# init

Martyn Jago

March 7, 2012

## Contents

**1 My Emacs Configurations** **4**
- 1.1 C-k to kill whole line . . . 4
- 1.2 Mark (select) all on S-C-SPC . . . 4
- 1.3 Map M-l to goto-line . . . 4
- 1.4 Remap M-j, M-l to navigation . . . 5
- 1.5 don't kill my window setup . . . 5
- 1.6 Changes all yes/no questions to y/n type . . . 5
- 1.7 CScope . . . 5
- 1.8 Confirm on exit . . . 5
- 1.9 Key mappings for F1 key . . . 5
- 1.10 Emacs default command modifications . . . 5
- 1.11 Setup bbdb Contact Database . . . 6
- 1.12 Visual config . . . 6
- 1.13 Disable visible bell (for removing white square on OSX . . . 6
- 1.14 Smooth Scrolling . . . 6
  - 1.14.1 web . . . 6
  - 1.14.2 initialisation . . . 6
  - 1.14.3 adjust margin to 3 lines . . . 6
- 1.15 Color theme initialisation . . . 7
  - 1.15.1 Configure color-theme . . . 7
  - 1.15.2 web . . . 7
  - 1.15.3 configuration . . . 7
  - 1.15.4 Theme Faves . . . 7
  - 1.15.5 Maximise frame at startup (hook) . . . 7
- 1.16 Inhibit scratch message . . . 8
- 1.17 Remove Paredit default from emacs-lisp mode hook . . . 8
- 1.18 Add .emacs.d/martyn/martyn to load-path . . . 8
- 1.19 my-load-init-files function . . . 8
- 1.20 my-load-drill-files function . . . 8
- 1.21 my-load-org-files function . . . 8
- 1.22 Emacs Customisation destination files . . . 8
  - 1.22.1 loaddefs . . . 9

1.22.2 custom . . . 9
1.23 Notification (todochiku / growl) . . . 9
1.24 Follow links with return . . . 9
1.25 Add org-mode contrib/lisp files to load path . . . 9
1.26 org-mode directory . . . 9
1.27 Org Agenda . . . 9
1.27.1 Org Agenda Files . . . 9
1.28 org-drill . . . 10
1.29 keywords global setup . . . 10
1.30 Lilypond mode . . . 11
1.31 Setup org-mode (development) . . . 11
1.32 Configure Babel languages . . . 11
1.32.1 For Test . . . 11
1.32.2 For production . . . 11
1.33 org-confirm-babel-evaluate . . . 12
1.34 Display images in org mode . . . 12
1.35 Org Attach Directory . . . 13
1.36 Don't insert blank lines on org heading generation . . . 13
1.37 Auto-Complete . . . 13
1.37.1 To load . . . 13
1.37.2 Add to init . . . 13
1.37.3 Screencast . . . 13
1.38 Redo+ . . . 13
1.38.1 web . . . 14
1.38.2 configuration . . . 14
1.39 Yasnippet . . . 14
1.40 HideShow . . . 14
1.40.1 Website . . . 14
1.40.2 HideShow Configuration . . . 14
1.40.3 hideshow.org . . . 15
1.40.4 To byte-compile-file . . . 15
1.41 Lusty Explorer . . . 15
1.41.1 emacswiki . . . 15
1.41.2 Github . . . 15
1.41.3 Config . . . 16
1.41.4 To byte-compile-file . . . 16
1.42 Map C-x C-o to other window (as C-x o) . . . 16
1.43 lua-mode . . . 16
1.44 Setup generic browser . . . 16
1.45 Setup MacGPG2 (for gpg) on mac OSX . . . 16
1.46 Make Password . . . 16
1.47 mj-versions . . . 17
1.48 my-org-test-setup . . . 17
1.49 my-header-arguments . . . 17
1.50 help-page . . . 17
1.51 Setup IRC user info and channels . . . 17

1.52 Boxquote . . . 17
1.53 lorem-ipsum . . . 18
1.54 magit-hide-diffs . . . 18
1.55 ispell . . . 18
1.56 ess . . . 18
1.57 **Message** buffer size . . . 18

**2 Under Test** **18**
2.1 **TODO** wdired (aliased to dired-edit) . . . 18
2.2 **TODO** plantuml . . . 18
2.3 **TODO** MPC . . . 18
2.4 **TODO** ansi-color . . . 19
2.5 **TODO** grep-scroll-output . . . 19
2.6 **TODO** mj-date-time . . . 19
2.7 **TODO** Sha1 Generate . . . 19
2.8 **TODO** debug python html export bug . . . 19
2.9 **TODO** org-export investigation . . . 20
2.10 **TODO** lower-case keywords in orgmode . . . 20
2.11 **TODO** continuous-testing . . . 20
2.12 **TODO** shen-mode . . . 20
2.13 **TODO** disable org-use-speed-commands (default) . . . 20
2.14 **TODO** Make windmove work in org-mode . . . 20
2.15 **TODO** Fix tab in org mode . . . 21
2.16 **TODO** One-touch timestamps . . . 21
2.17 **TODO** Fast TODO key selection . . . 21
2.18 **TODO** Configure org-mobile . . . 21
2.19 **TODO** my-run-drills function (doesn't work yet) . . . 21
2.20 **TODO** Setup org-capture . . . 22
2.20.1 Setup capture location and key . . . 22
2.20.2 **TODO** Setup capture templates . . . 22
2.20.3 **TODO** Setup refile targets . . . 22
2.21 **TODO** Org Agenda Custom Commands . . . 22
2.22 **TODO** Unity Mode . . . 23
2.22.1 load path and speed keys . . . 23
2.23 **TODO** unity-project () . . . 24
2.24 **TODO** unity-eval-src-and-tests () . . . 24
2.25 **TODO** unity-project-org() . . . 24
2.26 **TODO** Develop group agenda to show unscheduled TODO items 25
2.27 **TODO** Setup for latex / pdf export of org-mode . . . 25
2.28 **TODO** Setup for Auctex (ongoing) . . . 25
2.29 **TODO** Setup Clocking work time . . . 26
2.30 **TODO** Function mj-diary-org opens diary.org . . . 26
2.31 **TODO** Function mj-ng-pod-dev opens ng-pod.dev . . . 26
2.32 **TODO** Function mj-cycle-helps opens mj-helps . . . 27
2.33 **TODO** Flyspell mode error avoidance . . . 27
2.33.1 web discussion . . . 27

