Contents

1 Visual

1.1 Tweaks

1.1.1 Font

The best programming font I know of is Cascadia Code.

```
(setq doom-font (font-spec :family "Cascadia Code" :size 16))
(doom/reload-font)
```

1.1.2 Helm text un-enlargement

```
(setq +helm-posframe-text-scale 0)
```

1.1.3 Doom Modeline

```
(doom-modeline-mode t)
(setq doom-modeline-height 36)
```

1.2 Theme

```
(setq doom-theme 'doom-one)
(load-theme 'doom-gruvbox t)
```

1.3 Org

1.3.1 Superstar

```
(use-package! org-superstar
  :hook (org-mode . org-superstar-mode))

(after! org-superstar
  (setq org-superstar-headline-bullets-list '(9679 9673 9675))
  (setq org-superstar-item-bullet-alist '((42 . 9642) (43 . 9702) (45 . 8226)))
  (setq org-superstar-leading-bullet "-")
  (setq org-superstar-special-todo-items t)
  (org-superstar-restart))
```

1.3.2 Olivetti

```
(use-package! olivetti
  :config
  (setq olivetti-body-width '100
        olivetti-margin-width '20
        olivetti-style t)
  :hook (org-mode . olivetti-mode))
```

1.3.3 Variable-pitch font

```
(use-package! org-variable-pitch
  :config
  (setq org-variable-pitch-fixed-faces
        '(org-block
          org-block-begin-line
          org-block-end-line
          org-code
          org-document-info-keyword
          org-done
          org-formula
          org-indent
          org-meta-line
          org-special-keyword
          org-table
          org-todo
          org-verbatim
          org-date
          org-drawer
          org-property-value
          org-priority
          org-hide
          org-superstar-leading
          org-ellipsis
          line-number
          org-tag
          org-target))
  :hook ((org-mode) . org-variable-pitch-minor-mode))
(set-face-attribute 'variable-pitch nil :family "Noto Serif")
```

1.3.4 Don't enable line number or hl-line-mode in org

Please stop messing with my faces and stuff I can't write this code Pl

```
(global-hl-line-mode -1)
```

```
(setq doom--hl-line-mode -1)
(global-display-line-numbers-mode -1)
(add-hook! 'prog-mode #'display-line-numbers-mode)
(add-hook! 'prog-mode #'hl-line-mode)

(remove-hook! text-mode #'display-line-numbers-mode)
(add-hook! org-mode #'(lambda () (global-display-line-numbers-mode -1)))
(add-hook! org-mode #'(lambda () (global-hl-line-mode -1)))
```

1.3.5 Large heading fonts

```
(after! org
  (let (var (heading-scale 1.25))
    (set-face-attribute 'org-level-1 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-2 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-3 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-4 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-5 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-6 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-7 nil :height heading-scale :extend t)
    (set-face-attribute 'org-level-8 nil :height heading-scale :extend t)))
```

1.3.6 Make org-ref citations look pretty

Make org-ref look pretty.

```
(use-package! org-ref-prettify
  :after org
  :hook (org-mode . org-ref-prettify-mode))
```

1.4 Word wrap

Apparently according to someone on the internet evil-better-visual-line is better than regular visual line.

```
(use-package! evil-better-visual-line
  :hook (org-mode . evil-better-visual-line-on))
```

2 Org-mode tweaks

2.1 Blank line on headlines

It's nice to have some space before and after headlines.

```
(after! org
  (setq org-blank-before-new-entry '((heading . always) (plain-list-item))
        org-ascii-headline-spacing '(1 . 2))
  (map! :map evil-org-mode-map
        :desc "<C-return>"
        :nvi "<C-return>" #'evil-org-org-insert-heading-respect-content-below))
```

