Emacs configuration

Bas Chatel

September 30, 2020

Contents

1 File layout and paths

Here we provide important directories and files. For example, I use Dropbox to synchronise all my non-programming files and github for my programming projects. Furthermore the configuration files are written here for easy access in the z-map that is described later on.

- Dotfiles
- Github

- Programming folders
- Dropbox

- Org-roam directory

- Ideas

- $\text{Knowledge}_{\text{base}}$

- Org-journal

- $\text{Papers}_{\text{andarticles}}$

- Personal

- Refs

- Snippets

- Work

- Bibliography

- PhD

```
;; General directories
(setq emacs-dir "~/.emacs_test.d/")
(setq dropbox-dir "~/Dropbox/")
(setq github-dir "~/github/")

;; Configuration files
```

```
(setq emacs-config-org-file (concat emacs-dir "config.org"))
(setq zshrc-file "~/.zshrc")
(setq index-org-file (concat dropbox-dir "orgfiles/index.org"))
(setq skhdrc-file "~/.skhdrc")
(setq qmk-keymap-file "~/qmk_firmware/keyboards/keebio/iris/keymaps/popoiopo/keymap.c")
(setq yabai-file "~/.yabairc")
(setq qutebrowser-file "~/.qutebrowser/qutemacs.py")
(setq references-bib-file (concat dropbox-dir "bibliography/references.bib"))

;; More specific files
(setq amx-items (concat emacs-dir "amx-items"))
(setq org-reveal-root "http://cdn.jsdelivr.net/reveal.js/3.0.0/")
(setq roamnotes-path (concat dropbox-dir "RoamNotes/"))
(setq roam-db-path "~/org-roam.db")
(setq org-journal-path (concat roamnotes-path "org-journal/"))

;; DOEN HET NOG NIET
(setq backup-per-save (concat emacs-dir "backup/per-save"))
(setq backup-per-session (concat emacs-dir "backup/per-session"))
(setq libre-office-path "/Applications/LibreOffice.app/Contents/MacOS/soffice")
```

2 Configuration

2.1 Autocompletions

2.1.1 Company (autocompletion framework)

Company is a text completion framework for Emacs. The name stands for "complete anything". It uses pluggable back-ends and front-ends to retrieve and display completion candidates. See documentation on this site.

```
(use-package company
  :ensure t
  :init
  (add-hook 'after-init-hook 'global-company-mode))
```

2.1.2 Electric pairing

Automatically pair the following elements on autocorrect. This is enabled in the Standards (minor modes and minor improvements) through electric-pair-mode.

```
(setq electric-pair-pairs '(
    (?\( . ?\))
    (?\[ . ?\])
    (?\" . ?\")
    (?\{ . ?\})
    (?\< . ?\>)
    ))
```

2.1.3 IDO autocomplete filename searches

Ivy takes care of autocompleting filenames in the minibuffer at the bottom of the screen. It also keeps typing paths to a minimum as a folder is just an enter away.

```
(ivy-mode 1)
(setq ivy-use-virtual-buffers t)
(setq enable-recursive-minibuffers t)
(global-set-key "\C-s" 'swiper)
```

Makes sure that M-x also generates suggestions. Otherwise you'd have to remember everything and not get autocompleted in M-x functions. Amx is the newer version of smex.

```
(use-package amx
  :ensure t
  :after ivy
  :custom
  (amx-backend 'auto)
  (ams-save-file amx-items)
  (amx-history-length 50)
  (amx-show-key-bindings nil)
  :config
  (amx-mode 1))
```

2.1.4 Yassnippet

Yassnippet is the templating system that is used. It creates a folder called snippets in which you can make a folder for each major mode you'd want a template for. E.g., python can have a few snippets to prettyfie a matplotlib graph, or org can have a template for exporting to a latex article or an html webpage.

```
(use-package yasnippet
  :ensure t
  :config (use-package yasnippet-snippets
    :ensure t)
  (yas-reload-all))
(yas-global-mode 1)
```