2.33.2 configuration . . . 27
2.34 **DEPRECATED** ERT setup . . . 27
2.35 **TODO** Graphviz-dot-mode . . . 27
2.36 **TODO** Graphviz in babel doesn't appear to be implemented . . . 28
2.37 **TODO** org-contacts . . . 28
2.37.1 **TODO** Yasnippet for rails . . . 28
2.38 **TODO** Highlight Parantheses . . . 29
2.38.1 Site . . . 29
2.38.2 Setup . . . 29
2.39 **TODO** S-C-d to backward-delete-char() - doesn't work!! . . . 29
2.40 **TODO** backward-kill-line (SHIFT CTRL K) . . . 29
2.41 **TODO** doremi-frm . . . 29
2.42 **TODO** Comment or uncomment region with M-k . . . 29
2.43 **TODO** column-number-mode . . . 29
2.44 **TODO** Remap C-j to my custom goto-next-line function . . . 30
2.45 **TODO** elpa . . . 30
2.46 **TODO** Mode-compile . . . 30
2.47 **TODO** Ruby . . . 30
2.47.1 **TODO** (needs sorting) need to run rvm-autodetect-ruby to get ruby to work in babel **:DISABLED:** . . . 30
2.47.2 Setup rspec-mode (development) . . . 30
2.48 **TODO** Kill to start of line . . . 30
2.49 **TODO** Environment Path additions . . . 31

**3 Initial display** **31**
3.1 Display diary org and agenda . . . 31
3.1.1 "martyn-laptop" system specific... . . . 31

**4 Delete temp file martyn/init.el** **31**

# 1 My Emacs Configurations

## 1.1 C-k to kill whole line

```
(setq kill-whole-line t)
```

## 1.2 Mark (select) all on S-C-SPC

```
(define-key global-map (kbd "S-C-SPC") 'mark-whole-buffer)
```

## 1.3 Map M-l to goto-line

```
(global-unset-key "\M-l")
(global-set-key "\M-l" 'goto-line)
```

### 1.4 Remap M-j, M-l to navigation

```
; (global-set-key "\M-i" 'previous-line)
(global-set-key "\M-j" 'backward-char)
; (global-set-key "\M-k" 'forward-line)
(global-set-key "\M-l" 'forward-char)
```

### 1.5 don't kill my window setup

```
(setq org-agenda-window-setup 'current-window)
```

### 1.6 Changes all yes/no questions to y/n type

```
(fset 'yes-or-no-p 'y-or-n-p)
```

### 1.7 CScope

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/cscope"))
(require 'cscope)
 (setq cscope-do-not-update-database t)
```

### 1.8 Confirm on exit

```
(setq confirm-kill-emacs 'y-or-n-p)
```

### 1.9 Key mappings for F1 key

- State "TODO" from "" *2011-03-15 Tue 15:03*

```
(global-set-key [C-f1] 'help-for-help-internal)
(global-set-key [S-C-f1] 'mj-cycle-helps)
(global-set-key [f1] 'mj-cycle-helps)
```

### 1.10 Emacs default command modifications

```
(put 'upcase-region 'disabled nil)
```

## 1.11 Setup bbdb Contact Database

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/bbdb-2.35/lisp"))
(require 'bbdb)
(bbdb-initialize)
(setq bbdb-north-american-phone-numbers-p nil
)
```

## 1.12 Visual config

## 1.13 Disable visible bell (for removing white square on OSX

Since I don't like either the beep or the flashing white square (visible bell) the following is required to silence both (from here)

```
;; (setq visible-bell nil)
;; (setq ring-bell-function nil)

(defun my-bell-function ()
;; (unless (memq this-command
;; '(isearch-abort abort-recursive-edit exit-minibuffer
;; keyboard-quit mwheel-scroll down up next-line previous-line
;; backward-char forward-char next prior))
;; (ding))
)
(setq ring-bell-function 'my-bell-function)
```

## 1.14 Smooth Scrolling

### 1.14.1 web

link

### 1.14.2 initialisation

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/smooth-scrolling"))
(require 'smooth-scrolling)
```

### 1.14.3 adjust margin to 3 lines

```
(setq smooth-scroll-margin 3)
```

## 1.15 Color theme initialisation

### 1.15.1 Configure color-theme

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/color-theme-6.6.0"))
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/color-theme-6.6.0/themes"))
(require 'color-theme)
(eval-after-load "color-theme"
  '(progn
     (color-theme-initialize)))
;; (color-theme-hober)))
;; (color-theme-tangotango)))
```

### 1.15.2 web

github]]

### 1.15.3 configuration

### 1.15.4 Theme Faves

- configuration

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn/theme-faves"))
(require 'theme-faves)
(add-hook 'window-setup-hook 'theme-faves-init)

(global-set-key [C-f7] 'theme-faves-audition-cycle-up)
(global-set-key [S-C-f7] 'theme-faves-audition-cycle-down)
(global-set-key [f7] 'theme-faves-cycle-up)
(global-set-key [S-f7] 'theme-faves-cycle-down)
```

### 1.15.5 Maximise frame at startup (hook)

```
;;(add-to-list 'load-path
;; (concat dotfiles-dir "martyn/maxframe"))
;; (require 'maxframe)
;; (add-hook 'window-setup-hook 'maximize-frame t)
;; (maximize-frame)

(defun toggle-fullscreen ()
  (interactive)
  (if (string= system-name "martyn-laptop")
```

```
  (x-send-client-message nil 0 nil "_NET_WM_STATE" 32
                         '(2 "_NET_WM_STATE_MAXIMIZED_VERT" 0))
  (x-send-client-message nil 0 nil "_NET_WM_STATE" 32
                         '(2 "_NET_WM_STATE_MAXIMIZED_HORZ" 0))
  (add-hook 'window-setup-hook 'toggle-fullscreen)))
```

## 1.16 Inhibit scratch message

```
(setq initial-scratch-message nil)
```

## 1.17 Remove Paredit default from emacs-lisp mode hook

(turned on by emacs starter kit)

```
(remove-hook 'emacs-lisp-mode-hook 'turn-on-paredit nil)
```

hs-minor-mode ert–activate-font-lock-keywords ac-emacs-lisp-mode-setup run-coding-hook esk-remov

## 1.18 Add .emacs.d/martyn/martyn to load-path

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn"))
```

## 1.19 my-load-init-files function

```
(require 'my-load-init-files)
```

## 1.20 my-load-drill-files function

```
(require 'my-load-drill-files)
```

## 1.21 my-load-org-files function

```
(require 'my-load-org-files)
```

## 1.22 Emacs Customisation destination files

These are already defined in Starter Kit but are re-adjusted here to keep under my revision control...

