# Doom configuration #+STARTUP: overview

David Lewis

September 29, 2023

## Identity information

```
(setq user-full-name "David Lewis"
      user-mail-address "davidalewis00@gmail.com")
```

## Doom config

```
(setq! doom-private-dir "~/dotfiles/.doom.d/")
(setq! doom-font (font-spec :family "JetBrainsMono Nerd Font Mono"))
```

## Build information

```
./configure -with-json -with-imagemagick --with-xpm=ifavailable --with-native-compilation
```

## Theme

set theme with doom-theme or load theme with load-theme

```
(setq doom-theme 'doom-base16)
(setq doom-themes-treemacs-theme "doom-colors")
(add-hook 'pdf-tools-enabled-hook 'pdf-view-themed-minor-mode)
```

## Org Setup

### Org Directory

```
(setq org-directory "~/Dropbox/org/")
```

### ansi escape codes

```
(after! org
(defun david/babel-ansi ()
  (when-let ((beg (org-babel-where-is-src-block-result nil nil)))
    (save-excursion
      (goto-char beg)
      (when (looking-at org-babel-result-regexp)
        (let ((end (org-babel-result-end))
```

```
            (ansi-color-context-region nil))
          (ansi-color-apply-on-region beg end))))))
(add-hook 'org-babel-after-execute-hook 'david/babel-ansi)
)
```

## Hide end of file lines

```
;;(fringe-mode 'default')
;;(setq! indicate-empty-lines nil)
(remove-hook 'text-mode-hook #'vi-tilde-fringe-mode)
```

## Hide emphasis markers

```
(setq org-hide-emphasis-markers 't)
```

## Org Tab behaviour

```
  (after! evil-org
    (setq org-tab-first-hook nil))
```

## org code blocks

```
(defun insert-jupyter-python-block ()
  "Inserts a python code block"
  (interactive)
  (progn
    (insert "#+begin_src jupyter-python :session py\n"
            "#+end_src")
    )
  (org-edit-special)
  )

(defun insert-jupyter-ess-block()
  "Inserts an ESS(R) code block"
  (interactive)
  (insert "#+begin_src jupyter-R :session R\n"
          "#+end_src")
  (org-edit-special)
  )

(defun insert-elisp-block()
  "inserts an elisp code block"
  (interactive)
  (insert "#+begin_src elisp\n"
          "#+end_src")
  (org-edit-special)
  )

(defun insert-generic-block()
  "inserts an elisp code block"
  (interactive)
  (insert "#+begin_src " (read-string "Enter src type:") "\n"
          "#+end_src")
  (org-edit-special)
  )
(defun insert-gnuplot-block()
  (interactive)
  (insert "#+begin_src gnuplot :results output :file " (read-string "Enter file name:") "\n"
          "#+end_src")
  (org-edit-special)
  )
```

```
(defun copy-down-x (x)
  "copies the current cell down by the universal arg"
  (interactive "P")
  (cl-loop repeat (or x  1)
           do (org-table-copy-down 0))
        )
(defun universal-test (x)
  (interactive "P")
  (cl-loop repeat x
           do (message "hello"))
  (message "done"))
```

## Functions

```
(defun cache-address ()
 (interactive)
 (progn
   (setq size (read-string "Enter the MM size: "))
   (setq strunit (substring size -2 nil))
   (if (eq (compare-strings strunit nil nil "GB" nil nil) 't)
       (message "true"))
 )
 )
```

## Org default image size

```
(after! evil-org
(setq org-image-actual-width 400))
```

## emacs-jupyter direnv fix

```

(after! jupyter
  (advice-add 'jupyter-command :around #'envrc-propagate-environment)
  )
```

## Emacs anki setup

```
(defun insert-anki-note (heading)
  "Inserts an anki note"
  (interactive "MNote Title:")
  (progn
    (anki-editor--insert-note-skeleton "prefix" "IDA" heading "Basic" '("Back"))
    )
  )
(defun make-anki-note (deck)
  (interactive (list (read-string "Deck: " "IDA")))
  (progn
    (unless (save-excursion
                (org-up-heading-safe)
                ;; don't insert `ANKI_DECK' if some ancestor already has
                ;; the same value
                (and (not (string-blank-p deck))
                    (string= deck (org-entry-get-with-inheritance anki-editor-prop-deck)))))
    (org-set-property anki-editor-prop-note-type "Basic")
    )
  )
(use-package anki-editor
  :after org)
```