2.2 Buffers

All things buffer related

2.2.1 Buffer-move

Be able to swap buffers. See Custom keystrokes for shortcuts (buf-move-xxx).

```
(use-package buffer-move
  :ensure t)
```

2.2.2 Ibuffer

Just a new buffer that lists the open buffers. It provides easy ways to close multiple buffers at once and navigate through them.

```
(global-set-key (kbd "C-x C-b") 'ibuffer)
(setq ibuffer-expert t)
```

2.2.3 Killing buffers

1. Always kill current buffer

   ```
   (global-set-key (kbd "C-x k") 'kill-current-buffer)
   ```

2. Kill all buffers

   ```
   (defun kill-all-buffers ()
     (interactive)
     (mapc 'kill-buffer (buffer-list)))
   (global-set-key (kbd "C-M-s-k") 'kill-all-buffers)
   ```

2.2.4 Narrowing

Function to easily narrow and widen an area of code. If you select a piece of text, call this function, it will create a buffer with just that in it. This makes searching, or exporting just a part of something much easier.

```
(defun narrow-or-widen-dwim (p)
  "Widen if buffer is narrowed, narrow-dwim otherwise.
Dwim means: region, org-src-block, org-subtree, or
defun, whichever applies first. Narrowing to
org-src-block actually calls ‘org-edit-src-code’.

With prefix P, don’t widen, just narrow even if buffer
is already narrowed."
  (interactive "P")
  (declare (interactive-only))
  (cond ((and (buffer-narrowed-p) (not p)) (widen))
((region-active-p)
 (narrow-to-region (region-beginning)
   (region-end)))
((derived-mode-p ’org-mode)
 ;; ‘org-edit-src-code’ is not a real narrowing
 ;; command. Remove this first conditional if
 ;; you don’t want it.
 (cond ((ignore-errors (org-edit-src-code) t)
(delete-other-windows))
       ((ignore-errors (org-narrow-to-block) t))
       (t (org-narrow-to-subtree))))
((derived-mode-p ’latex-mode)
 (LaTeX-narrow-to-environment))
(t (narrow-to-defun))))
```

2.2.5 Switch to previous buffer

Small function to switch to previously used buffer.

```
(defun er-switch-to-previous-buffer ()
  "Switch to previously open buffer.
   Repeated invocations toggle between the two most recently open buffers."
  (interactive)
  (switch-to-buffer (other-buffer (current-buffer))))
```

```
(global-set-key (kbd "C-c b") #'er-switch-to-previous-buffer)
```

2.2.6 switchwindow

Make switching buffer with C-x o easier. It provides you with shortcuts on the homerow to which buffer you want to go. Otherwise, you'd need to cycle through them which is aweful if you have multiple buffers on the screen.

```
(use-package switch-window
  :ensure t
  :config
  (setq switch-window-input-style 'minibuffer)
  (setq switch-window-increase 4)
  (setq switch-window-threshold 2)
  (setq switch-window-shortcut-style 'qwerty)
  (setq switch-window-qwerty-shortcuts
'("a" "s" "d" "f" "h" "j" "k" "l"))
  :bind
  ([remap other-window] . switch-window))
```

2.2.7 Toggle fullscreen buffer

When using multiple buffers at the same time, sometimes it's nice to toggle a single buffer as fullscreen.

```
(defun toggle-maximize-buffer () "Maximize buffer"
  (interactive)
  (if (= 1 (length (window-list)))
      (jump-to-register '_)
    (progn
      (window-configuration-to-register '_)
      (delete-other-windows))))
(global-set-key (kbd "C-M-f") 'toggle-maximize-buffer)
```