2.2 Roam

Base roam package:

```
(use-package org-roam
  :hook ((after-init . org-roam-setup)
         (org-roam-backlinks-mode . visual-line-mode))
  :config
  '(org-roam-dailies-capture-templates
    '(("d" "default" entry "* %?\
        " :target
        (file+head "%<%Y-%m-%d>.org" "#+title: %<%Y-%m-%d>
              ")
        :empty-lines-after 1
        :empty-lines-before 1)))

  (setq org-roam-capture-templates
        '(("d" "default" plain "%?" :target
           (file+head "pages/%<%Y%m%d%H%M%S>-${slug}.org" "
  #+filetags:
  #+title: ${title}
  - Links ::


      ")
           :unnarrowed t)))
  (setq org-roam-v2-ack t)
  (setq org-roam-directory "~/notes")
  (setq org-roam-dailies-directory "journals/")
  (setq org-roam-db-node-include-function
        (lambda ()
          (not (member "roam_exclude" (org-get-tags)))))
  (setq org-roam-mode-section-functions
        (list #'org-roam-backlinks-section
              #'org-roam-reflinks-section
              ;; #'org-roam-unlinked-references-section
              ))
  :bind (("C-c n f" . org-roam-node-find)
         ("C-c n c" . org-roam-capture)
         ("C-c n g" . org-roam-ui-mode)
         ("C-c n r" . org-roam-node-random)
         ("C-c n d" . org-roam-dailies-capture-today)
         (:map org-mode-map
          (("C-c n i" . org-roam-node-insert)
           ("C-c C-w" . org-roam-refile)
           ("C-c n o" . org-id-get-create)
           ("C-c n t" . org-roam-tag-add)
           ("C-c n a" . org-roam-alias-add)
           ("C-c n l" . org-roam-buffer-toggle)))))
(org-roam-db-autosync-mode)
```

Org-roam-bibtex:

```
  (use-package! org-roam-bibtex
    :after org-roam
    :hook (org-roam-mode . org-roam-bibtex-mode)
    :config (require 'org-ref)
    (org-roam-bibtex-mode t))
```

2.3 Org-ref

```
  (use-package! org-ref
    ;; :after org-cite
    :config (setq org-ref-default-bibliography "~/notes/pages/sources.bib")
    :init
    (setq bibtex-completion-bibliography "~/notes/pages/sources.bib")
    :bind ("C-c r i" . org-ref-cite-insert-helm))
```