### 1.22.1 loaddefs

```
(setq autoload-file (concat dotfiles-dir "martyn/loaddefs.el"))
```

### 1.22.2 custom

Customisations made using emacs Customizations go here...

```
(setq custom-file (concat dotfiles-dir "martyn/custom.el"))
```

## 1.23 Notification (todochiku / growl)

```
(when (string= "darwin" system-type)
 (add-to-list 'load-path
  (concat dotfiles-dir "martyn/todochiku"))
 (require 'todochiku)
 (setq todochiku-icons-directory
  (concat dotfiles-dir "martyn/todochiku/todochiku-icons"))
 (defun mj-notify-pomodoro-done ()
  (todochiku-message "Pomodoro"
"
␣␣This␣Pomodoro␣Session␣is␣Complete

␣␣␣␣␣␣␣␣Release␣Concentration

␣␣␣␣␣␣␣␣␣␣␣␣Take␣a␣Break

" (todochiku-icon 'bell)))
  (add-hook 'org-timer-done-hook 'mj-notify-pomodoro-done))
```

## 1.24 Follow links with return

```
(setq org-return-follows-link t)
```

## 1.25 Add org-mode contrib/lisp files to load path

```
(add-to-list 'load-path "~/org-mode/contrib/lisp")
```

## 1.26 org-mode directory

```
(setq org-directory "~/.emacs-config/OrgData")
```

## 1.27 Org Agenda

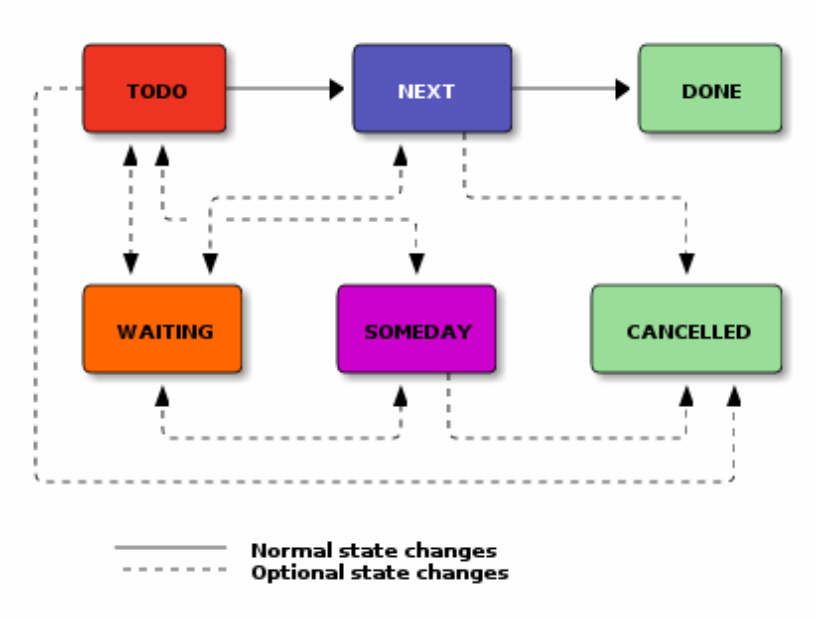
### 1.27.1 Org Agenda Files

```
(setq org-agenda-files (quote (
  "~/.emacs-config/OrgData/diary.org"
  "~/ngpod/dev/ngpod.org")))
```

## 1.28 org-drill

```
(require 'org-drill)
(setq org-learn-always-reschedule t)
(setq org-drill-use-visible-cloze-face-p t)
(setq org-drill-add-random-noise-to-intervals-p t)
(put 'scroll-left 'disabled nil)
```

## 1.29 keywords global setup




```
(setq org-todo-keywords (quote ((sequence "TODO(t!)" "|" "DONE(d!/!)")
                          (sequence "WAITING(w@/!)"
                          "SOMEDAY(s!)"
                          "IGNORED"
                          "FAIL"
                          "PASS"
                          "MISSING"
                          "DEPRECATED"
                          "|" "CANCELLED(c@/!)"))))

(setq org-todo-keyword-faces
```

```
    (quote (("TODO" :foreground "red" :weight bold)
           ("DONE" :foreground "forest␣green" :weight bold)
           ("WAITING" :foreground "yellow" :weight bold)
           ("SOMEDAY" :foreground "goldenrod" :weight bold)
           ("CANCELLED" :foreground "orangered" :weight bold)
           ("EXPIRED" :foreground "olivedrab1" :weight bold)
           ("IGNORED" :foreground "yellow" :weight bold)
           ("FAIL" :foreground "red" :weight bold)
           ("MISSING" :foreground "olivedrab1" :weight bold)
           ("PASS" :foreground "forest␣green" :weight bold)
           ("DEPRECATED":foreground "orangered" :weight bold))))
```

## 1.30 Lilypond mode

```
 (add-to-list 'load-path
           (concat dotfiles-dir "martyn/lilypond"))
(autoload 'LilyPond-mode "lilypond-mode" "LilyPond␣Editing␣Mode" t)
(add-to-list 'auto-mode-alist '("\\.ly$" . LilyPond-mode))
(add-to-list 'auto-mode-alist '("\\.ily$" . LilyPond-mode))
```

## 1.31 Setup org-mode (development)

```
(add-to-list 'auto-mode-alist '("\\.org\\'" . org-mode))
(global-set-key "\C-cl" 'org-store-link)
(global-set-key "\C-ca" 'org-agenda)
(global-set-key "\C-cb" 'org-iswitchb)
(global-font-lock-mode 1) ; for all buffers
(setq load-path (cons "~/org-mode" load-path))
```

## 1.32 Configure Babel languages

### 1.32.1 For Test

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

### 1.32.2 For production

Configured for...

- Org
- Ruby
- Python
- C

- emacs-lisp
- shell
- graphviz
- clojure
- ditaa
- lilypond

```
(org-babel-do-load-languages
 'org-babel-load-languages
 '((org . t)
   (ruby . t)
   (python . t)
   (R . t)
   (C . t)
   (emacs-lisp . t)
   (sh . t)
   (dot . t)
   (clojure . t)
   (ditaa . t)
   (lilypond . t)
   (plantuml . t)
   ))

(setq org-src-fontify-natively t) ;; color blocks
```