2.2.8 window splitting function

If you split the window into two buffers, follow the new buffer. You make a new one to work in there right?!

```
(defun split-and-follow-horizontally ()
  (interactive)
```

```
  (split-window-below)
  (balance-windows)
  (other-window 1))
(global-set-key (kbd "C-x 2") 'split-and-follow-horizontally)

(defun split-and-follow-vertically ()
  (interactive)
  (split-window-right)
  (balance-windows)
  (other-window 1))
(global-set-key (kbd "C-x 3") 'split-and-follow-vertically)
```

2.3 Custom functions

2.3.1 Backup files

Here we set where each file is backed up, how many versions of each file is backed

```
(setq version-control t     ;; Use version numbers for backups.
      kept-new-versions 10  ;; Number of newest versions to keep.
      kept-old-versions 0   ;; Number of oldest versions to keep.
      delete-old-versions t ;; Don't ask to delete excess backup versions.
      backup-by-copying t  ;; Copy all files, don't rename them.
      auto-save-interval 100 ;; Change interval of characters to which auto-save is ena
      )

(setq vc-make-backup-files t)

;; Default and per-save backups go here:
(setq backup-directory-alist '(("" . "~/.emacs_test.d/backup/per-save")))

(defun force-backup-of-buffer ()
  ;; Make a special "per session" backup at the first save of each
  ;; emacs session.
  (when (not buffer-backed-up)
    ;; Override the default parameters for per-session backups.
    (let ((backup-directory-alist '(("" . "~/.emacs_test.d/backup/per-session")))
  (kept-new-versions 3))
      (backup-buffer)))
  ;; Make a "per save" backup on each save.  The first save results in
```

```
  ;; both a per-session and a per-save backup, to keep the numbering
  ;; of per-save backups consistent.
  (let ((buffer-backed-up nil))
    (backup-buffer)))

(add-hook 'before-save-hook  'force-backup-of-buffer)
```

2.3.2 Edit and reload config

Small function to easily configure and reload the configuration file.

```
(defun config-visit ()
  (interactive)
  (find-file emacs-config-org-file))

(defun config-reload ()
  (interactive)
  (org-babel-load-file (expand-file-name emacs-config-org-file)))
```

2.4 Custom keystrokes

All (most) the custom key combinations that I use regularly.

```
;; set up my own map for files, folder and windows
(define-prefix-command 'z-map)
(global-set-key (kbd "C-z") 'z-map)
(define-key z-map (kbd "a") 'org-agenda-show-agenda-and-todo)
(define-key z-map (kbd "c") 'avy-goto-char)
(define-key z-map (kbd "n") 'narrow-or-widen-dwim)
(define-key z-map (kbd "t") 'toggle-transparency)
(define-key z-map (kbd "e") 'config-visit)
(define-key z-map (kbd "r") 'config-reload)
(define-key z-map (kbd "z") (defun zshrcEdit () (interactive) (find-file zshrc-file)))
(define-key z-map (kbd "i") (defun indexEdit() (interactive) (find-file index-org-file)
(define-key z-map (kbd "s") (defun skhdEdit() (interactive) (find-file skhdrc-file)))
(define-key z-map (kbd "k") (defun keyboardEdit() (interactive) (find-file qmk-keymap-f
(define-key z-map (kbd "y") (defun yabaiEdit() (interactive) (find-file yabai-file)))
(define-key z-map (kbd "q") (defun qutebrowserEdit() (interactive) (find-file qutebrows
(define-key z-map (kbd "b") (defun bibtexEdit() (interactive) (find-file references-bib
(define-key z-map (kbd "<left>") 'shrink-window-horizontally)
(define-key z-map (kbd "<right>") 'enlarge-window-horizontally)
```

```
(define-key z-map (kbd "<down>") 'shrink-window)
(define-key z-map (kbd "<up>") 'enlarge-window)
(define-key z-map (kbd "C-j") 'org-journal-new-entry)
(define-key z-map (kbd "C-t") 'org-journal-today)
(define-key z-map (kbd "C-<up>") 'buf-move-up)
(define-key z-map (kbd "C-<down>") 'buf-move-down)
(define-key z-map (kbd "C-<left>") 'buf-move-left)
(define-key z-map (kbd "C-<right>") 'buf-move-right)

;; ORG extra keybinding
;; Store a reference link to an org mode location
(global-set-key (kbd "C-c l") 'org-store-link)

;; Add an extra cursor above or below current cursor
(global-set-key (kbd "C-<") 'mc/mark-previous-like-this)
(global-set-key (kbd "C->") 'mc/mark-next-like-this)

;; Remove an extra cursor above or below current cursor
(global-set-key (kbd "C-,") 'mc/unmark-previous-like-this)
(global-set-key (kbd "C-.") 'mc/unmark-next-like-this)

;; Skip a spot in adding a new cursor above or below
(global-set-key (kbd "C-M-<") 'mc/skip-to-previous-like-this)
(global-set-key (kbd "C-M->") 'mc/skip-to-next-like-this)

;; Mark all entries in current selection (useful if you want to rename a variable in th
(global-set-key (kbd "C-M-,") 'mc/mark-all-like-this)

;; Create cursors on every line in selected area
(global-set-key (kbd "C-M-.") 'mc/edit-lines)

;; Insert numbers with increased index for exery cursor (useful for lists)
(global-set-key (kbd "C-;") 'mc/insert-numbers)

;; Same as numbers but then with letters
(global-set-key (kbd "C-M-;") 'mc/insert-letters)

;; With control shift and a mouse-click add cursor
(global-set-key (kbd "C-S-<mouse-1>") 'mc/add-cursor-on-click)
```

