Contents

1 MELPA

```
(require 'package)
(let* ((no-ssl (and (memq system-type '(windows-nt ms-dos))
^^I^^I    (not (gnutls-available-p))))
       (proto (if no-ssl "http" "https")))
  (when no-ssl (warn "\
Your version of Emacs does not support SSL connections,
which is unsafe because it allows man-in-the-middle attacks.
There are two things you can do about this warning:
1. Install an Emacs version that does support SSL and be safe.
2. Remove this warning from your init file so you won't see it again."))
  (add-to-list 'package-archives (cons "melpa" (concat proto "://melpa.org/packages/"
  ;; Comment/uncomment this line to enable MELPA Stable if desired.  See `package-arc
  ;; and `package-pinned-packages`. Most users will not need or want to do this.
  ;;(add-to-list 'package-archives (cons "melpa-stable" (concat proto "://stable.melp
```

```
  )
(package-initialize)
```

2 Coding styles

2.1 C

```
(setq-default c-basic-offset 8
^^I      c-default-style "k&r"
^^I      tab-width 8
^^I      indent-tabs-mode t)
```

2.2 Python

```
(setq-default python-basic-offset 8
^^I      tab-width 8
^^I      indent-tabs-mode t)
```

2.3 HTML

```
(add-hook 'html-mode-hook
^^I  (lambda ()
^^I    ;; Default indentation is usually 2 spaces, changing to 4.
^^I    (set (make-local-variable 'sgml-basic-offset) 4)))

(defun html-show-toc ()
  "Shows a TOC based on headings tags <H[1-6]>"
  (interactive)
  (if sgml-tags-invisible
      (error "SGML tags are invisible")
    (occur "<h[1-6]>")
    (pop-to-buffer "*Occur*")
    (vc-toggle-read-only)
    (goto-char (point-min))
    (replace-string "<h1>" "")
    (goto-char (point-min))
    (replace-string "<h2>" "  ")
    (goto-char (point-min))
    (replace-string "<h3>" "    ")
    (goto-char (point-min))
```

```
    (replace-string "<h4>" "      ")
    (goto-char (point-min))
    (replace-string "<h5>" "        ")
    (goto-char (point-min))
    (replace-string "<h6>" "          ")
    (goto-char (point-min))
    (replace-regexp "</h[1-6]>" "")
    (goto-char (point-min))
    (toggle-read-only 1)))

(defun html-end-of-line ()
  "If there is an HTML tag at the end of the line, then go to start of tag.
 Otherwise go to the real end of the line."
  (interactive)
  (if (or (looking-at ".*>$") ; if we're on a line that ends with a tag
^^I  (and (= (char-before) 62)
^^I       (= (point) (save-excursion
^^I^^I^^I    (end-of-line)
^^I^^I^^I    (point))))) ; or we're at the end of a line
^^I^^I^^I^^I^^I; with a tag
      (let ((where-now (point)))
^^I(narrow-to-region
^^I (save-excursion
^^I   (beginning-of-line)
^^I   (point))
^^I (save-excursion
^^I   (end-of-line)
^^I   (point)))
^^I(end-of-line)
^^I(re-search-backward "<" nil t)
^^I(if (= (point) where-now)
^^I    (end-of-line))
^^I(widen))
    (end-of-line)))

(add-hook 'html-helper-mode-hook
^^I  (lambda ()
^^I    (define-key html-helper-mode-map "\C-e" 'html-end-of-line)))
```

3 Backup control

```
;; Instea of putting *~ backups in current directory,
;; put them in local .saves
(setq backup-directory-alist `(("." . ".saves")))

;; Just to stop spamming backup files
(setq delete-old-versions t
      kept-new-versions 6
      kept-old-versions 2
      version-control t)
```

4 Display control

4.1 Hide top bar

```
(menu-bar-mode -1)
```

4.2 Line numbers

```
(setq linum-format "%4d  ")
(global-linum-mode 1)
```

4.3 Scrolling

```
(setq redisplay-dont-pause t
      scroll-margin 1
      scroll-step 1
      scroll-conservatively 10000
      scroll-preserve-screen-position 1)
```

4.4 Silence annoying bell

```
(setq ring-bell-function 'ignore)
```

4.5 Character limit in Org-Mode

```
(add-hook 'org-mode-hook '(lambda () (setq fill-column 80)))
(add-hook 'org-mode-hook 'turn-on-auto-fill)
```

4.6 Pair the braces

```
(require 'autopair)
(autopair-global-mode)
```

4.7 Open shell on F1

```
(global-set-key (kbd "<f1>") 'shell)
```

4.8 Show Paren Mode

```
(show-paren-mode 1)
(setq show-paren-delay 0)
```

4.9 Disable validate link in HTML

```
(setq org-html-validation-link nil)
```

4.10 Window management

```
(global-set-key (kbd "C-x <up>") 'windmove-up)
(global-set-key (kbd "C-x <down>") 'windmove-down)
(global-set-key (kbd "C-x <left>") 'windmove-left)
(global-set-key (kbd "C-x <right>") 'windmove-right)
```