## 1.33 org-confirm-babel-evaluate

```
(setq org-confirm-babel-evaluate nil)
```

## 1.34 Display images in org mode

```
  ;; display images
  (local-set-key "\M-I" 'org-toggle-iimage-in-org)

;; -- Display images in org mode
;; enable image mode first
(iimage-mode)
;; add the org file link format to the iimage mode regex
(add-to-list 'iimage-mode-image-regex-alist
  (cons (concat "\\[\\[file:\\(~?" iimage-mode-image-filename-regex "\\)\\]")
1))
;; add a hook so we can display images on load
```

```
(add-hook 'org-mode-hook '(lambda () (org-turn-on-iimage-in-org)))
;; function to setup images for display on load
(defun org-turn-on-iimage-in-org ()
  "display␣images␣in␣your␣org␣file"
  (interactive)
  (turn-on-iimage-mode)
  (set-face-underline-p 'org-link nil))
;; function to toggle images in a org bugger
(defun org-toggle-iimage-in-org ()
  "display␣images␣in␣your␣org␣file"
  (interactive)
  (if (face-underline-p 'org-link)
      (set-face-underline-p 'org-link nil)
      (set-face-underline-p 'org-link t))
  (call-interactively 'iimage-mode))
```

## 1.35 Org Attach Directory

```
(setq org-attach-directory "~/.emacs-config/OrgData/attach-directory")
```

## 1.36 Don't insert blank lines on org heading generation

```
(setq org-blank-before-new-entry nil)
```

## 1.37 Auto-Complete

### 1.37.1 To load

M-x load-file RET file to load: ~/.emacs.d/martyn/auto-complete/etc/install.el
destination: ~/.emacs.d/martyn/auto-complete/installation

### 1.37.2 Add to init

```
(add-to-list 'load-path "~/.emacs.d/martyn/auto-complete/installation")
(require 'auto-complete-config)
(add-to-list 'ac-dictionary-directories "~/.emacs.d/martyn/auto-complete/installation/ac-dict")
(ac-config-default)
```

### 1.37.3 Screencast

YouTube

## 1.38 Redo+

This appears to be an upgrade of redo (although I never had any problems with redo). I shall monitor this!

### 1.38.1 web

emacswiki older-redo

### 1.38.2 configuration

```
(add-to-list 'load-path (concat dotfiles-dir "martyn/redoplus"))
(require 'redo+)
```

- Make C-z undo, and M-z redo

```
(define-key global-map (kbd "M-z") 'redo)
(define-key global-map (kbd "C-z") 'undo)
```

## 1.39 Yasnippet

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/yasnippet-read-only"))
(require 'yasnippet) ;; not yasnippet-bundle
(yas/initialize)

(setq yas/root-directory
     '(
    ;; "~/.emacs.d/martyn/martyn/unity-mode/snippets"
      "~/.emacs.d/martyn/martyn/snippets"
      "~/.emacs.d/martyn/martyn/snippets/yasnippet-org-mode-fork/snippets"
    ;; "~/.emacs.d/martyn/yasnippets-rails/rails-snippets"
      "~/.emacs.d/martyn/yasnippet-read-only/snippets"))

;; Load the snippets (NOTE: the mapc required for multiple directories)
;; (Map 'yas/load-directory' to every element)

(mapc 'yas/load-directory yas/root-directory)
```

## 1.40 HideShow

CLOSED: *2011-01-25 Tue 14:49*

### 1.40.1 Website

emacsWiki/hideshow

### 1.40.2 HideShow Configuration

```
(add-hook 'c-mode-common-hook 'hs-minor-mode)
   (add-hook 'emacs-lisp-mode-hook 'hs-minor-mode)
   (add-hook 'java-mode-hook 'hs-minor-mode)
   (add-hook 'lisp-mode-hook 'hs-minor-mode)
   (add-hook 'perl-mode-hook 'hs-minor-mode)
   (add-hook 'sh-mode-hook 'hs-minor-mode)

  (defun me-toggle-hiding-all()
  "Fast␣cycling␣of␣all␣folded␣all␣unfolded␣utilising
␣␣␣hideshow␣minor␣mode␣hs-minor-mode"
  (interactive)
  (defvar me-hideshow-active nil)
  (setq me-hideshow-active
    (if me-hideshow-active
       (progn (hs-show-all) nil)
     (progn (hs-hide-all) t))))

  ; add hook for hideshow minor mode
  (add-hook 'lisp-mode-hook 'hs-minor-mode)
  (global-set-key [f3] 'hs-toggle-hiding)
  (global-set-key [C-f3] 'me-toggle-hiding-all)
```

#### 1.40.3 hideshow.org

- Website
  github/hideshow-org
- Configuration

```
(add-to-list 'load-path (concat dotfiles-dir "martyn/hideshow-org"))
(require 'hideshow-org)
```

#### 1.40.4 To byte-compile-file

### 1.41 Lusty Explorer

CLOSED: *2011-01-25 Tue 14:49*

#### 1.41.1 emacswiki

emacswiki

#### 1.41.2 Github

lusty-emacs

### 1.41.3 Config

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/lusty-emacs"))
(require 'lusty-explorer)
(define-key global-map (kbd "C-x␣C-d") 'lusty-file-explorer)
```

### 1.41.4 To byte-compile-file

NOTE: lusty-explorer.el doesn't want to compile

## 1.42 Map C-x C-o to other window (as C-x o)

CLOSED: *2011-01-26 Wed 10:41*

```
(global-set-key [C-x-o] 'other-window)
```

## 1.43 lua-mode

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/lua-mode/testing/lisp"))
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/lua-mode/lisp"))
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/lua-mode/testing/examples"))
(autoload 'lua-mode "lua-mode" "Lua␣editing␣mode." t)
(add-to-list 'auto-mode-alist '("\\.lua$" . lua-mode))
(add-to-list 'interpreter-mode-alist '("lua" . lua-mode))
(setq lua-default-application "/opt/local/bin/lua")
```

## 1.44 Setup generic browser

```
(if (string= "darwin" system-type)
   (setq browse-url-browser-function 'browse-url-generic
     browse-url-generic-program "open")
  (setq browse-url-browser-function 'browse-url-generic
   browse-url-generic-program "conkeror"))
```

## 1.45 Setup MacGPG2 (for gpg) on mac OSX

```
(when (string= "darwin" system-type)
 (setq epg-gpg-program "/usr/local/MacGPG2/bin/gpg2")
 (message "MacGPG2␣set␣up"))
```

## 1.46 Make Password

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/make-password"))

(require 'make-password)
(defalias 'mj-generate-password 'make-password)
(defalias 'generate-password 'make-password)
```

## 1.47 mj-versions

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn/mj-versions"))
(require 'mj-versions)
```

## 1.48 my-org-test-setup

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn/my-org-test-setup"))
(require 'my-org-test-setup)
```

## 1.49 my-header-arguments

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn/my-header-arguments"))
(require 'my-header-arguments)
```

## 1.50 help-page

```
(add-to-list 'load-path
          (concat dotfiles-dir "martyn/martyn/help-page"))
(require 'help-page)
  (defalias 'hp 'help-page)
```

## 1.51 Setup IRC user info and channels

