# Contents



# EMACS BASIC CONFIGURATIONS

**Table of Contents**



## CONCEPTS

The following configurations for my `Emacs` are based on `spacemecas` configurations. Now, everything works flawlessly.

## Quick way to navigate between buffers

| keybinding | Description |
|---|---|
| SPC TAB or SPC b b: | This keybinding shows a list of open buffers in the current frame. You can navigate through the list using the arrow keys and press RET (Enter) to switch to the selected buffer. |
| SPC b n or SPC b p: | These keybindings allow you to cycle forward (SPC b n) or backward (SPC b p) through the open buffers. Press the respective keybinding multiple times to switch to different buffers. |
| SPC b 0 to SPC b 9: | These keybindings allow you to quickly switch to buffers numbered from 0 to 9. For example, pressing SPC b 1 will switch to the first buffer in the buffer list. |
| SPC b d : | This keybinding shows a list of recently visited buffers. You can navigate through the list using the arrow keys and press RET (Enter) to switch to the selected buffer. |

## Tips and Tricks

- You can use either `\`` or'' to repeat last command you used in your given buffer.

## Configurations with SpaceEmacs

Here is the list of things that I change

- Working with lisp files (barfage, slurpage, & more) ' | Key | Binding | Description | |———–|———————————————|———–-| | SPC m g g | go to definition of symbol under point | | | SPC m h h | describe symbol at point | | | SPC m c c | byte compile the current file | | | SPC m c l | popup compile-log buffer | | | SPC m e $ | or SPC m e l go to end of current line and evaluate | | | SPC m e b | evaluate current buffer | | | SPC m e c | evaluate current form (start with defun, setq, etc. . . ) | | | SPC m e e | evaluate sexp before point | | | SPC m e r | evaluate current region | | | SPC m e f | evaluation current function | | | SPC m | toggle lisp state | | | SPC m t b | run tests of current buffer | | | SPC m t q | run ert | | | SPC m d m | open macrostep transient-state | |
- emacs-lisp

## Keybinding I use

| Key binding | Description |
|---|---|
| `<leader>w hjkl` | Moving between buffers |
| `<leader>bp` | buffer previous |
| `<leader>bn` | buffer next |
| `<leader>bn` | buffer next |
| package-show-list-packages | `space`, `space` then write this command |
| `<leader>m` | for any LSP (you can use also `,`) |
| `helm-find-file` | for finding file in tree |
| customize-create-theme | Creating a color theme |
| `<leader>saf` | To open Ag to find file |
| `<leader>sd`-B2 -A2 keyword | Find the correct keyword among many files in directory |
| `::lsp-ui-imenu` | Will show object tree for the LSP |
| `<leader>fed` | Will open SpaceEmacs buffer quickly (shortcut) |
| `<leader>fj` | in any buffer you back to origin of where is the file in dried |
| `<leadr>fy` | This will allow to copy the file name or other features |
| `<leader>a u` | undo-tree toggle |
| `<leader>w m` | buffer maximized |
| `<leader>x` | So many features you can find such as c for count buffer elements |
| `<leader>xgt` | google translate for selected words in visual mode |
| `gt and gT` | with `tabs` it allows to move between tabs |
| `<leader>t` | many functionality |
| `<leader>a w /` | to search in google search engine |
| mini-map-mode | to toggle the minimap mode |
| M-:`markdown-toc-generate-toc` | this will allow you to generate table of content |
| `help-xref-interned` | offer help on certain function or command in emacs |
| `describe-mode` | describe with help for a given mode |
| `describe-key` | describe a keybinding |
| `describe-face` | about the theme/colors in `emacs` |
| `Rainbow Mode` | Activate the colors in the buffer base don the color layer. |
| `open-command-log-buffer` | This will allow you to see the keystrokes |
| `close-command-log-buffer` | This will close the log-command-buffer |
| `<leader>sw` | Built-in web-browser for `Emacs` , it called `eww` |
| `<leader>` | |