2.4 Exporting

2.4.1 Only export the content of headlines tagged as “ignore”

```
;;; ox-extra.el --- Convenience functions for org export

;; Copyright (C) 2014  Aaron Ecay

;; Author: Aaron Ecay <aaronecay@gmail.com>

;; This program is free software; you can redistribute it and/or modify
;; it under the terms of the GNU General Public License as published by
;; the Free Software Foundation, either version 3 of the License, or
;; (at your option) any later version.

;; This program is distributed in the hope that it will be useful,
;; but WITHOUT ANY WARRANTY; without even the implied warranty of
;; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;; GNU General Public License for more details.

;; You should have received a copy of the GNU General Public License
;; along with this program.  If not, see <http://www.gnu.org/licenses/>.

;;; Commentary:

;; This file contains some convenience functions for org export, which
;; are not part of org's core.  Call `ox-extras-activate' passing a
;; list of symbols naming extras, which will be installed globally in
;; your org session.
;;
;; For example, you could include the following in your .emacs file:
;;
;;    (require 'ox-extra)
;;    (ox-extras-activate '(latex-header-blocks ignore-headlines))
;;

;; Currently available extras:

;; - `latex-header-blocks' -- allow the use of latex blocks, the
;; contents of which which will be interpreted as #+latex_header lines
;; for export.  These blocks should be tagged with #+header: :header
;; yes.  For example:
;; #+header: :header yes
;; #+begin_export latex
;;   ...
;; #+end_export

;; - ‘ignore-headlines’ -- allow a headline (but not its children) to
;; be ignored.  Any headline tagged with the ’ignore’ tag will be
;; ignored (i.e. will not be included in the export), but any child
;; headlines will not be ignored (unless explicitly tagged to be
;; ignored), and will instead have their levels promoted by one.

;; TODO:
;; - add a function to org-mode-hook that looks for a ox-extras local
;;   variable and activates the specified extras buffer-locally
;; - allow specification of desired extras to be activated via
;;   customize

;;; Code:

(require ’ox)
(eval-when-compile (require ’cl))

(defun org-latex-header-blocks-filter (backend)
  (when (org-export-derived-backend-p backend ’latex)
    (let ((positions
     (org-element-map (org-element-parse-buffer ’greater-element nil) ’export-bloc
       (lambda (block)
         (when (and (string= (org-element-property :type block) "LATEX")
        (string= (org-export-read-attribute
            :header block :header)
           "yes"))
     (list (org-element-property :begin block)
           (org-element-property :end block)
           (org-element-property :post-affiliated block)))))))
      (mapc (lambda (pos)
        (goto-char (nth 2 pos))
        (destructuring-bind
      (beg end &rest ignore)
      (org-edit-src-find-region-and-lang)
    (let ((contents-lines (split-string
              (buffer-substring-no-properties beg end)
              "\n")))
      (delete-region (nth 0 pos) (nth 1 pos))
      (dolist (line contents-lines)
        (insert (concat "#+latex_header: "
            (replace-regexp-in-string "\\‘ *" "" line)
            "\n"))))))
      ;; go in reverse, to avoid wrecking the numeric positions
      ;; earlier in the file
      (reverse positions)))))


;; During export headlines which have the "ignore" tag are removed
;; from the parse tree.  Their contents are retained (leading to a
;; possibly invalid parse tree, which nevertheless appears to function
;; correctly with most export backends) all children headlines are
;; retained and are promoted to the level of the ignored parent
;; headline.
;;
;; This makes it possible to add structure to the original Org-mode
;; document which does not effect the exported version, such as in the
;; following examples.
;;
;; Wrapping an abstract in a headline
;;
;;     * Abstract                        :ignore:
;;     #+LaTeX: \begin{abstract}
;;     #+HTML: <div id="abstract">
;;
;;     ...
;;
;;     #+HTML: </div>
;;     #+LaTeX: \end{abstract}
;;
;; Placing References under a headline (using ox-bibtex in contrib)
;;
;;     * References                     :ignore:
;;     #+BIBLIOGRAPHY: dissertation plain
```

```
;;
;; Inserting an appendix for LaTeX using the appendix package.
;;
;;     * Appendix                       :ignore:
;;     #+LaTeX: \begin{appendices}
;;     ** Reproduction
;;     ...
;;     ** Definitions
;;     #+LaTeX: \end{appendices}
;;
(defun org-export-ignore-headlines (data backend info)
  "Remove headlines tagged \"ignore\" retaining contents and promoting children.
Each headline tagged \"ignore\" will be removed retaining its
contents and promoting any children headlines to the level of the
parent."
  (org-element-map data 'headline
    (lambda (object)
      (when (member "ignore" (org-element-property :tags object))
        (let ((level-top (org-element-property :level object))
              level-diff)
          (mapc (lambda (el)
                  ;; recursively promote all nested headlines
                  (org-element-map el 'headline
                    (lambda (el)
                      (when (equal 'headline (org-element-type el))
                        (unless level-diff
                          (setq level-diff (- (org-element-property :level el)
                                              level-top)))
                        (org-element-put-property el
                          :level (- (org-element-property :level el)
                                    level-diff)))))
                  ;; insert back into parse tree
                  (org-element-insert-before el object))
                (org-element-contents object)))
        (org-element-extract-element object)))
    info nil)
  data)

(defconst ox-extras
  '((latex-header-blocks org-latex-header-blocks-filter org-export-before-parsing-h
      (ignore-headlines org-export-ignore-headlines org-export-filter-parse-tree-func
    "A list of org export extras that can be enabled.
  Should be a list of items of the form (NAME FN HOOK).  NAME is a
  symbol, which can be passed to ‘ox-extras-activate’.  FN is a
  function which will be added to HOOK.")

  (defun ox-extras-activate (extras)
    "Activate certain org export extras.
  EXTRAS should be a list of extras (defined in ‘ox-extras’) which
  should be activated."
    (dolist (extra extras)
      (let* ((lst (assq extra ox-extras))
       (fn (nth 1 lst))
       (hook (nth 2 lst)))
        (when (and fn hook)
    (add-hook hook fn)))))

  (defun ox-extras-deactivate (extras)
    "Deactivate certain org export extras.
  This function is the opposite of ‘ox-extras-activate’.  EXTRAS
  should be a list of extras (defined in ‘ox-extras’) which should
  be activated."
    (dolist (extra extras)
      (let* ((lst (assq extra ox-extras))
       (fn (nth 1 lst))
       (hook (nth 2 lst)))
        (when (and fn hook)
    (remove-hook hook fn)))))

(ox-extras-activate '(ignore-headlines))
```

2.4.2 Latex Configuration

Always use these packages in latex exports.

```
(setq org-latex-default-packages-alist '(("AUTO" "inputenc" t
                                        ("pdflatex"))
                                           ("T1" "fontenc" t
                                        ("pdflatex"))
                                           (#1="" "graphicx" t)
```

```
                                          (#1# "grffile" t)
                                          (#1# "longtable" nil)
                                          (#1# "wrapfig" nil)
                                          (#1# "rotating" nil)
                                          ("normalem" "ulem" t)
                                          (#1# "amsmath" t)
                                          (#1# "textcomp" t)
                                          (#1# "amssymb" t)
                                          (#1# "capt-of" nil)
                                          (#1# "biblatex" nil)
                                          (#1# "hyperref" nil)))
```