```
(setq erc-header-line-format nil)
(setq erc-hide-list '("JOIN" "PART" "QUIT"))

(setq erc-log-channels-directory t)
(setq erc-log-file-name-function "~/erc")
(setq erc-truncate-buffer-on-save t)
(setq erc-nick "mjago")
```

## 1.52 Boxquote

```
(add-to-list 'load-path
 (concat dotfiles-dir "martyn/boxquote"))
(require 'boxquote)
```

### 1.53 lorem-ipsum

```
(add-to-list 'load-path
 (concat dotfiles-dir "martyn/lorem-ipsum"))
(require 'lorem-ipsum)
```

### 1.54 magit-hide-diffs

```
(setq magit-hide-diffs t)
```

### 1.55 ispell

```
(setq ispell-program-name "/opt/local/bin/ispell")
(require 'ispell)
```

### 1.56 ess

```
(add-to-list
 'load-path
  (concat dotfiles-dir "martyn/ess-5.14/lisp/"))
(require 'ess-site)
```

### 1.57 Message buffer size

```
(setq message-log-max 2000)
```

## 2 Under Test

### 2.1 TODO wdired (aliased to dired-edit)

```
(defalias 'dired-edit 'wdired-change-to-wdired-mode)
```

### 2.2 TODO plantuml

```
(setq org-plantuml-jar-path "~/plantuml/plantuml.jar")
```

### 2.3 TODO MPC

```
(setq mpc-mode-map
  (let ((map (make-keymap)))
    (suppress-keymap map)
    ;; (define-key map "\e" 'mpc-stop)
```

```
  (define-key map "q" 'mpc-quit)
  (define-key map "\r" 'mpc-select)
  (define-key map [(shift return)] 'mpc-select-toggle)
  (define-key map [mouse-2] 'mpc-select)
  (define-key map [S-mouse-2] 'mpc-select-extend)
  (define-key map [C-mouse-2] 'mpc-select-toggle)
  (define-key map [drag-mouse-2] 'mpc-drag-n-drop)
  ;; We use `always' because a binding to t is like a binding to nil.
  (define-key map [follow-link] 'always)
  ;; Doesn't work because the first click changes the buffer, so the second
  ;; is applied elsewhere :-(
  ;; (define-key map [(double mouse-2)] 'mpc-play-at-point)
  (define-key map "x" 'mpc-play)
  (define-key map "X" 'mpc-play-at-point)
  (define-key map "s" 'mpc-pause)
  (define-key map "S" 'mpc-stop)
  (define-key map "n" 'mpc-next)
  (define-key map "p" 'mpc-prev)
  (define-key map "f" 'mpc-ffwd)
  (define-key map "b" 'mpc-rewind)
  map))
(defalias 'music 'mpc)
```

## 2.4 TODO ansi-color

```
(add-to-list
 'load-path
  (concat dotfiles-dir "martyn/ansi-color"))
(require 'ansi-color)
```

## 2.5 TODO grep-scroll-output

```
(setq grep-scroll-output t)
```

## 2.6 TODO mj-date-time

```
(defun mj-date-time()
  (interactive)
  (message "%S" (shell-command "date" t )))
```

## 2.7 TODO Sha1 Generate

Generate sha1 of selected region

```
(load-file (concat dotfiles-dir "martyn/martyn/sha1/sha1.el"))
```

## 2.8 TODO debug python html export bug

```
(setq org-babel-python-command "python2.6")
```

## 2.9 TODO org-export investigation

```
(require 'org-export)
```

## 2.10 TODO lower-case keywords in orgmode

ref: Bernt Hansen (Re: Capitalisation and good taste ?)

```
(setq org-babel-results-keyword "results")

(setq org-structure-template-alist
     (quote (("s" "#+begin_src␣?\n\n#+end_src" "<src␣lang=\"?\">\n\n</src>")
            ("e" "#+begin_example\n?\n#+end_example" "<example>\n?\n</example>")
            ("q" "#+begin_quote\n?\n#+end_quote" "<quote>\n?\n</quote>")
            ("v" "#+begin_verse\n?\n#+end_verse" "<verse>\n?\n/verse>")
            ("c" "#+begin_center\n?\n#+end_center" "<center>\n?\n/center>")
            ("l" "#+begin_latex\n?\n#+end_latex" "<literal␣style=\"latex\">\n?\n</literal>")
            ("L" "#+latex:␣" "<literal␣style=\"latex\">?</literal>")
            ("h" "#+begin_html\n?\n#+end_html" "<literal␣style=\"html\">\n?\n</literal>")
            ("H" "#+html:␣" "<literal␣style=\"html\">?</literal>")
            ("a" "#+begin_ascii\n?\n#+end_ascii")
            ("A" "#+ascii:␣")
            ("i" "#+index:␣?" "#+index:␣?")
            ("I" "#+include␣%file␣?" "<include␣file=%file␣markup=\"?\">"))))
```

## 2.11 TODO continuous-testing

- State "TODO" from "" *2011-12-17 Sat 15:24*

```
(load-file (concat dotfiles-dir "martyn/martyn/continuous-testing/lib/continuous-testing.el"))
```

## 2.12 TODO shen-mode

```
(add-to-list
 'load-path
  (concat dotfiles-dir "martyn/shen-mode"))
```

## 2.13 TODO disable org-use-speed-commands (default)

```
(setq org-use-speed-commands nil)
```

## 2.14 TODO Make windmove work in org-mode

```
(add-hook 'org-shiftup-final-hook 'windmove-up)
(add-hook 'org-shiftleft-final-hook 'windmove-left)
(add-hook 'org-shiftdown-final-hook 'windmove-down)
(add-hook 'org-shiftright-final-hook 'windmove-right)
```

; Targets complete in steps so we start with filename, TAB shows the next level of targets etc (setq org-outline-path-complete-in-steps t)

; Allow refile to create parent tasks with confirmation (setq org-refile-allow-creating-parent-nodes (quote confirm))

; Use IDO only for buffers ; set ido-mode to buffer and ido-everywhere to t via the customize interface ; '(ido-mode (quote both) nil (ido)) ; '(ido-everywhere t)

## 2.15 TODO Fix tab in org mode

Not sure this is of any benefit anymore!