2.5 Exporting

2.5.1 Org to latex blank lines

Here we make a small adaption in exporting to latex file. A double newline is translated to a bigskip, thus creating an extra whitespace in the resulting pdf.

```
;; replace \n\n with bigskip
(defun my-replace-double-newline (backend)
  "replace multiple blank lines with bigskip"
  (interactive)
  (goto-char (point-min))
  (while (re-search-forward "\\(^\\s-*$\\)\n\n+" nil t)
    (replace-match "\n#+LATEX: \\par\\vspace{\\baselineskip}\\noindent\n" nil t)
    ;;(replace-match "\n#+LATEX: \\bigskip\\noindent\n" nil t)
    (forward-char 1)))

(add-hook 'org-export-before-processing-hook 'my-replace-double-newline)
```

2.5.2 Export to word

Make sure that export (C-e) to odt, will be formatted to a .doc document for word.

```
;; This setup is tested on Emacs 24.3 & Emacs 24.4 on Linux/OSX
;; org v7 bundled with Emacs 24.3
(setq org-export-odt-preferred-output-format "doc")
;; org v8 bundled with Emacs 24.4
(setq org-odt-preferred-output-format "doc")
;; BTW, you can assign "pdf" in above variables if you prefer PDF format

;; Only OSX need below setup
(defun my-setup-odt-org-convert-process ()
  (setq process-string "/Applications/LibreOffice.app/Contents/MacOS/soffice --headless
  (interactive)
  (let ((cmd libre-office-path))
    (when (and (eq system-type 'darwin) (file-exists-p cmd))
      ;; org v7
      (setq org-export-odt-convert-processes '(("LibreOffice" "/Applications/LibreOffi
      ;; org v8
      (setq org-odt-convert-processes '(("LibreOffice"  "/Applications/LibreOffice.app/
```

```
    ))
(my-setup-odt-org-convert-process)
```

2.5.3 Reveal.js

Provide the option to export (C-e) an org-file to a reveal presentation.

```
(use-package ox-reveal
:ensure ox-reveal)
(setq org-reveal-mathjax t)
(use-package htmlize :ensure t)
```

2.5.4 Export to subdirectory

```
(defun org-export-output-file-name-modified (orig-fun extension &optional subtreep pub-
  (unless pub-dir
    (setq pub-dir "exported-org-files")
    (unless (file-directory-p pub-dir)
      (make-directory pub-dir)))
  (apply orig-fun extension subtreep pub-dir nil))
(advice-add 'org-export-output-file-name :around #'org-export-output-file-name-modifie
```

2.6 Gimmicks

Just some small functions that can be used for (almost) useless things.