## org keymap

```
(map! (:after org
       :map org-mode-map
       :nvi "S-<return>" #'copy-down-x
       :leader (:prefix ("C" . "Anki cards")
                :desc "insert anki note" "i" #'insert-anki-note
                :desc "make anki note" "m" #'make-anki-note)
       :leader
       (:prefix ("j" . "jupyter-source")
        :desc "jupyter-python" "p" #'insert-jupyter-python-block
        :desc "jupyter-R" "r" #'insert-jupyter-ess-block
        :desc "elisp" "e" #'insert-elisp-block
        :desc "gnuplot" "g" #'insert-gnuplot-block
        :desc "generic" "b" #'insert-generic-block)))
```

## org headline

```
(setq org-fontify-todo-headline t)
```

## org-fancy-priorities-mode work around

```
(defadvice! +org-dont-fontify-my-thangs-a (orig-fn &rest args)
  :around '(org-superstar-mode org-fancy-priorities-mode)
  (letf! ((#'font-lock-ensure #'ignore)
          (#'font-lock-flush #'ignore)
          (#'font-lock-fontify-buffer #'ignore))
    (apply orig-fn args)))
```

## org-safe-variables

### html export

```
(add-to-list 'safe-local-eval-forms '(add-hook 'after-save-hook 'org-html-export-to-html t t))
(add-to-list 'safe-local-eval-forms '(add-hook 'after-save-hook 'org-re-reveal-export-to-html t t))
```

### Latex Export

```
(add-to-list 'safe-local-eval-forms '(add-hook 'after-save-hook 'org-latex-export-to-pdf t t))
```

### Org-babel

```
(add-to-list 'safe-local-eval-forms '(add-hook 'after-save-hook 'org-babel-tangle t t))
```

## format all error

```
(add-to-list '+format-on-save-enabled-modes 'web-mode 1)
```

## gnuplot

This block changes the default term type for png file types

```
(after! gnuplot (add-to-list '*org-babel-gnuplot-terms* '(png . "pngcairo transparent")))
```

## org-fragtog

```
(use-package! org-fragtog
  :after org
  :hook (org-mode . org-fragtog-mode)
  :config)
```

## ispell

## Startup

```
(after! org
  (setq! org-startup-with-latex-preview t)
  (setq! org-startup-with-inline-images t)
  (setq! org-latex-default-figure-position "H") ;; requires float package
  (setq! org-latex-image-default-width "0.7\\textwidth")
  (setq! org-cite-global-bibliography (list"~/dotfiles/citations.json"))
  (setq! org-cite-export-processors '(t csl))
  (setq! yas/triggers-in-field t)
  (setq! org-xournalpp-image-type 'png)
  (add-hook 'org-mode-hook 'turn-on-auto-fill)
  (setq! org-export-allow-bind-keywords t))
(with-eval-after-load 'ox-latex
  (add-to-list 'org-latex-classes '("apa" "\\documentclass[11pt]{apa7}"
                                    ("\\part{%s}" . "\\part*{%s}")
                                    ("\\chapter{%s}" . "\\chapter*{%s}")
                                    ("\\section{%s}" . "\\section*{%s}")
                                    ("\\subsection{%s}" . "\\subsection*{%s}")
                                    ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))))

(with-eval-after-load 'ox-latex
  (add-to-list 'org-latex-classes '("apa" "\\documentclass[11pt]{apa7}"
                                    ("\\part{%s}" . "\\part*{%s}")
                                    ("\\chapter{%s}" . "\\chapter*{%s}")
                                    ("\\section{%s}" . "\\section*{%s}")
                                    ("\\subsection{%s}" . "\\subsection*{%s}")
                                    ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))))

(with-eval-after-load 'ox-latex
 (add-to-list 'org-latex-classes `("Assignment"
                                   ,(string-join '("\\documentclass{assignments}"
                                                   ) "\n")
                                  ("\\section{%s}" . "\\section*{%s}")
                                  ("\\subsection{%s}" . "\\subsection*{%s}")
                                  ("\\subsubsection{%s}" . "\\subsubsection*{%s}")
                                  ("\\paragraph{%s}" . "\\paragraph*{%s}")
                                  ("\\subparagraph{%s}" . "\\subparagraph*{%s}"))))
(after! org
  (setq! org-latex-default-table-environment "tabular"))
(after! org
  (setq! org-latex-default-class "Assignment"))

(after! org
  (setq! org-babel-default-header-args:python '((:exports . "both"))))
(after! org
  (setq! org-export-with-toc nil))

(after! org
  (setq! org-export-with-section-numbers nil))


(with-eval-after-load 'ox-latex
  (add-to-list 'org-latex-classes '("mla" "\\documentclass{mla}"
                                    ("\\part{%s}" . "\\part*{%s}")
                                    ("\\chapter{%s}" . "\\chapter*{%s}")
                                    ("\\section{%s}" . "\\section*{%s}")
```

```
                                    ("\\subsection{%s}" . "\\subsection*{%s}")
                                    ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))))
(with-eval-after-load 'ox-latex
  (add-to-list 'org-latex-classes
               '("memo" "\\documentclass{texMemo}"
                  ("\\section{%s}" . "\\section*{%s}")
                  ("\\subsection{%s}" . "\\subsection*{%s}")
                  ("\\subsubsection{%s}" . "\\subsubsection*{%s}"))
                  ))
```