```
;; fix tab
(local-set-key "\C-y" 'yank)
```

## 2.16 TODO One-touch timestamps

```
(setq org-agenda-skip-additional-timestamps nil)
(defun mj-one-touch-timestamp ()
  (interactive)
  (when (eq major-mode 'org-mode)
    (org-insert-time-stamp nil t t)))
(define-key global-map (kbd "<f8>") 'mj-one-touch-timestamp)
```

## 2.17 TODO Fast TODO key selection

```
(setq org-treat-S-cursor-todo-selection-as-state-change nil)
```

## 2.18 TODO Configure org-mobile

```
(setq org-mobile-directory "~/.emacs-config/MobileOrg")
(setq org-mobile-files org-agenda-files)
(setq org-mobile-inbox-for-pull "~/.emacs-config/OrgData/inbox.org")
```

## 2.19 TODO my-run-drills function (doesn't work yet)

```
(defun my-run-drills ()
"Function␣to␣run␣all␣drills␣that␣exist␣in␣agenda␣files"
 (interactive)
 (org-drill 'agenda))

(provide 'my-run-drills)
```

## 2.20 TODO Setup org-capture

### 2.20.1 Setup capture location and key

```
(setq org-default-notes-file "~/.emacs-config/OrgData/capture.org")
(define-key global-map "\C-cc" 'org-capture)

(setq org-capture-templates
     '(("t" "Agenda␣Todo" entry
        (file+datetree+prompt "~/.emacs-config/OrgData/diary.org")
;; "\n\n** TODO %?\n%T\n\n%i\n%a\n\n\n"
        "\n\n**␣TODO␣%?\n%t\n\n"
        :empty-lines 1)))

;; ("n" "Agenda Notes" entry
;; (file+headline "~/.emacs-config/OrgData/diary.org" "Agenda")
;; "\n\n** %?\n%T\n%i\n%a\n\n\n"
;; :empty-lines 1))
```

### 2.20.2 TODO Setup capture templates

### 2.20.3 TODO Setup refile targets

```
; Use IDO for target completion
(setq org-completion-use-ido t)

; Targets include this file and any file contributing to the agenda - up to 5 levels deep
(setq org-refile-targets (quote ((org-agenda-files :maxlevel . 5) (nil :maxlevel . 5))))

; Targets start with the file name - allows creating level 1 tasks
;; TESTING (setq org-refile-use-outline-path (quote file))
```

## 2.21 TODO Org Agenda Custom Commands

```
(defun run-todo-unscheduled ()
"agenda␣of␣unscheduled␣todos"
(interactive)
(setq org-agenda-todo-ignore-scheduled t)
(org-todo-list '("t"))
(setq org-agenda-todo-ignore-scheduled nil))

(provide 'run-todo-unscheduled)

 (setq org-agenda-custom-commands
        '(("e" todo "-SCHEDULED")
          ("W" todo-tree "WAITING")
          ("u" tags "+boss-urgent")
          ("v" tags-todo "+boss-urgent")
```

```
          ("U" tags-tree "+boss-urgent")
          ("f" occur-tree "\\<FIXME\\>")
          ("hl" tags "+home+Lisa")
          ("hp" tags "+home+Peter")
          ("hk" tags "+home+Kim")))

(setq org-agenda-custom-commands
     (quote (("w" "Tasks␣waiting␣on␣something" tags "WAITING/!"
            ((org-use-tag-inheritance nil)
             (org-agenda-todo-ignore-scheduled nil)
             (org-agenda-todo-ignore-deadlines nil)
             (org-agenda-todo-ignore-with-date nil)
             (org-agenda-overriding-header "Waiting␣Tasks")))
           ("r" "Refile␣New␣Notes␣and␣Tasks" tags "LEVEL=1+REFILE"
            ((org-agenda-todo-ignore-with-date nil)
             (org-agenda-todo-ignore-deadlines nil)
             (org-agenda-todo-ignore-scheduled nil)
             (org-agenda-overriding-header "Tasks␣to␣Refile")))
           ("N" "Notes" tags "NOTE"
            ((org-agenda-overriding-header "Notes")))
           ("p" "Projects" tags-todo "LEVEL=2-REFILE|LEVEL=1+REFILE/!-DONE-CANCELLED"
            ((org-agenda-skip-function 'bh/skip-non-projects)
             (org-agenda-overriding-header "Projects")))
           ("o" "Other␣(Non-Project)␣tasks" tags-todo "LEVEL=2-REFILE|LEVEL=1+REFILE/!-DONE-CANCELLED"
            ((org-agenda-skip-function 'bh/skip-projects)
             (org-agenda-overriding-header "Other␣Non-Project␣Tasks")))
           ("A" "Tasks␣to␣be␣Archived" tags "LEVEL=2-REFILE/DONE|CANCELLED"
            ((org-agenda-overriding-header "Tasks␣to␣Archive")))
           ("h" "Habits" tags "STYLE=\"habit\""
            ((org-agenda-todo-ignore-with-date nil)
             (org-agenda-todo-ignore-scheduled nil)
             (org-agenda-todo-ignore-deadlines nil)
             (org-agenda-overriding-header "Habits")))
           ("#" "Stuck␣Projects" tags-todo "LEVEL=2-REFILE|LEVEL=1+REFILE/!-DONE-CANCELLED"
            ((org-agenda-skip-function 'bh/skip-non-stuck-projects)
             (org-agenda-overriding-header "Stuck␣Projects")))
           ("c" "Select␣default␣clocking␣task" tags "LEVEL=2-REFILE"
            ((org-agenda-skip-function
              '(org-agenda-skip-subtree-if 'notregexp "^\\*\\*␣Organization"))
             (org-agenda-overriding-header "Set␣default␣clocking␣task␣with␣C-u␣C-u␣I"))))))
```

## 2.22 TODO Unity Mode

### 2.22.1 load path and speed keys

```
(add-to-list
 'load-path
  (concat dotfiles-dir "martyn/martyn/unity-mode"))
(require 'unity-mode)
```

```
;(add-hook 'c-mode-hook 'unity-mode)

(global-set-key [f5] 'unity-find-and-open-a-primative-match-forward)
(global-set-key [S-f5] 'unity-find-and-open-a-primative-match-reverse)
(global-set-key [C-f5] 'unity-cycle-MCH-buffer)
(global-set-key [f6] 'unity-cycle-alpha-ascending)
(global-set-key [S-f6] 'unity-cycle-alpha-descending)
```

## 2.23 TODO unity-project ()

```
 (defun unity-project ()
  (interactive)
  (find-file "~/.emacs.d/martyn/martyn/unity-mode/unity-parse-project-file.el")
  (find-file "~/.emacs.d/martyn/martyn/unity-mode/unity-auto-config.el")
  (delete-other-windows)
  (split-window-horizontally)
  (windmove-right)
  (find-file "~/.emacs.d/martyn/martyn/unity-mode/unity-mode.el")
  (windmove-left)
;; (find-file "~/.emacs.d/martyn/martyn/unity-mode/unity-mode-tests.el")
  (switch-to-buffer "#emacs")
  (split-window-vertically)
  (switch-to-buffer "*ert*")
  (windmove-down)
  (switch-to-buffer "#emacs"))
```

## 2.24 TODO unity-eval-src-and-tests ()