## How to use the LSP

After accessing the `lsp` for the specialized programming language, you can use either `,` or `<leader>m` (e.g., my leader is mapped to `SPAC`) you can find

at `~/.emacs.d/layers/+lang/common-lisp/README.org` more details to deal with the `LSP` features.

| Key binding | Description |
|---|---|
| ~SPC m h a~ | SLIME apropos |
| ~SPC m h d~ | Disassemble symbol at point |
| ~SPC m h h~ | Describe symbol at point |
| ~SPC m h i~ | Inspect definition |
| ~SPC m h H~ | Hyperspec lookup symbol at point |
| ~SPC m h p~ | Browse apropos results for a package's exported symbols |
| ~SPC m h t~ | Toggle tracing of the function at point |
| ~SPC m h T~ | Untrace all functions |
| ~SPC m h <~ | Show all known callers |
| ~SPC m h >~ | Show all known callers |
| ~SPC m h m~ | Show all usages of a macro |
| ~SPC m h r~ | Show references to global variable |
| ~SPC m h s~ | Show all methods specialized on a class |

## Pacakges to be installed

| idex | Package name | Package Description | website |
|---|---|---|---|
| 1 | all-the-icons | For installing all the icons of netree | link |
| 2 | doom-themes | Many themese for my current workflow | link |
| 3 | all-the-icons-install-fonts | For the font supporting the icons | link |
| 4 | `tabnine` | For allowing `tabnine` AI code assistance | link |
| 5 | brew install aspell | It will allow auto-spell to work | link |

## Types of shells in Emacs

1. `shell`
2. `e-shell`

3. `ansi-shell`
4. `vterm` (my favorite)

## How to change the color of specific element in Emacs

I followed the description from the `Chat-GPT`. In `spacemacs`, you can change the color of strings by modifying the syntax highlighting for the relevant programming language mode. Here are the general steps you can follow:

1. Open the file in the relevant programming language mode (e.g. python-mode for Python files, ruby-mode for Ruby files, etc.).
2. Press SPC SPC to open the Spacemacs command interface.
3. Type `customize-face` and select it with TAB.
4. Enter the face name for the string in the prompt (e.g. `font-lock-string-face` for many programming languages).
5. Press ENTER to select the face name.
6. Select the "Foreground" property and modify the color as desired.

- You may need to restart Spacemacs or
- **Note** I used the customized colors to follow more the `Github` theme using
  - `builtin-face` to change the macros like `prinln!` in Rust.
  - `function-name-face` to change the function name and function call.
  - `string` change the string in `Emacs`
  - `variable-name-face` to change the variable names in emacs.

```
(custom-set-faces
 ;; custom-set-faces was added by Custom.
 ;; If you edit it by hand, you could mess it up, so be careful.
 ;; Your init file should contain only one such instance.
 ;; If there is more than one, they won't work right.
 '(font-lock-builtin-face ((t (:foreground "#dcbdfb"))))
 '(font-lock-comment-face ((t (:foreground "#768390"))))
 '(font-lock-function-name-face ((t (:foreground "#dcbdfb"))))
 '(font-lock-keyword-face ((t (:foreground "#f47067"))))
 '(font-lock-string-face ((t (:foreground "#96d0ff"))))
 '(font-lock-variable-name-face ((t (:foreground "#adbac7"))))
 '(markdown-pre-face ((t (:foreground "#f69d50"))))
 '(markdown-url-face ((t (:foreground "#96d0ff" :weight bold))))
 )
```

- Some highlights is not possible for that you can use `customize-face` like the one I encoutered with when I used the `markdown` elements to get them correctly

## Language servers and layers

```
;; Languages
python
```

```
  rust
  emacs-lisp
  git
  helm
  lsp
  csv
  yaml
  html
  emoji
  typescript
  lua
  javascript
  ;;Emacs Basic Plugins
  auto-completion
  better-defaults
  systemd
  markdown
  multiple-cursors
  org
  (shell :variables
         shell-default-height 30
         shell-default-position 'bottom)
  spell-checking
  syntax-checking
  version-control
  docker
  ;;treemacs
  neotree
```