These are necessary for something, I'm not sure why but I'll keep them anyway.

```
(require 'org-ref-scopus)
(require 'org-ref-pubmed)
(require 'org-ref-sci-id)
```

CSL styles are found under `file:///home/jadench/.emacs.doom/.local/straight/repos/org-ref/citeproc/csl-styles/` Possibilities are:

- `elsevier-with-titles.csl`
- `chicago-author-date-16th-edition.csl`
- `apa-5th-edition.csl`
- `elsevier-harvard.csl`
- `apa-numeric-superscript-brackets.csl`

I also have my own styles in `file:///home/jadench/.emacs.doom/tex/csl/`. More can be found online at `https://github.com/citation-style-language/styles`

```
(setq org-ref-csl-default-style "~/.emacs.doom/tex/csl/association-for-computing-machi
```

Set up the default bibliography and export variables:

```
(setq org-ref-default-bibliography "~/notes/pages/sources.bib")
(setq reftex-default-bibliography "~/notes/pages/sources.bib")
(setq org-export-with-broken-links t)
```

```
(setq latex-run-command "pdflatex")
(setq bibtex-dialect 'biblatex)
(setq org-latex-pdf-process
'("%latex -shell-escape -interaction nonstopmode -output-directory %o %b" "biber %b" "r
```

2.4.3 Pandoc

```
(use-package! ox-pandoc)
```

2.5 Org-noter

```
  (use-package! org-noter-pdftools)
  (use-package! org-noter
    :config
    (require 'org-noter-pdftools))
```

2.6 Scientific Writing

2.6.1 Bibliography stuff

```
(setq
 bibtex-autokey-titlewords 3
 bibtex-completion-bibliography '("~/notes/pages/sources.bib")
 bibtex-completion-library-path '("~/notes/pages/bibtex-pdfs/"))

(use-package! helm-bibtex
  :after org-ref)
```

2.7 Agenda

2.7.1 Misc config

```
(add-hook! org-agenda #'org-agenda-to-appt)
(map! :map global :m "C-c a" 'org-agenda)

(after! org
  (setq org-agenda-columns-add-appointments-to-effort-sum t
        org-todo-keywords '((sequence "TODO(t)" "NEXT(n)" "PROG(r)" "EXTD(e!)" "POST(p@
        org-agenda-span 1
        org-deadline-warning-days 99
        org-agenda-skip-deadline-if-done nil
        org-habit-show-all-today t
        org-habit-show-habits-only-for-today nil
        org-agenda-start-day "-0d"
        org-agenda-skip-timestamp-if-done nil
        org-agenda-skip-deadline-if-done nil
        org-agenda-skip-scheduled-if-deadline-is-shown 'not-today;'repeated-after-deadl
        org-agenda-skip-timestamp-if-deadline-is-shown nil
        org-agenda-entry-text-maxlines 20
        org-agenda-include-diary t
        org-agenda-prefix-format " %?-3t %-11s %3e "
        org-agenda-keyword-format '("")
        org-agenda-remove-tags t
        org-agenda-sorting-strategy '(;(agenda deadline-down todo-state-up effort-down
                                      (agenda time-up deadline-up priority-down todo-st
                                      (todo priority-down category-keep)
                                      (tags priority-down category-keep)
                                      (search category-keep))
        org-agenda-skip-scheduled-if-done t
        org-agenda-span 'day))

(after! org
  (setq org-agenda-custom-commands
        '(("g" "Good agenda"
           ((agenda ""
                    ((org-agenda-overriding-header "Agenda and Tonight's Homework")))
            (alltodo ""
                     ((org-agenda-overriding-header "PROJECTS")
                      (org-agenda-prefix-format " %?-3t %?-11s %3e ")
                      (org-super-agenda-groups
                       '((:discard (:todo "SOMEDAY" :not (:tag "PROJECT")))
                         (:auto-outline-path t)
                         (:discard
                          (:anything))))))
            (alltodo ""
                     ((org-agenda-overriding-header "Other")
                      (org-agenda-prefix-format " %?-3t %3e ")
                      (org-super-agenda-groups
                       '((:name "Bucket List" :and
                          (:todo "SOMEDAY" :tag "PERSONAL")
                          :order 1)
                         (:name "Someday Maybe" :todo "SOMEDAY" :order 10)
                         (:name "Everything Else" :anything t :order 20))))))
           nil nil)
          ("n" "Agenda and all TODOs"
           ((agenda "" nil)
            (alltodo "" nil))
           nil))))
```