## citations

```
(use-package! bibtex-completion
  :defer t
  :config
  (setq bibtex-completion-additional-search-fields '(keywords)
        bibtex-completion-pdf-field "file")) ; This tell bibtex-completion to look at the File field of
        ↪  the bibtex to figure out which pdf to open

(use-package! bibtex-actions
  :after embark bibtex-completion
  :config
  (add-to-list 'embark-keymap-alist '(bibtex . bibtex-actions-map)))

(use-package! citeproc
  :defer t)

;;; Org-Cite configuration

(use-package! oc
  :after org bibtex-completion bibtex-actions
  :config
  (require 'ox)
  (map! :map org-mode-map
        :localleader
        :desc "Insert citation" "@" #'org-cite-insert)
  (defvar bibtex-actions-bibliography nil)
  (setq org-cite-global-bibliography
        (let ((paths (or bibtex-actions-bibliography
                         bibtex-completion-bibliography)))
          ;; Always return bibliography paths as list for org-cite.
          (if (stringp paths) (list paths) paths)))
  ;; setup export processor; default csl/citeproc-el, with biblatex for latex
  (setq org-cite-export-processors '((t csl))))

;;; Org-cite processors

;;;; Core

(use-package! oc-basic
  :after oc)

(use-package! oc-biblatex
  :after oc)

(use-package! oc-csl
  :after oc
  :config
  (setq org-cite-csl-styles-dir "~/Zotero/styles"))

(use-package! oc-natbib
  :after oc)
```

## set header args

```
(setq! org-global-properties '(("header-args:latex" . ":results output file graphics :imagemagick yes
↪  :headers '(\"\\\\usepackage{tikz}\ \\\\usepackage{siunitx}\ \\\\usepackage{gensymb}\") :fit yes
↪  :iminoptions -density 600")
                               ("header-args" . ":pandoc t")))
```

