default elfeed-search-filter my/default-elfeed-search-filter)
  (setq elfeed-db-direcory "~/scimax/user/elfeeddb")
  :config
  (elfeed-org)
  (elfeed-goodies/setup)
  (setq elfeed-use-curl t)

  ;;
  ;; linking and capturing
  ;;

  (defun elfeed-link-title (entry)
    "Copy the entry title and URL as org link to the clipboard."
    (interactive)
    (let* ((link (elfeed-entry-link entry))
           (title (elfeed-entry-title entry))
           (titlelink (concat "[[" link "][" title "]]")))
      (when titlelink
        (kill-new titlelink)
        (x-set-selection 'PRIMARY titlelink)
        (message "Yanked: %s" titlelink))))

  ;; show mode

  (defun elfeed-show-link-title ()
    "Copy the current entry title and URL as org link to the clipboard."
```

(continues on next page)

(continued from previous page)

```
    (interactive)
    (elfeed-link-title elfeed-show-entry))

  (defun elfeed-show-quick-url-note ()
    "Fastest way to capture entry link to org agenda from elfeed show mode"
    (interactive)
    (elfeed-link-title elfeed-show-entry)
    (org-capture nil "n")
    (yank)
    (org-capture-finalize))

  (bind-keys :map elfeed-show-mode-map
             ("l" . elfeed-show-link-title)
             ("v" . elfeed-show-quick-url-note))

  ;; search mode

  (defun elfeed-search-link-title ()
    "Copy the current entry title and URL as org link to the clipboard."
    (interactive)
    (let ((entries (elfeed-search-selected)))
      (cl-loop for entry in entries
               when (elfeed-entry-link entry)
               do (elfeed-link-title entry))))

  (defun elfeed-search-quick-url-note ()
    "In search mode, capture the title and link for the selected
     entry or entries in org aganda."
    (interactive)
    (let ((entries (elfeed-search-selected)))
      (cl-loop for entry in entries
               do (elfeed-untag entry 'unread)
               when (elfeed-entry-link entry)
               do (elfeed-link-title entry)
               do (org-capture nil "n")
               do (yank)
               do (org-capture-finalize)
                (mapc #'elfeed-search-update-entry entries))
      (unless (use-region-p) (forward-line))))

  (bind-keys :map elfeed-search-mode-map
             ("l" . elfeed-search-link-title)
             ("v" . elfeed-search-quick-url-note))

   ;;functions to support syncing .elfeed between machines
  ;;makes sure elfeed reads index from disk before launching
  (defun bjm/elfeed-load-db-and-open ()
    "Wrapper to load the elfeed db from disk before opening"
    (interactive)
    (elfeed-db-load)
    (elfeed)
    (elfeed-search-update--force))

  ;;write to disk when quiting
  (defun bjm/elfeed-save-db-and-bury ()
    "Wrapper to save the elfeed db to disk before burying buffer"
    (interactive)
```

(continues on next page)

(continued from previous page)

```
    (elfeed-db-save)
    (quit-window))
  )
```

Using an org source is the easiest way to organise my RSS feeds for reading with Elfeed.

```
(require 'scimax-elfeed)

;; use an org file to organise feeds
(use-package elfeed-org
  :ensure t
  :config
  (setq rmh-elfeed-org-files (list "~/my_org/elfeed.org"))
  )

(use-package elfeed-goodies
  :ensure t
  :init
  (elfeed-goodies/setup)
  )
```

- Note taken on [2019-02-17 Sun 18:11] This will need an export to a source org file per the settings.

```
(message "Loaded Elfeed customisations")
```

5.16 5.16 ediff

I have to diff between org files pretty often, and need the headings to be unfolded.

Source: http://emacs.stackexchange.com/questions/21335/prevent-folding-org-files-opened-by-ediff

```
;; Check for org mode and existence of buffer
(defun f-ediff-org-showhide (buf command &rest cmdargs)
  "If buffer exists and is orgmode then execute command"
  (when buf
    (when (eq (buffer-local-value 'major-mode (get-buffer buf)) 'org-mode)
      (save-excursion (set-buffer buf) (apply command cmdargs)))))

(defun f-ediff-org-unfold-tree-element ()
  "Unfold tree at diff location"
  (f-ediff-org-showhide ediff-buffer-A 'org-reveal)
  (f-ediff-org-showhide ediff-buffer-B 'org-reveal)
  (f-ediff-org-showhide ediff-buffer-C 'org-reveal))

(defun f-ediff-org-fold-tree ()
  "Fold tree back to top level"
  (f-ediff-org-showhide ediff-buffer-A 'hide-sublevels 1)
  (f-ediff-org-showhide ediff-buffer-B 'hide-sublevels 1)
  (f-ediff-org-showhide ediff-buffer-C 'hide-sublevels 1))

(add-hook 'ediff-select-hook 'f-ediff-org-unfold-tree-element)
(add-hook 'ediff-unselect-hook 'f-ediff-org-fold-tree)
```

5.17 5.17 Spell Checking

- Note taken on [2019-02-09 Sat 11:51] disabling flycheck for the moment and enabling flymake

Source: https://writequit.org/org/

Basic config

```
(use-package flycheck
  :defer 5
  :bind (("M-g M-n" . flycheck-next-error)
         ("M-g M-p" . flycheck-previous-error)
         ("M-g M-=" . flycheck-list-errors))
  :init (global-flycheck-mode)
  :diminish flycheck-mode
  :config
  (progn
    (setq-default flycheck-disabled-checkers '(emacs-lisp-checkdoc json-jsonlint json-
↪python-json ess iess))
    (use-package flycheck-pos-tip
      :init (flycheck-pos-tip-mode))
    (use-package helm-flycheck
      :init (define-key flycheck-mode-map (kbd "C-c ! h") 'helm-flycheck))
    (use-package flycheck-haskell
      :init (add-hook 'flycheck-mode-hook #'flycheck-haskell-setup))))
```

Reference: https://alhassy.github.io/init/

Org mode is derived from text mode, therefore it is sufficient to activate for text mode.

```
(use-package flyspell
  :hook (
           (prog-mode . flyspell-prog-mode)
           (text-mode . flyspell-mode))
)
```

This is especially for python modules at the moment.

```
(when (require 'flycheck nil t)
  (setq elpy-modules (delq 'elpy-module-flymake elpy-modules))
  (add-hook 'elpy-mode-hook 'flycheck-mode))
```

- Note taken on [2019-07-12 Fri 20:22] So far this is working out rather well, and as expected.

Facing trouble enabling flyspell in the mac. This seems to be a solution, as outlined in this SO discussion.

```
(if (system-type-is-darwin)
    (setq ispell-program-name "/usr/local/Cellar/aspell/0.60.6.1_1/bin/aspell")
  )
```