2.7.2 Agenda files

```
 (setq org-agenda-files
 '("~/notes/pages/20220204195459-english_essay_the_black_cat_due_2022_02_11.org" "~/Doc
```

2.7.3 Org-super-agenda

```
(use-package! org-super-agenda)
(setq org-super-agenda-header-map (make-sparse-keymap)
      org-super-agenda-groups
      `(
        (:name "Today's Schedule" :time-grid t :order 2)
        (:name "Meetings" :tag "meeting" :tag "clubs" :tag "club" :order 2)
        (:name "OVERDUE"
         :and (:not (:todo "CNCL" :todo "DONE" :todo "FAIL") :deadline past)
         :order 1)
        (:name "School Habits" :and (:tag "school" :tag "habit") :order 4)
        (:name "Homework"
         :and (:tag "school" :tag "homework" :deadline (before ,(org-read-date nil nil
         :order 5 )
        (:name "Tests and Quizzes" :tag
         ("test" "quiz" "assessment" "conference")
         :order 3)
        (:name "Upcoming Schoolwork/Homework"
         :and (:tag ("school" "homework") :deadline future)
         :order 6)
        (:name "Personal Habits"
         :and (:tag "personal" :habit t)
         :order 8)
        (:name "Personal TODO list"
         :tag ("personal")
         :order 7)
        (:name "Emails" :tag "email" :order 8)
        (:name "Scheduled work"
         :scheduled t
         :order 10)
        (:time-grid t)
        (:discard (:tag "drill"))))
(add-hook! org-agenda-before-finalize #'org-super-agenda-mode)
(org-super-agenda-mode)
```

2.7.4 Capture Templates

```
  (global-set-key (kbd "C-c c") 'org-capture)
  (setq org-capture-templates
        '(("p" "Personal TODO" entry
           (file+headline "~/Documents/personal.org" "Personal TODO list")
           "* TODO %^{Headline} :personal:%^{Tags}:
SCHEDULED: %^{Scheduled}t DEADLINE: %^{Deadline}t
:PROPERTIES:
:EFFORT: %^{Effort}
:END:
  ")
          ("H" "Habit" entry
           (file+headline "~/Documents/personal.org" "Personal TODO list")
           "* TODO %^{Headline} :personal:habit:%^{Tags}:
SCHEDULED: %^{Scheduled}t
:PROPERTIES:
:EFFORT: %^{Effort}
:STYLE: habit
:END:
  ")
          ("n" "Quick note" entry
           (file+headline "~/Documents/personal.org" "Quick Notes")
           "* %^{Headline}
      ENTERED: %U
    " :prepend t)
          ("a" "Test/Assessment/Quiz " entry
           (file "~/org/todo.org")
           "* %^{Test Name} :school:%^{Tags}:
DEADLINE: %^{Deadline}t ENTERED: %U" :prepend t :time-prompt t)
        ("P" "Project TODO" entry
         (file "~/org/todo.org")
         "* TODO %^{Project name} [/] :project:%^{Tags}:
SCHEDULED: %^{Scheduled}t DEADLINE: %^{Deadline}t ENTERED: %U" :prepend t :time-prompt
        ("e" "Email TODO" entry
         (file "~/org/todo.org")
         "* TODO %^{Task} :email:%^{Tags}:
DEADLINE: %^{Deadline}t ENTERED: %U" :prepend t :time-prompt t)
        ("m" "Meeting entry" entry
         (file "~/org/todo.org")
         "* %^{prompt} :meeting:%^{tags}:
    DEADLINE: %^{Deadline}T ENTERED: %U" :prepend t :time-prompt t)
        ("h" "Homework entry" entry
         (file "~/org/todo.org")
         "* TODO %^{prompt}    :school:homework:
DEADLINE: %^{Deadline}t ENTERED %U
:PROPERTIES:
:EFFORT: %^{Effort}
:END:
  " :prepend t :time-prompt t)))
```

2.8 Auto list

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

2.9 List promotion and demotion

```
(after! org (setq org-list-demote-modify-bullet '(("-" . "+") ("+" . "*") ("*" . "*")))
```

3 Completion

3.1 Ivy

```
(use-package! ivy
  :demand t
  :config
  (ivy-mode t))
```