2.6.1 Transparency

```
;;(set-frame-parameter (selected-frame) 'alpha '(<active> . <inactive>))
;;(set-frame-parameter (selected-frame) 'alpha <both>)
(set-frame-parameter (selected-frame) 'alpha '(100 . 100))
(add-to-list 'default-frame-alist '(alpha . (100 . 100)))

(defun toggle-transparency ()
   (interactive)
   (let ((alpha (frame-parameter nil 'alpha)))
     (set-frame-parameter
      nil 'alpha
      (if (eql (cond ((numberp alpha) alpha)
     ((numberp (cdr alpha)) (cdr alpha))
     ;; Also handle undocumented (<active> <inactive>) form.
```

```
    ((numberp (cadr alpha)) (cadr alpha)))
      100)
  '(95 . 95) '(100 . 100)))))
```

2.7 Navigation

2.7.1 avy

Avy is a powerful search package that lets you quickly navigate to wherever in your screen you want to go.

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

2.7.2 Multiple Cursors

Use multi cursor editing easily. For keybindings, see Custom keystrokes.

```
(require 'multiple-cursors)
```

2.8 Management - Knowledge

2.8.1 Citations

2.8.2 Org roam

One of the cornerstones of my knowledge management. This is based on Roam research or zettelkasten.

```
(use-package org-roam
      :ensure t
      :hook
      (after-init . org-roam-mode)
      :custom
      (org-roam-directory  roamnotes-path)
      (org-roam-db-location roam-db-path)
      :bind (:map org-roam-mode-map
      (("C-c n l" . org-roam)                          ;; Show backlinks in an extra bu
       ("C-c n f" . orb-find-non-ref-file)             ;; Find your notes easily throug
       ("C-c n g" . org-roam-graph-show))              ;; Show your knowledge-base in g
      :map org-mode-map
      (("C-c n i" . orb-insert-non-ref))               ;; Insert a link to a note
      (("C-c n I" . org-roam-insert-immediate))))      ;; Same as previous
```

```
(setq org-roam-capture-templates
      '(
;; Alle informatie met referenties naar waar ik het vandaan heb. Dit wordt het grootste
("k" "Knowledge base" plain (function org-roam--capture-get-point)
"%?"
:file-name "knowledge_base/%<%Y%m%d%H%M%S>-${slug}"
:head "#+title: ${title}\n\n- tags :: [[file:20200729175519-knowledge_base.org][Knowled
:unnarrowed t)

;; Hier staat alle informatie over mensen die ik ken; waar ik ze van ken, waar ze goed
("p" "Personal" plain (function org-roam--capture-get-point)
  "%?"
  :file-name "personal/%<%Y%m%d%H%M%S>-${slug}"
  :head "#+title: ${title}\n\n- tags ::  [[file:20200729175551-personal.org][personal]]
  :unnarrowed t)

;; Hier komen alle interessante ideeën die niet perse met literatuur versterkt worden,
("i" "Ideas" plain (function org-roam--capture-get-point)
  "%?"
  :file-name "ideas/%<%Y%m%d%H%M%S>-${slug}"
  :head "#+title: ${title}\n\n- tags :: [[file:20200729175615-ideas.org][Ideas]]\n\n* "
  :unnarrowed t)

;; Alle volledig uitgewerkte papers, blog posts, werken die ik doe (nog even nadenken o
("a" "Papers and Articles" plain (function org-roam--capture-get-point)
  "%?"
  :file-name "papers_and_articles/%<%Y%m%d%H%M%S>-${slug}"
  :head "#+title: ${title}\n\n- tags :: [[file:20200729175758-papers_and_articles.org]
  :unnarrowed t)

;; Alle volledig uitgewerkte papers, blog posts, werken die ik doe (nog even nadenken o
("w" "Work" plain (function org-roam--capture-get-point)
  "%?"
  :file-name "work/%<%Y%m%d%H%M%S>-${slug}"
  :head "#+title: ${title}\n\n- tags :: [[file:20200902142233-work.org][work]]\n\n* "
  :unnarrowed t)

;; Hier staan, labelled per programmeer taal en functie (optimization, plotting, etc.),
("s" "Snippets" plain (function org-roam--capture-get-point)
  "%?"
```

```
  :file-name "snippets/%<%Y%m%d%H%M%S>-${slug}"
  :head "#+title: ${title}\n\n- tags :: [[file:20200729175823-snippets.org][snippets]]\
  :unnarrowed t)
)
      )

;; On search for notes, prepend its respective directory name
(setq org-roam-tag-sources '(prop last-directory))
```

