Vim Workshop

By ?

time?

Part 0 -- Intro

Our aim

Learn how to use vim in your future project, and **THOWR YOUR MOUSE**.

What is vim

- An editor
- vi -> visual interface
- vim -> vi Improved

As you can see from the advertisement of this workshop, vim has many short-cuts and other fabulous functions that can aid our programming.

Why vim

- No switch between keyboard and mouse
- Easy to move
- No duplicate operation

How to learn vim

Practice Practice Practice.

From a small program ~~or from an English essay~~

Part 1 -- Basic Operation

Menu

Before Start

1. Use `:help`

   You can use the command in the vim just like what you've done in the matlab

2. Vimtutor

   You can type `vimtuor` in your terminal to start the vimtutor, a tutor for vim with language related to your system language. It only takes you about 25~30 min. If you feel just give it a try if you want to go over the basic command of vim.

3. Official Website

   Community is an important part of editor. There are a lot of good quality resource made by the developers and experienced vimers on the official website. You can join the community or search on the official website to get supported.

4. Vim Adventure

   Vim adventure is a great game intended to help beginners to learn the vim with fun. Since it is non-free, you can pay for the rest of the game or find a reverse engineered version.

5. Openvim
6. Cheatsheet
7. You should know how to work in a terminal.

To get a quick start, we recommned you use our sample config file.

```
cd ~/
git clone git@github.com:TechJI-2023/Vim_wksp_2024.git
mkdir ~/.vim
cp ~/Vim_wksp_2024/vimrc-noplugin/vimrc ~/.vim/vimrc
cd ~/.vim
vim ~/.vimrc
# Then quit by using :q
```

Mode Intro

1. Insert Mode
2. Normal Mode
3. Visual Mode
4. Replace Mode*
5. Command Mode

Insert Mode

```
vim ./test-Vim/article.txt
```

1. Open Insert Mode:
    1. `i`: insert
    2. `a`: append
    3. `o`: insert in next line
    4. `A`: append at the end of the line
    5. `I`: insert text before the first non-blank in the line
    6. `O`: insert in previous line
    7. `s`: delete char and insert
    8. `S`: delete line and insert
    9. `c`: `c` is equal to "change". `c` is motion like `d` and `y`.
    10. `C`: delete until the end of the line and insert
2. Exit Insert Mode
    1. Press `Esc`: exit insert mode
    2. ctrl c: exit insert mode and abort current command

Normal Mode

1. Navigating
    1. Basic: hjkl

    You can use `hjkl` to replace the direction key on your key board.

    `h` for ⬅️, `j` for ⬇️, `k` for ⬆️, `l` for ➡️

    2. Word: bwe

       You can use the wbe to jump word by word.

       Difference between `word` and `WORD`: The `word` is the combination of (a-zA-Z0-9), or the combination of punctuation and character like `@##$%` while the WORD is any non-empty string between the white-space.

       NOTE THAT `as213df%##^>?^&` is two `word`, `as213df` and `%#^>?^&`.

       - `w`: go to the beginning of next word
       - `b`: go to the beginning of previous word
       - `e`: go to the end of the word
       - `W`: go to the beginning of next WORD
       - `B`: go to the beginning of previous WORD
       - `E`: end of the WORD

    3. Line: 0, ^, $

       - `0`: Go to the beginning of line
       - `^`: Go to the beginning of line (after whitespace)
       - `$`: Go to the end of line

    4. Char:

       - `f{char}`: Go forward to character
       - `F{char}`: Go backward to character

    5. Document

       - `gg`: First line
       - `G`: Last line
       - `:{number}`: Go to line {number}
       - `{number}G`: Go to line {number}
       - `{number}j`: Go down {number} lines
       - `{number}k`: Go up {number} lines

    6. Window

       - `zz`: Center this line
       - `zt`: Top this line
       - `zb`: Bottom this line
       - `H`: Move to top of screen
       - `M`: Move to middle of screen
       - `L`: Move to bottom of screen

2. Clipboard:

   1. `x`: Delete the character
   2. `dd`: Cut the line
   3. `yy`: Yank line
   4. `p`: Paste
   5. `P`: Paste before
   6. `"*p / "+p:` Paste from system clipboard
   7. `"*y / "+y`: Paste to system clipboard

   Note: For users using wsl, you may encounter some problems when using the clipboard, this issue cannot be solved by part2

3. Editing
   1. `u`: You can use `u` to undo the editing
   2. `<C-R>`: You can use `<C-R>` to redo the editing
4. Opeartors

   > Operators let you operate in a range of text (defined by motion). These are performed in normal mode.

