Emacs Configuration

Zeno

Personal Information

```
(setq user-full-name "Adrian Eichelbaum"
      user-mail-address "ad.eichelbaum@gmail.com")
```

UI Customization

Sets a custom scratch-buffer message and disables some UI elements to make emacs more minimal in appearence.

```
(setq initial-scratch-message "Welcome back, Zeno")

(scroll-bar-mode -1)        ; Disable visible scrollbar
(tool-bar-mode -1)          ; Disable the toolbar
(tooltip-mode -1)           ; Disable tooltips
(set-fringe-mode 10)        ; Give some breathing room
(menu-bar-mode -1)          ; Disable the menu bar
```

Set the cursor to a bar

```
(setq-default cursor-type 'bar)
```

Line Numbers

Show line numbers, highlight current line

```
(global-hl-line-mode)                ; Highlight current line
(global-display-line-numbers-mode t) ; Show line numbers
```

Hife line numbers in certain modes. To add a mode, add the corrosponding mode-hook to the `shell-mode-hook-list`

```
(dolist (mode '(shell-mode-hook))
(add-hook mode (lambda() (display-line-numbers-mode 0))))
```

Set font and theme

```
(set-face-attribute 'default nil :font "Tamzen 12")
(load-theme 'doom-horizon t)
```

Folder Setup

Adds the custom folder to the load-path. This is where custom emacs addons are stored, if not installed with the package manager.

```
(add-to-list 'load-path "~/.emacs.d/custom")
```

Changes, so that custom config snippets generated by emacs get saved in the custom.el file, so they dont clutter up the config file.

```
;; write custom configs to seperate file
(setq custom-file (expand-file-name "custom.el" user-emacs-
        directory))
(load custom-file)
```

Changes the location where backupfiles and autosave files are stored, so they dont clutter up the working directory.

```
;; Keep all backup and auto-save files in one directory
(setq backup-directory-alist '(("." . "~/.emacs.d/backups")))
(setq auto-save-file-name-transforms '((".*" "~/.emacs.d/auto-save-
        list/" t)))
```

QOL Changes

Delete the selected region, when typing over it, like on expects

```
(delete-selection-mode t)
```

Set the escape key to quit prompts

```
(global-set-key (kbd "<escape>") 'keyboard-escape-quit)
```

Close on buffer-kill also the window

```
(substitute-key-definition 'kill-buffer 'kill-buffer-and-window
        global-map)
```

Split buffers vertically

```
(setq split-height-threshold nil)
(setq split-width-threshold 0)
```

Packages

Custom splash-screen

```
(require 'splash-screen)
```

Change theme based on time

Set your location, to get sunrise and sunset

```
(setq calendar-location-name "Leizpig, DE")
(setq calendar-latitude 51.33)
(setq calendar-longitude 12.37)
```

Load the package and set the theme based on time. The first argument is day, the second the theme night.

```
(require 'theme-changer)
(change-theme 'doom-horizon 'doom-horizon)
```

Move buffers around

Install package through use-package and define keybindings in the `:bind`-section

```
(use-package buffer-move
  :bind (
    ("<C-S-up>" . buf-move-up)
    ("<C-S-down>" . buf-move-down)
    ("<C-S-left>" . buf-move-left)
    ("<C-S-right>" . 'buf-move-right)
    )
  )
```

Doom-themes

Install using use-package. In the `:config`-part you can define config options for the package. Enables a custom neotree/teemacs theme and corrects the org-modes native fontification. For this all-the-icons must be installed

```
(use-package doom-themes
  :config
```

```
  (setq doom-themes-enable-bold t    ; if nil, bold is universally
        disabled
  doom-themes-enable-italic t) ; if nil, italics is universally
        disabled
  (doom-themes-neotree-config)
  (setq doom-themes-treemacs-theme "doom-colors")
  (doom-themes-treemacs-config)
  (doom-themes-org-config))
```

Doom-modeline

Change the modeline to the one of doom emacs, because I think it is more beautiful.

```
(use-package doom-modeline
  :init (doom-modeline-mode 1)
  :custom ((doom-modeline-height 15)))
```

Rainbow-delimiters

Changes paranthesis to dfferent colors, corosponding paranthesis have the same color. It is enabled in every `prog-mode`, which means, it is activated in every mode, which is classified as a programming mode

```
(use-package rainbow-delimiters
  :hook (prog-mode . rainbow-delimiters-mode))
```

Markdown

Load the markdown-mode, when files with certain extensions are opened. To activate this mode on further files extensions, just add the fitting regex to the array.

```
(use-package markdown-mode
  :mode (("README\\.md\\'" . gfm-mode)
   ("\\.md\\'" . markdown-mode)
   ("\\.markdown\\'" . markdown-mode))
  :init (setq markdown-command "multimarkdown"))
```

ORG-Mode

Initialize org-mode/Set org keybindings

Activate the package on certain file extensions. If configs are to be added, just add the `:config` section to the use-package section.

`org-store-link` is used to store a link to the current location and cna later be inserted into another org file using `org-link-insert`. `org-agenda` opens a window, where one can specify a org-agenda command `org-capture` will let you select a template and then file the newlu captured information. The text is immediately inserted at the target location.

```
(use-package org
  :mode (("\\.org$" . org-mode))
  :bind(
    ("C-c l" . 'org-store-link)
    ("C-c a" . 'org-agenda)
    ("C-c c" . 'org-capture)
    )
  )
```

Org-bullets

Change the headline bullet points to icons, is pretty

```
(use-package org-bullets
  )
(add-hook 'org-mode-hook (lambda () (org-bullets-mode 1)))
```

Ox-pandoc

Add pandoc export to org files.

```
(use-package ox-pandoc
  :defer 10)
```

Ivy/Counsel

An completion framework for promps with fuzzyfinding and highlighting and more information.

```
(use-package ivy
  :diminish
  :bind  (("C-s" . swiper)
   :map ivy-minibuffer-map
   ("TAB" . ivy-alt-done)
   ("C-l" . ivy-alt-done)
   ("C-j" . ivy-next-line)
   ("C-k" . ivy-previous-line)
   :map ivy-switch-buffer-map
   ("C-k" . ivy-previous-line)
   ("C-l" . ivy-done)
   ("C-d" . ivy-switch-buffer-kill)
   :map ivy-reverse-i-search-map
```

```
   ("C-k" . ivy-previous-line)
   ("C-d" . ivy-reverse-i-search-kill))
  )
(ivy-mode 1)
```

Remap keys to use Ivy instead of the default

```
(use-package counsel
  :bind  (("M-x" . counsel-M-x)
   ("C-x b" . counsel-ibuffer)
   ("C-x C-f" . counsel-find-file)
   :map minibuffer-local-map
   ("C-r" . 'counsel-minibuffer-history)))
```

Ivy-rich provides a more friendly interface for Ivy

```
(use-package ivy-rich
  :init
  (ivy-rich-mode 1))
```