### Selected themes

```
dotspacemacs-themes '(
                     ;;cyberpunk
                     ;;material
                     ;;darkokai
                     ;;doom-one-gh
                     doom-one
                     nord
                     doom-dracula
                     ;;spacemacs-dark
                     spacemacs-light)
```

### Fonts Configurations

```
dotspacemacs-default-font '("SFMono Nerd Font"
                            :size 15.0
```

```
                                    :weight normal
                                    :width normal
                                    :powerline-scale 3
                                    ;; :powerline-offset 1.3
                                    )
```

### Other configurations

```
  ;; Show the scroll bar while scrolling. The auto hide time can be configured
  ;; by setting this variable to a number. (default t)
  dotspacemacs-scroll-bar-while-scrolling nil

  ;; Control line numbers activation.
  dotspacemacs-line-numbers 'relative
  ;;
  ;; If non-nil, start an Emacs server if one is not already running.
  dotspacemacs-enable-server nil
```

### User configurations

```
(defun dotspacemacs/user-config ()
  "Configuration for user code:
This function is called at the very end of Spacemacs startup, after layer
configuration.
Put your configuration code here, except for variables that should be set
before packages are loaded."
  (setq neo-theme 'icons)                    ;; This will show icons in neotree, instead the
  (setq all-the-icons-scale-factor 1.0)      ;; This will control the icon size in neotree
  (setq scroll-margin 5) ; set the scroll margin to 5 lines
  ;;(setq ns-auto-hide-menu-bar t)
  ;;(add-to-list 'default-frame-alist '(undecorated . t)) ;; This will remove the close butt
  (set-background-color "#2d333b") ;; This will add a background similar to Github-pages
  ;; Increase the speed of mouse and cursor while scrolling
  (setq mouse-wheel-scroll-amount '(1 ((shift) . 5) ((control) . nil)))
  (setq mouse-wheel-progressive-speed nil)

  ;;(add-to-list 'custom-theme-load-path "~/.emacs.d/custom_theme/doom-one-ghasak-theme.el")
  ;;(load-theme 'doom-one-theme-ghasak t)

  ;; (define-key evil-window-map (kbd "C-h") 'evil-window-left)
  ;; (define-key evil-window-map (kbd "C-j") 'evil-window-down)
  ;; (define-key evil-window-map (kbd "C-k") 'evil-window-up)
  ;; (define-key evil-window-map (kbd "C-l") 'evil-window-right)
  ;; -------------------- Adding function to hover on defintion ------------

  ;; (define-key evil-normal-state-map (kbd "gh") (lambda ()
```

```
  ;;                                                    (interactive)
  ;;                                                    (spacemacs/jump-to-definition)))

  ;; (define-key evil-normal-state-map (kbd "gh") (lambda ()
  ;;                                                    (interactive)
  ;;                                                    (spacemacs/describe-thing-at-point)))

)
```

## From Scratch

I followed the configurations mentioned in the following thread - Emacs from Scratch

## Configurations wiht DOOM Emacs

But before you doom yourself, here are some things you should know: 1. Do not forget to run 'doom sync', then restart Emacs, after modifying init.el or packages.el in ~/.config/doom. This command ensures needed packages are installed, orphaned packages are removed, and your autoloads/cache files are up to date. When in doubt, run 'doom sync'! 2. If something goes wrong, run `doom doctor`. It diagnoses common issues with your environment and setup, and may offer clues about what is wrong. 3. Use 'doom upgrade' to update Doom. Doing it any other way will require additional steps. Run 'doom help upgrade' to understand those extra steps. 4. Access Doom's documentation from within Emacs via 'SPC h d h' or 'C-h d h' (or 'M-x doom/help')