   For Example:

Operator	**Motion**
`d`	`w`
`c`	`w`
`dl`	delete character (alias: "x")
`dw`	delete next word
`dW`	delete next WORD
`diw`	delete inner word
`daw`	delete a word
`diW`	delete inner WORD
`daW`	delete a WORD
`db`	delete to beginning of the word
`dgn`	delete the next search pattern match
`dd`	delete one line(actually cut one line)
`dis`	delete inner sentence
`das`	delete a sentence
`dib`	delete inner '(' ')' block
`dab`	delete a '(' ')' block
`dip`	delete inner paragraph
`dap`	delete a paragraph
`diB`	delete inner '{' '}' block
`daB`	delete a '{' '}' block

- Text Objects

key	**Object**
p	Paragraph
w	Word
s	Sentence
[({ < A [], (), or {}	block
' " `	A quoted string
b	A block (
B	A block in {
t	A XML tag block

- Sentence and Paragraph in vim

  For more combinations, use `:help motion.txt`

  You can repeat the command by assigning a number.

  For example: 3dd means delete three lines.Times of repetition should be the multiplication of the numbers. Eg. 2y3y means yank 6 lines.

Why key in vim is so uncomfortable?

Life Hack

- Use `.` to repeat previous edit
- Use `:!` to run command in terminal
- Use `:r` to insert the file below the cursor
- Change `esc` with `capslock`

Visual Mode

1. Visual Mode `v`

   Start Visual mode per character.
2. Visual Line Mode `V` (`shift`+`v`)

   Start Visual mode linewise.
3. Visual Block Mode (`ctrl`+`v`)

   Start Visual mode blockwise.
4. Use `gu` (lowercase) and `gU`(uppercase).
5. Use `>` and `<`(change indentation)
6. d, y, c, o in the visual mode

Exercise

Change `wallpaper.txt`

```
wallpaper_1.jpg
wallpaper_1.jpg
......
wallpaper_1.jpg
```

Into

```
Wallpaper_1.jpg
Wallpaper_1.jpg
......
Wallpaper_1.jpg
```

Replace Mode*

You can use `r` to replace a character.

You can use `R` to enter the replace mode.

Command Mode

1. Exiting Vim:
    1. `:q`: quit the file
    2. `:qa`: quit all the file
    3. `:w`: Write(Save the file)
    4. `:wq`: Write and quit the file
    5. `:x`: Write and quit the file(only write when there is change).
    6. `ZZ`: Save and quit （the same to :x)
    7. `ZQ`: Quit without checking changes
    8. `!`: You can add a `!` at the end of the command to do the command by force. For example to quit the file by force, you can use `:q!`
2. Call the terminal

    You can call the terminal by using the command `:term`. You can also use term to run command like compiling a c-file. For example:

    ```
    :term gcc vim.c
    :term ./a.out
    ```

    You can also open a terminal window in the vim using the command; `:term`

    You can use command to determine the position of the terminal. For example:

```
:belowright term ++rows=10
```

3. Run Vim command

   Believe it or not, vim has its own language. You can use the command of vim to personalize your vim. This part will be included in the `Modify .vimrc(noplugin)`

- Search

  You can use search the string by entering `/` and `?`

  Difference between `/` and `?`:
  `/` means search from top to bottom,
  `?` means search from bottom to top.

  For example, you can use /\<hello\> to search the word `hello` from top to bottom

  - `n`: Search for the next pattern
  - `N`: Previous match
  - `*`: Next whole word under cursor
  - `#`: Previous whole word under cursor

  You can use \<{word}\> to search for the "word"

  - range is the searching range, You can define the range in following way:

    ```
    {start},{end}
    ```

    There are some characters have special meaning when defining the range:

    `%` stands for all the lines in the file

    `.` stands for the current line

    `$` stand for the last line in the file

    If range is not set, subtitute will only happens on the current line.

  - `s` stands for substitute
  - Commonly used flags
    - `g`: Replace all occurrences in the line. Without this argument, replacement occurs only for the first occurrence in each line.
    - `i`: Ignore case for the pattern.
    - `I`: Don't ignore case for the pattern.
    - `n`: Report the number of matches, do not actually substitute.

  Example:

```
"replace all int with double
:%s/int/double/g

"replace first int in each line with double
:%s/int/double/
```

```
"replace all int in line1-line3 with double
:1,3s/int/double/g

"replace all int from current line to the next 4 line with double
:.,.+4s/int/double/g

"do :%s/int/double/g for 4 times(from current line to current+3 line)
:%s/int/double/g4
```

For more information, you can type `:help substitute` in vim.