### org-xournalpp

```
;;(use-package! org-xournalpp
 ;; :config
  ;;(add-hook 'org-mode-hook 'org-xournalpp-mode))
```

### Disable Line wrapping

```
(after! org
  (setq! org-startup-truncated 'nil)
  )
```

### Org pomodoro

```
(setq alert-user-configuration (quote ((((:category . "org-pomodoro")) libnotify nil))))
(defun david/org-pomodoro-time ()
  "Return the remaining pomodoro time"
  (if (fboundp 'org-pomodoro-active-p)
  (if (org-pomodoro-active-p)
      (cl-case org-pomodoro-state
        (:pomodoro
           (format "🍅 %d minutes - %s" (/ (org-pomodoro-remaining-seconds) 60) org-clock-heading))
        (:short-break
         (format "☕ %d minutes" (/ (org-pomodoro-remaining-seconds) 60)))
        (:long-break
         (format "☕ %d minutes" (/ (org-pomodoro-remaining-seconds) 60)))
        (:overtime
         (format "☹ %d minutes" (/ (org-pomodoro-remaining-seconds) 60))))
    "No active pomo") "no active pomo"))
(use-package org-pomodoro
  :ensure t
  :commands (org-pomodoro)
  :config
  (setq
   org-pomodoro-length 50
   org-pomodoro-short-break-length 10
   ))
```

### nix hack

Org mode (latex export) has the wrong time. Not sure how to fix. This does not work. I guess this only happens when inside a flake environment. I wouldn't worry about it.

### plantuml

```
(after! org (setq! org-plantuml-exec-mode 'plantuml))
```

### mathjax

## Latex setup

### Use LuaTex

```
(setq! TeX-engine 'luatex)
(after! org
  ;(setq! org-latex-pdf-process '("PDFLATEX=lualatex LATEX=lualatex texi2dvi --pdf --clean --verbose
  ↪  --batch --shell-escape -output-directory=%o %f")))
(setq! org-latex-pdf-process '("latexmk -f -pdf -pdflatex=%latex -interaction=nonstopmode -shell-escape
↪  -output-directory=%o %f")))
(after! org
  (setq! org-latex-compiler "lualatex"))
(after! org
  (setq! org-latex-listings 'minted))
(after! org
  (setq! org-latex-minted-options
         '(
             ("fontsize" "\\scriptsize")
             ("breaklines" "true")
             ("breakanywhere" "true")
             )
                ))
```

### Extra Packages

```
(after! org
  (setq! org-latex-packages-alist '())
  (add-to-list 'org-latex-packages-alist '("" "physics" t))
  (add-to-list 'org-latex-packages-alist '("" "minted" nil))
)
```

### Keymap

```
(map!  (:after auctext
       :map LaTeX-mode-map
       :leader
       :desc "compile" "c" #'TeX-command-master))
```

## Spell setup

### Personal Dictionary

```
(setq! ispell-personal-dictionary "~/.config/spell/dict.txt")
```

### Fix hunspell bug

```
(setq ispell-program-name "hunspell")
;;(ispell-check-version)
```

# Python setup

## Anaconda directory

```
(setq conda-anaconda-home "~/opt/anaconda")
```

## LSP nix

```
(after! lsp-python-ms
  (setq lsp-python-ms-executable (executable-find "python-language-server"))
  (set-lsp-priority! 'mspyls 1))
(after! lsp-rust-rls
  (setq lsp-rust-rls-server-command (executable-find "rls"))
  (set-lsp-priority! 'rls 1))

(after! lsp-clients-lua-language-server
  (setq lsp-clients-lua-language-server-bin (executable-find "lua-language-server"))
  (set-lsp-priority! 'lua-language-server 1))

(after! lsp-clangd
  (setq lsp-clients-clangd-executable (executable-find "clangd"))
  (set-lsp-priority! 'clangd 1))
(after! lsp-ts-ls
  (setq lsp-clients-clangd-executable (executable-find "clangd"))
  (set-lsp-priority! 'clangd 1))
```

# R setup

## keymap

```
(map! (:after ess-mode
       :map ess-mode-map
       :nvi "C-<return>" #'ess-eval-line-and-step
       :nvi "M-e" #'insert-R-assign
       )
      )
```

## Font lock keywords

### R-major-mode

```
(setq ess-R-font-lock-keywords '(
 (ess-R-fl-keyword:keywords . t)
 (ess-R-fl-keyword:constants . t)
 (ess-R-fl-keyword:modifiers . t)
 (ess-R-fl-keyword:fun-defs . t)
 (ess-R-fl-keyword:assign-ops . t)
 (ess-R-fl-keyword:%op% . t)
 (ess-fl-keyword:fun-calls . t)
 (ess-fl-keyword:numbers . t)
 (ess-fl-keyword:operators . t)
 (ess-fl-keyword:delimiters . t)
 (ess-fl-keyword:= . t)
 (ess-R-fl-keyword:F&T . t)
 )
)
```