```
(defun unity-eval-src-and-tests ()
 (interactive)
 (let ((original-buffer buffer-file-name)
      (original-window (selected-window)))

  (if(string-match "^unity-" (file-name-nondirectory original-buffer))
     (progn
       (eval-buffer "unity-mode-tests.el")
       (eval-buffer "unity-mode.el")
       (ert t)))
  (select-window original-window)))
;; (error "%s" (windmove-find-other-window 'right))
;; (if(equal original-buffer (windmove-find-other-window 'right))
;; (windmove-right)
;; (windmove-down))))
```

## 2.25 TODO unity-project-org()

Moved to unity-mode.el

```
; (global-set-key [f4] 'unity-eval-src-and-tests)
; (global-set-key [C-f4] 'unity-switch-src-control-file)
; (global-set-key [C-c C-v] 'eval-region)
```

## 2.26 TODO Develop group agenda to show unscheduled TODO items

Suggestions

## 2.27 TODO Setup for latex / pdf export of org-mode

1. NOTE: remember ubuntu requires texlive-latex-extra for this
2. NOTE: also required ttf-marvosym
3. For missing packages try searching here and apt-get install package
4. Useful latex prettifying link

```
;; remember ubuntu requires texlive-latex-extra for this

(require 'org-latex)
(setq org-export-latex-listings t)
(add-to-list 'org-export-latex-packages-alist '("" "listings"))
(add-to-list 'org-export-latex-packages-alist '("" "color"))
```

## 2.28 TODO Setup for Auctex (ongoing)

;; Setup gleaned from here ;; #+begin$_{src}$ emacs-lisp :results silent ;; ;; ;; AUCTeX ;; (setq TEX-auto-save t) ;; (setq TEX-parse-self t) ;; ;set up AUCTeX to deal with multiple file documents. ;; (setq-default TEX-master nil) ;; ;turn on pdf-mode. AUCTeX will call pdflatex to compile instead of latex. ;; (add-hook 'LaTeX-mode-hook 'TeX-PDF-mode) ;; ;turn on flyspell mode - this will automatically spell check the document in LaTex-mode ;; (add-hook 'LaTeX-mode-hook 'flyspell-mode) ;; ;; ; allow for export=>beamer by placing ;; ;; #+LATEX$_{C}LASS$ : $beamerinorgfiles;;(unless(boundp`org-export-latex-classes);;(setqorg-export-latex-classesnil));;(add-to-list`org-export-latex-classes;;;;beamerclass, forpresentations;;`("beamer";;"documentclass[11pt]beamer;;mode<{{beamermode}}>;;usetheme{{beamertheme}};;usecolortheme{{beamercolortheme}};;beamertemplateballitem;;setbeameroptionshownotes;;usepackage[utf8]inputenc;;usepackage[T1]fontenc;;$

$usepackagehyperref;;$
$usepackagecolor;;$
$usepackagelistings;;$
$lstsetnumbers = none, language = [ISO]C++, tabsize = 4,;;frame = single,;;basicstyle = small,;;shows$
$usepackageverbatim;;$
$institute\{\{beamerinstitute\}\};;$
$subject\{\{beamersubject\}\}'';;org-beamer-sectioning;;(“$
$section;;(“beginframe[fragile]frametitle;;“endframe'';;“beginframe[fragile]frametitle;;“endframe'');;$
2” ;; “#+COLUMNS: %35ITEM %10BEAMER$_{\text{env}}$(Env) %10BEAMER$_{\text{envargs}}$(Env
Args) %4BEAMER$_{\text{col}}$(Col) %8BEAMER$_{\text{extra}}$(Extra)” ;; “#+OPTIONS: tags:nil”
;; “#+MACRO: BEAMERINSTITUTE My Affiliation” ;; “#+AUTHOR:
Tudor-Ioan Salomie” ;; “” ;; “#+TITLE: Presentation title” ;; “” ;; “” ;; “* My
Section” ;; “” ;; “** Frame 1

with a subtitle” ;; “” ;; “*** Idea” ;; “” ;; “#+begin$_{LaTeX}$'';;“'';;“# +
$end_{LaTeX}$'';;“'';;);;;;;;#+$end_{\text{src}}$ ;; ;; ;; : orgmode-beamer-my-skeleton ;;

## 2.29 TODO Setup Clocking work time

```
(setq org-clock-persist 'history)
    (org-clock-persistence-insinuate)
```

## 2.30 TODO Function mj-diary-org opens diary.org

```
(defun mj-diary-org()
  "Fast␣route␣to␣diary.org"
  (interactive)
  (find-file "~/.emacs-config/OrgData/diary.org"))
```

## 2.31 TODO Function mj-ng-pod-dev opens ng-pod.dev

```
(defun mj-ng-pod-dev()
  "Fast␣route␣to␣ng-pod.dev"
  (interactive)
  (find-file "~/ngpod/dev/ng-pod.dev"))


(defun mj-init-org()
  "Fast␣route␣to␣init.org"
  (interactive)
  (find-file "~/.emacs.d/martyn/init.org"))


(defun mj-org-agenda ()
  (interactive)
  (org-agenda nil "a"))
```

## 2.32 TODO Function mj-cycle-helps opens mj-helps

```
(defun mj-cycle-helps()
  (interactive)
  (unless (boundp 'mj-current-help)
    (setq mj-current-help nil))
(cond
     ((equal (buffer-name) "diary.org")
      (bury-buffer "diary.org")
      (setq mj-current-help 'mj-ng-pod-dev))
     ((equal (buffer-name) "ng-pod.dev")
      (bury-buffer "ng-pod.dev")
      (setq mj-current-help 'mj-org-agenda))
     ((equal (buffer-name) "*Org␣Agenda*")
      (org-agenda-quit)
      (setq mj-current-help 'help-page))
     ((equal (buffer-name) "help.org")
      (bury-buffer "help.org")
      (setq mj-current-help 'mj-init-org))
     (t (setq mj-current-help 'mj-diary-org)))
  (message "%S" mj-current-help)
(funcall mj-current-help))
```

## 2.33 TODO Flyspell mode error avoidance

### 2.33.1 web discussion

stackoverflow launchpad

### 2.33.2 configuration

;; (require ‘ispell) ;; (setq-default ispell-program-name “/usr/local/lib/aspell-0.60.6/aspell”) ;; (setq ispell-really-aspell t) ;; (setq ispell-dictionary “/usr/local/lib/aspell6-en-7.1-0/english”) ;; ;;# sudo rm /usr/share/emacs/site-lisp/dictionaries-common/debian-ispell.el ;;# sudo rm /usr/share/emacs/site-lisp/dictionaries-common/flyspell.el ;;# sudo rm /usr/share/emacs/site-lisp/dictionaries-common/ispell.el ;;# cd /usr/share/emacs23/site-lisp/dictionaries-common ;;# sudo rm *.el *.elc ;; the above didn’t work for me so...

(unload-feature ‘flyspell-mode) ;;neither did this! (not loaded)

##+end$_{\text{src}}$

## 2.34 DEPRECATED ERT setup

## 2.35 TODO Graphviz-dot-mode