4.11 Go to line preview

```
(global-set-key [remap goto-line] 'goto-line-preview)
```

5 LSP

```
;; Give emacs some RAW power, yes
(setq gc-cons-threshold 100000000)
(setq read-process-output-max (* 1024 1024)) ;; 1mb

;; Configure LSP-UI by https://emacs-lsp.github.io/lsp-ui/
;; Optional - provides fancier overlays.
(use-package lsp-ui
  :ensure t
  :commands lsp-ui-mode)
;; Sideline options
```

```
(setq lsp-ui-sideline-show-diagnostics t)
(setq lsp-ui-sideline-show-hover nil)
(setq lsp-ui-sideline-show-code-actions t)
(setq lsp-ui-sideline-update-mode nil)

(use-package lsp-mode
  :hook ((go-mode . lsp)
^^I (rust-mode . lsp)
^^I (c++-mode . lsp)
^^I (c-mode . lsp)
^^I (js-mode . lsp)
^^I (html-mode . lsp)
^^I (python-mode . lsp)
^^I (lsp-mode . lsp-enable-which-key-integration))
  :commands lsp)

(setq lsp-keymap-prefix "C-c l")

(global-set-key (kbd"C-c C-c") 'lsp-find-definition)
(global-set-key (kbd"C-c f") 'lsp-find-definition)

;; Set up before-save hooks to format buffer and add/delete imports.
;; Make sure you don't have other gofmt/goimports hooks enabled.
(defun lsp-go-install-save-hooks ()
  (add-hook 'before-save-hook #'lsp-format-buffer t t)
  (add-hook 'before-save-hook #'lsp-organize-imports t t))
(add-hook 'go-mode-hook #'lsp-go-install-save-hooks)

;; Automatically format code on save
(setq gofmt-command "goimports")
(add-hook 'before-save-hook 'gofmt-before-save)

;; Optional - provides snippet support.
(use-package yasnippet
  :ensure t
  :commands yas-minor-mode
  :hook (go-mode . yas-minor-mode))
```

6 Dashboard

```
;; Enable dashboard
(require 'dashboard)
;; Add the hook
(dashboard-setup-startup-hook)
;; Set the dashboard as the default buffer
(setq initial-buffer-choice (lambda () (get-buffer "*dashboard*")))
```

7 Magit

Press `C-x g` to open magit

```
(global-set-key (kbd "C-x g") 'magit-status)
```

8 Powerline theme

```
(require 'powerline)
;; Use the vim powerline theme
(powerline-default-theme)
```

9 TRAMP

```
;; Default to ssh when using tramp
(setq tramp-default-method "ssh")
```

10 M-x Autocomplete

```
;; Fuzzy command complete on M-x
(global-set-key (kbd "M-x") 'smex)
```

11 Default theme

```
;; I like lush and use it by default
;(load-theme 'lush t)
(load-theme 'abyss t)
```

12 Chef

```
(setq org-capture-templates
      '(("c" "Cookbook" entry (file "~/org/cookbook.org")
^^I "%(org-chef-get-recipe-from-url)"
^^I :empty-lines 1)
^^I("m" "Manual Cookbook" entry (file "~/org/cookbook.org")
^^I "* %^{Recipe title: }\n  :PROPERTIES:\n  :source-url:\n  :servings:\n  :prep-time
```

13 Org mode

```
(use-package org-special-block-extras
  :ensure t
  :hook (org-mode . org-special-block-extras-mode))

;; Add the Unicode bullets package
(add-hook 'org-mode-hook (lambda () (org-superstar-mode 1)))
;; This is usually the default, but keep in mind it must be nil
(setq org-hide-leading-stars nil)
;; This line is necessary.
(setq org-superstar-leading-bullet ?\s)
;; Add the new fancy extra org mode blocks
(add-hook 'org-mode-hook #'org-special-block-extras-mode)
;; Add timestamp when marked DONE
(setq org-log-done 'time)

;; Use org-ref
(require 'org)
(require 'ox-latex)
(add-to-list 'org-latex-packages-alist '("" "minted"))
(setq org-latex-listings 'minted)

(setq org-latex-custom-lang-environments
      '(
^^I(emacs-lisp "common-lispcode")
^^I))
(setq org-latex-minted-options
      '(("frame" "lines")
^^I("fontsize" "\\scriptsize")
```

```
^^I("linenos" "")))

;; Build nonstopmode with xelatex
(setq org-latex-pdf-process
      '("xelatex -shell-escape -interaction nonstopmode -output-directory %o %b %f"
^^I"bibtex %b"
^^I"makeindex %b"
^^I"xelatex -shell-escape -interaction nonstopmode -output-directory %o %b %f"
^^I"xelatex -shell-escape -interaction nonstopmode -output-directory %o %b %f"))

(setq org-src-fontify-natively t)

(org-babel-do-load-languages
 'org-babel-load-languages
 '((R . t)
   (latex . t)))
```

14 Olivetti

```
(setq olivetti-body-width 80)
```

15 Git messenger

```
;; Press C-c c to open git-messenge
(global-set-key (kbd "C-c c") 'git-messenger:popup-message)
(custom-set-variables
 '(git-messenger:use-magit-popup t))
```

16 Emojify

```
(add-hook 'after-init-hook #'global-emojify-mode)
```

17 Ripgrep

```
(global-set-key (kbd "<f5>") #'deadgrep)
```

18 Anzu search

```
(global-anzu-mode +1)
```

19 Art bollocks

```
(autoload 'artbollocks-mode "artbollocks-mode")
(add-hook 'text-mode-hook 'artbollocks-mode)
```

20 EMMS

```
(emms-all)
(emms-default-players)
```

21 Artist

```
(put 'narrow-to-region 'disabled nil)
```

22 Which key

```
(which-key-mode)
```

23 Projectile

```
(projectile-mode +1)
(define-key projectile-mode-map (kbd "C-c p") 'projectile-command-map)
```

24 Gemini protocol

```
(add-hook 'gemini-mode-hook '(lambda () (setq fill-column 80)))
(add-hook 'gemini-mode-hook 'turn-on-auto-fill)
```

25 LaTeX in org

```
(setq org-highlight-latex-and-related '(latex script entities))
```

26 Default GUI font

```
(set-frame-font "InputMono 10" nil t)
```