#### R-inferior-mode

```
(setq inferior-ess-r-font-lock-keywords '(
 (ess-R-fl-keyword:keywords . t)
 (ess-R-fl-keyword:constants . t)
 (ess-R-fl-keyword:modifiers . t)
 (ess-R-fl-keyword:fun-defs . t)
 (ess-R-fl-keyword:assign-ops . t)
 (ess-R-fl-keyword:%op% . t)
 (ess-fl-keyword:fun-calls . t)
 (ess-fl-keyword:numbers . t)
 (ess-fl-keyword:operators . t)
 (ess-fl-keyword:delimiters . t)
 (ess-fl-keyword:= . t)
 (ess-R-fl-keyword:F&T . t)
 )
)
```

### Custom Functions

```
(defun insert-R-assign ()
  "Inserts the assign statement in R <-"
  (interactive)
  (insert "<-")
  )
```

## General configuration

### remove line numbers

```
(setq display-line-numbers-type nil)
```

### Fix treemacs ace-window bug

```
(require 'ace-window)
```

### deletes compilation buffer if successful (ignores python buffers)

```
(add-hook 'compilation-finish-functions
          (lambda(buffer string)
            (if (and (null (string-match ".*exited abnormally.*" string))
                     (null(eq major-mode 'inferior-python-mode)))
                ;; make compilation window go away after a few seconds
                (progn
                  (run-at-time
                   "1 sec" nil 'delete-windows-on
                   (get-buffer-create "*compilation*"))
                  (print major-mode)
                  (message "Compilation finished successfully")))))
```

### ligatures

```
(setq! +ligatures-extras-in-modes nil)
```

### ispell dictionary

```
(setq! ispell-dictionary "en_US")
```

## Key Map

### General Buffers

```
(map! :leader
      :desc "treemacs" "0" #'treemacs
      :desc "last-buffer" "l" #'evil-switch-to-windows-last-buffer
      :nv "`" nil
      (:prefix ("w")
       :desc "ace-window" "a" #'ace-window))
```

### Python mode map

```
(map!  (:map python-mode-map
        :localleader
        :desc "repl" "'" #'+python/open-ipython-repl
        (:prefix ("s" . "send")
         :desc "buffer" "b" #'python-shell-send-buffer
         :desc "function" "f" #'python-shell-send-defun
         :desc "region" "r" #'python-shell-send-region
         :desc "statement" "s" #'python-shell-send-statement)))
```

### prolog map

```
(map!  (:map prolog-mode-map
        :localleader
        :desc "repl" "'" #'run-prolog
        :desc "file" "f" #'prolog-consult-buffer
        :desc "region" "r" #'prolog-consult-region
        :desc "predicate" "p" #'prolog-consult-region))
```

### Doc-view mode map

```
(map! (:map doc-view-mode-map
       :nv "l" #'doc-view-next-page
       :nv "h" #'doc-view-previous-page))
```

### mips mode map

```
(map! (:map mips-mode-map
       :localleader
       (:prefix ("s" . "send")
        :desc "file" "f" #'mips-run-file
        :desc "region" "r" #'mips-run-region
        :desc "buffer" "b" #'mips-run-region)))
```

### haskell map

```
(map! (:map haskell-mode-map
       :localleader
       :desc "send-file" "f" #'haskell-process-load-file
       :desc "open-haskell" "'" #'run-haskell))
```

## remote

```
(setq projectile-file-exists-remote-cache-expire nil)
(setq tramp-auto-save-directory "~/Documents/tramp-autosave")
```

# Helpful info

### Font variables

- doom-font `(normal font)`
- doom-variable-pitch-font `(easy reading font)`
- doom-big-font `(doom-big-font-mode for presentations)`

### Useful customization functions

- load! `(load external .el files)`
- use-package! `(for configuring packages)`
- after! `(runs config after packages has loaded)`
- add-load-path! `(adds directories to load-path variable)`
- map! `(binds keys)`