```
(load-file (concat dotfiles-dir "martyn/graphviz-dot-mode/graphviz-dot-mode.el"))
```

## 2.36 TODO Graphviz in babel doesn't appear to be implemented

## 2.37 TODO org-contacts

```
(require 'org-contacts)

 (custom-set-variables
 '(org-contacts-files '("~/.emacs-config/OrgData/contacts.org")))

;; ("c" "Contacts" entry (file "~/Org/contacts.org")
;; "* %(org-contacts-template-name)
;; :PROPERTIES:
;; :EMAIL: %(org-contacts-template-email)
;; :END:")
```

### 2.37.1 TODO Yasnippet for rails

```
;; (add-to-list 'load-path
;; (concat dotfiles-dir "/martyn/yasnippets-rails"))

;; (add-hook 'ruby-mode-hook ; or rails-minor-mode-hook ?
;; '(lambda ()
;; (make-variable-buffer-local 'yas/trigger-key)
;; (setq yas/trigger-key [tab])))

;; (require 'yasnippet)
;; ;;(add-to-list 'yas/extra-mode-hooks
;; ;; 'ruby-mode-hook)

;; (yas/initialize)
;; (setq yas/window-system-popup-function 'yas/x-popup-menu-for-template)

;; ;;(yas/load-directory
;; ;; (concat
;; ;; dotfiles-dir "/martyn/snippets/yasnippets-rails/rails-snippets/"))
;; ;;(make-variable-buffer-local 'yas/trigger-key)

;; ;; yasnippet-org-mode
;; (yas/load-directory
;; (concat
;; dotfiles-dir "/martyn/snippets/yasnippet-org-mode/"))

  ;; yasnippet (allow yasnippet to do it's thing in org files)
;; (make-variable-buffer-local 'yas/trigger-key)
;; (setq yas/trigger-key [tab])
;; (define-key yas/keymap [tab] 'yas/next-field-group)))
```

## 2.38 TODO Highlight Parantheses

### 2.38.1 Site

site

### 2.38.2 Setup

```
(add-to-list 'load-path (concat dotfiles-dir
 "martyn/highlight-parentheses"))
(require 'highlight-parentheses)
```

## 2.39 TODO S-C-d to backward-delete-char() - doesn't work!!

```
(global-set-key [C-S-d] 'backward-delete-char)
```

## 2.40 TODO backward-kill-line (SHIFT CTRL K)

```
(defun mj-backward-kill-line ()
 (interactive)
 (if (bolp)
    (progn (previous-line)
          (org-kill-line))
  (let ((here (point)))
    (beginning-of-line)
    (kill-region (point) here))))

(global-set-key (kbd "C-M-k") 'mj-backward-kill-line)
```

## 2.41 TODO doremi-frm

## 2.42 TODO Comment or uncomment region with M-k

```
(global-set-key "\M-k" 'comment-or-uncomment-region)
```

## 2.43 TODO column-number-mode

Re: emacs and gnome-terminal (Anonymous) 2008-05-20 04:51 am UTC (link)
Not a bug fix, but this seems to make the problem go away.

Turn on column-number-mode.

You can put the following into your ~/.emacs to have it on on loading

```
(setq-default column-number-mode t)
```

```
(setq cursor-type 'hbar)
(set-cursor-color "white")
```

## 2.44 TODO Remap C-j to my custom goto-next-line function

```
;; function here
```

## 2.45 TODO elpa

;;#+begin$_{src}$ emacs-lisp :results silent ;;(setq package-user-dir (concat dotfiles-dir "martyn/elpa")) ;;#+end$_{src}$

## 2.46 TODO Mode-compile

Setup Mode-compile (emacs-wiki)

```
 ;;mode-compile
 (autoload 'mode-compile "mode-compile"
  "Command␣to␣compile␣current␣buffer␣file␣based␣on␣the␣major␣mode" t)
;; (global-set-key "\C-cc" 'mode-compile)
 (autoload 'mode-compile-kill "mode-compile"
  "Command␣to␣kill␣a␣compilation␣launched␣by␣`mode-compile'" t)
;; (global-set-key "\C-ck" 'mode-compile-kill)
```

## 2.47 TODO Ruby

### 2.47.1 TODO (needs sorting) need to run rvm-autodetect-ruby to get ruby to work in babel :DISABLED:

```
(add-to-list 'load-path (concat dotfiles-dir "martyn/rvm/rvm.el/"))
(require 'rvm)
(rvm-activate-corresponding-ruby)
(rvm-autodetect-ruby)
(rvm-use "1.9.2" "default")
```

### 2.47.2 Setup rspec-mode (development)

```
(add-to-list 'load-path (concat dotfiles-dir "martyn/rspec-mode"))
```

## 2.48 TODO Kill to start of line

```
(defun mj-kill-to-start-of-line ()
"kill␣from␣point␣to␣start␣of␣line"
(interactive)
(kill-line 0)
)
(global-set-key [C-j] 'mj-kill-to-start-of-line)
```

### 2.49 TODO Environment Path additions

```
; sane path
(setq path "/bin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/mysql/bin:/usr/texbin/")
(setenv "PATH" path)
```

# 3 Initial display

## 3.1 Display diary org and agenda

### 3.1.1 "martyn-laptop" system specific...

- NOTE: Currently loads ALL org files - might consider just "tracked" org files
- Switch to diary.org
- Bring up schedule agenda in other window

```
(defun me-startup-in-agenda ()
  (if (string= system-name "martyn-laptop")
      (progn
        (split-window-horizontally)
        (my-load-org-files)
        (switch-to-buffer "diary.org")
        (org-agenda-list))))

(add-hook 'window-setup-hook 'me-startup-in-agenda)
```

# 4 Delete temp file martyn/init.el

```
(delete-file "~/.emacs.d/martyn/init.el")
```

```
(setq org-export-directory "~/.emacs-config/OrgData/html")
```