2.9 Management - Workflow/project

2.9.1 Tags

```
(setq org-tag-alist '(("@short" . ?s) ("@medium" . ?m) ("@long" . ?l) ("@very long" . ?
      ("@write" . ?w) ("@read" . ?r) ("@code" . ?c) ("@email" . ?e) ("@bellen" . ?b)
      ("@kopen" . ?k) ("@terugbetalen" . ?t) ("@gaan" . ?g)))
```

2.9.2 Org Journal

A big part of my workflow. In the beginning of each day I create a journal entry that uses org-journal to take all my current TODOs to the new day. The new entry is filled with org-journal-file-header-func as a template, carries over all elements that are defined in org-journal-carryover-items and puts them under the TODO header.

```
(use-package org-journal
  :ensure t
  :defer t
  :config
  (setq org-journal-dir org-journal-path
org-journal-enable-agenda-integration t
org-journal-date-prefix "#+TITLE: "
org-journal-file-format "%Y-%m-%d.org"
org-journal-date-format "%A, %d %B %Y"))

(setq org-journal-carryover-items "TODO=\"TODO\"|TODO=\"DOING\"|TODO=\"WAITING\"|TODO=\

(defun org-journal-file-header-func (time)
  "Custom function to create journal header."
  (concat
    (pcase org-journal-file-type
```

```
      (‘daily (concat "#+TITLE: " (format-time-string org-journal-date-format time) "\n

(setq org-journal-file-header ’org-journal-file-header-func)

(require ’org-journal)

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

(defun org-journal-save-entry-and-exit()
  "Simple convenience function.
  Saves the buffer of the current day’s entry and kills the window
  Similar to org-capture like behavior"
  (interactive)
  (save-buffer)
  (kill-buffer-and-window))
(define-key org-journal-mode-map (kbd "C-x C-s") ’org-journal-save-entry-and-exit)

(defun org-journal-today ()
    (interactive)
    (org-journal-new-entry t))
```

2.10 Org

2.10.1 Org-bullets

Small package that makes the org hierachy a little bit more appealing. The stars are changed into icons for example.

```
(use-package org-bullets
  :ensure t
  :config
  (add-hook ’org-mode-hook (lambda () (org-bullets-mode))))
```

2.10.2 Writing improvements

Some of the adjustments are **stolen** from this guy. It mainly revolves around using screenspace efficiently instead of randomly adding whitespace, or not displaying whitespace where there actually is whitespace.

```
(setq org-indent-indentation-per-level 1)            ;; Shorten the space on the left
(setq org-adapt-indentation nil)                     ;; Adapt indentation to outline n
(setq org-hide-emphasis-markers t)                   ;; When making something bold *Ha
(setq org-cycle-separator-lines 1)                   ;; Leave a single empty line betw
(setq org-hide-leading-stars 't)                     ;; Hide the extra stars in front
(customize-set-variable 'org-blank-before-new-entry
'((heading . nil)
  (plain-list-item . nil)))   ;; Dont randomly remove newlines below headers
```

2.10.3 Org Capture

Org capture makes creating a template from org a little easier. It creates a new file in which a function can be called and the template will be inserted.

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

(setq org-capture-templates '(("j" "Journal entry" entry (function org-journal-find-loc
       "* Day journal\n** %(format-time-string org-journal-time-format)%?")))
```

2.10.4 Image size

Make standard size for org images. Otherwise they can become gigantic!

```
(setq org-image-actual-width 600)
```

2.11 Standards (minor modes and minor improvements)

2.11.1 Alter annoying defaults

Small defaults to be changed as minor improvements. The changes are summarized next to it.

```
(setq save-interprogram-paste-before-kill t) ;; Perpetuates system clipboard
(setq scroll-conservatively 1)      ;; Keep from making huge jumps when scrolling
(setq ring-bell-function 'ignore)   ;; Unable annoying sounds
(setq visible-bell 1)               ;; disable annoying windows sound
```

```
(setq inhibit-startup-message t)    ;; Hide the startup message
(setq display-time-24hr-format t)   ;; Format clock
(setq-default display-line-numbers 'relative) ;; Setting the line numbers
(when window-system (global-hl-line-mode t)) ;; Get a current line shadow in IDE
(defalias 'yes-or-no-p 'y-or-n-p)   ;; Replace yes questions to y
```

2.11.2 Hungry-delete

Deletes all whitespace with a single delete of backspace.

```
(use-package hungry-delete
  :ensure t
  :config (global-hungry-delete-mode))
```

2.11.3 Minor modes

Some minor modes that are turned on or off. Next to each, a short description is given of what it changes.

```
(tool-bar-mode -1)                  ;; Get rid of tool-bar
(menu-bar-mode -1)                  ;; Git rid of menu
(scroll-bar-mode -1)                ;; Get rid of scroll-bar
(global-auto-revert-mode 1)         ;; Make sure that you're always looking at the late
(delete-selection-mode 1)           ;; Remove text from selection instead of just inser
(display-time-mode 1)               ;; Set clock on lower right side
(electric-pair-mode t)              ;; Enable electric pair mode. It autocompletes cert
(global-subword-mode 1)             ;; Cause M-f to move forward per capitalization wit
(visual-line-mode 1)                ;; Make sure that lines do not disapear at the righ
```

2.11.4 Popup kill-ring

Show options out of the kill ring instead of cycling through each option.

```
(use-package popup-kill-ring
  :ensure t
  :bind ("M-y" . popup-kill-ring))
```

2.11.5 Which key

Provides options for keystrokes. Super useful!

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

2.11.6 TODOS

```
(setq org-todo-keyword-faces
      '(
("DOING" . (:foreground "#05d3fc" :weight bold :box (:line-width 2 :style released-butt
("WAITING" . (:foreground "#fcca05" :weight bold :box (:line-width 2 :style released-bu
("FLEETING" . (:foreground "#f62af9" :weight bold :box (:line-width 2 :style released-b
("LONGTERM" . (:foreground "#c4013c" :weight bold :box (:line-width 2 :style released-b
("CANCELED" . (:foreground "#fc4205" :weight bold :box (:line-width 2 :style released-b
))

(setq org-todo-keywords
      '((sequence "TODO(t)" "DOING(d)" "WAITING(w)" "FLEETING(f)" "|" "LONGTERM(l)" "CA
```

2.12 Try package

Have the option to try packages without actually installing them. If you do

```
M-x try
```

It will give you the option to temporarily install the package. If you close and reopen emacs, the tried out package is removed.

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

2.13 Visual

2.13.1 Beacon

Small package to provide an idea where in which buffer the cursor is atm by showing a small light in the current frame.

```
(use-package beacon
    :ensure t
    :init
    (beacon-mode 1))
```

2.13.2 rainbow

Visualize color codings. So RGB will be colored in its respective color.

```
(use-package rainbow-mode
  :ensure t
  :init (add-hook 'prog-mode-hook 'rainbow-mode))
```

Make visual pairs of delimeters ({<[]>}) etc. Each level gets its own color so it's easy to spot which are pairs.

```
(use-package rainbow-delimiters
  :ensure t
  :init
  (rainbow-delimiters-mode 1))
```

2.13.3 Doom theme

This causes emacs to look a lot better overall. This package makes the coloring and font decissions, so you don't have to. There is a seperate package for the mode-line (the line that containts the time and which file etc.)

```
(use-package doom-themes
  :ensure t
  :config
  ;; Global settings (defaults)
  (setq doom-themes-enable-bold t    ; if nil, bold is universally disabled
doom-themes-enable-italic t) ; if nil, italics is universally disabled
  (load-theme 'doom-one t)

  ;; Enable flashing mode-line on errors
  (doom-themes-visual-bell-config)

  ;; Enable custom neotree theme (all-the-icons must be installed!)
  (doom-themes-neotree-config)
  ;; or for treemacs users
  (setq doom-themes-treemacs-theme "doom-colors") ; use the colorful treemacs theme
  (doom-themes-treemacs-config)

  ;; Corrects (and improves) org-mode's native fontification.
  (doom-themes-org-config))
```

```
(use-package doom-modeline
  :ensure t
  :init (doom-modeline-mode 1))
```

2.13.4 all-the-icons

```
(use-package all-the-icons :ensure t)
```

3 Shortcuts

Here I provide some common shortcuts that I tend to use or want to remember.

3.1 Shorthand notations

There are some shorthands for certain keys, these are as follows:

Shorthand	Corresponding key
C	Control
M	Meta, option or alt (depending on OS)
RET	Return or enter
S	Shift
SPC	Space bar
TAB	Tab key
prefix	C-u followed by the shortcut
VERT	This is a pipe sign
, it is described	
	like this since Org treats this as a new column

3.2 Navigation

Shortcut	Description
Beginner	
PREFIX NUMBER	By typing the PREFIX (C-u) with a number after it followed by a command, you in principle repeat the command number of times
C-n	Move cursor down (next)
C-p	Move cursor up (previous)
C-b	Move cursor left (back)
M-b	Move cursor back one word
C-f	Move cursor right (forward)
M-f	Move cursor forward one word
C-a	Move cursor to beginning of line
M-a	Move cursor to beginning of current or previous sentence
C-e	Move cursor to end of line
M-e	Move cursor to end of current or next sentence
M-{	Move cursor to beginning or previous paragraph
M-}	Move cursor to end or next paragraph
Intermediate	
M-g M-g	Asks you a number and then goes to that line number

3.3 Org

3.3.1 Regular ORG

3.3.2 Tables

Here are some shortcuts that I regularly use or want to have handy nearby.
For a more exhaustive list, check the org table manual.

Shortcut	Description
Re-aligning and motion	
C-c C-c	Re-align without moving
TAB	Re-align table, move to next field and create new row if needed
RET	Move cursor to next row or create a row (useful with horizontal lines)
M-a or M-e	Move to beginning or end of current field, or next field
Regions	
M-RET	Split current field at point and paste what comes after to new line
Column and Row editing	
C-c RET	Insert horizontal line
C-c -	Insert a horizontal line below, with prefix it is above
C-c ^	Sort by column on point
S-ARROW$_{\text{KEY}}$	Move current cell in the direction of the used arrow key
M-ARROW$_{\text{KEY}}$	Move current column or row in the direction of the used arrow key
M-S-LEFT or M-S-UP	Kill current column or row, respectively
M-S-RIGHT or M-S-DOWN	Add column or row
Calculations	
C-c +	You can paste (C-y) a column sum into a field
S-RET	Auto-increment downward is current field is a number or if current is empty, copy the one from above.
Misc	
C-c VERT	Create table or convert from region (csv style)
C-c ‘	Used if you want a seperate buffer to alter a field

3.3.3 Exporting