Part 2 -- vimrc

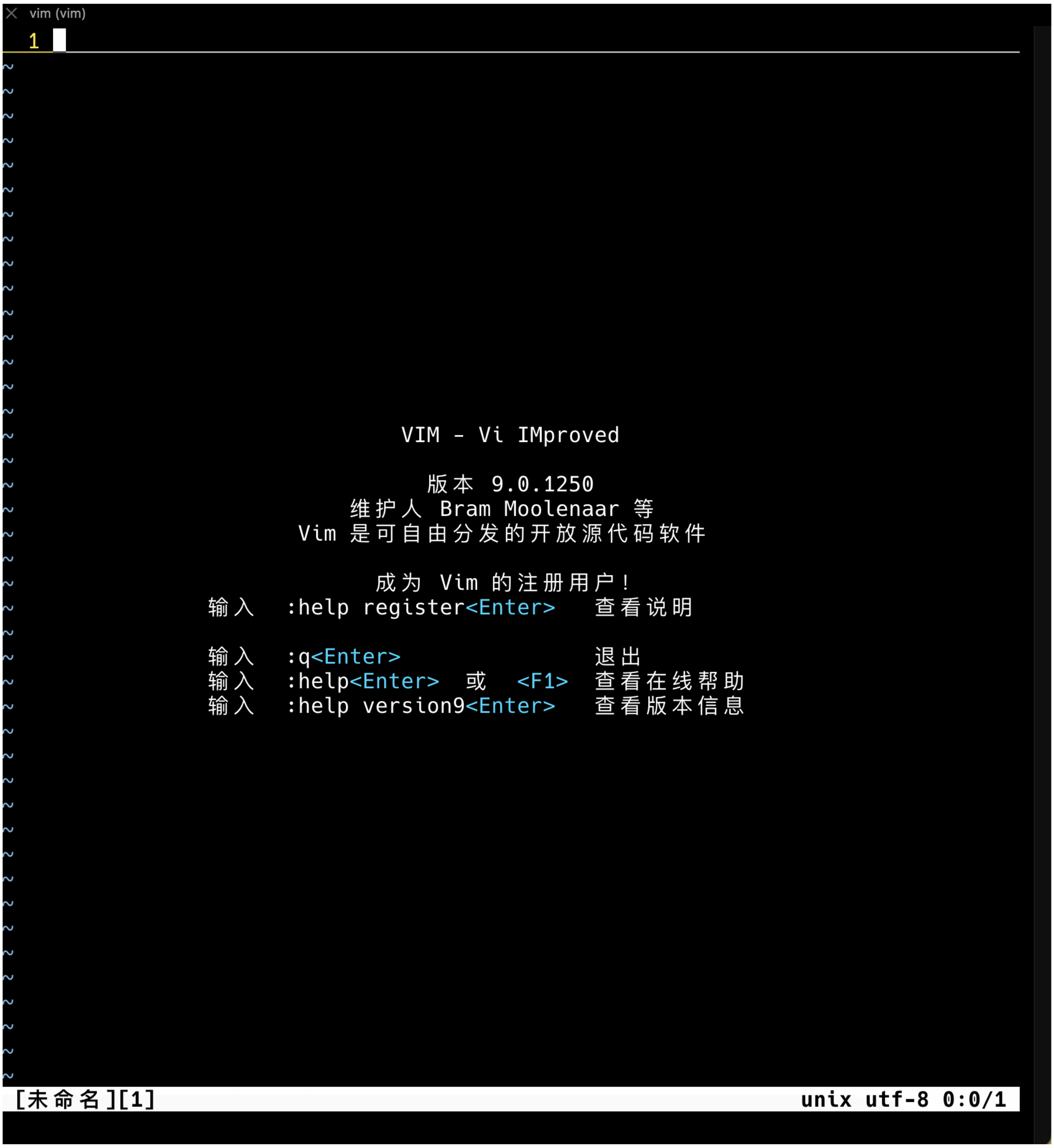
Without Plugin

See -> `Vim_wksp_2024/vimrc-noplugin/vimrc`

We strongly suggest you do this:

```
cd ~/
git clone git@github.com:TechJI-2023/Vim_wksp_2024.git
mkdir ~/.vim
cp ~/Vim_wksp_2024/vimrc-noplugin/vimrc ~/.vim/vimrc
cd ~/.vim
vim ~/.vimrc
```

Now your vim should be look like this



```
vim (vim)
  1 
~
~                        VIM - Vi IMproved
~
~                          版本 9.0.1250
~                    维护人 Bram Moolenaar 等
~                Vim 是可自由分发的开放源代码软件
~
~                        成为 Vim 的注册用户！
~           输入  :help register<Enter>   查看说明
~
~           输入  :q<Enter>               退出
~           输入  :help<Enter>  或  <F1>  查看在线帮助
~           输入  :help version9<Enter>   查看版本信息
~
[未命名][1]                                              unix utf-8 0:0/1
```

Next `:q`

And `vim vimrc`

Read it line by line! (cover `split` btw)

Shortcut key&key mapping

1. `map` Equivalent key

`noremap` Change key

change `a` into `b`

```
noremap a b
```

Move faster!

```
noremap H 5h
noremap J 5j
noremap K 5k
noremap L 5l
```

Quick save and source

```
map s <nop>
"s为删掉当前字符并进入写入模式
map S :w<CR>
"<CR>表示回车，效果：S即保存写入
map Q :wq<CR>
"效果：Q即退出vim
map R :source R $MYVIMRC<CR>
"效果：R刷新vimrc
map ; :
```

2. `set` -> see detail in `vimrc-noplugin/vimrc`
3. split screen

`:spilt` split up and down

In `vimrc-noplugin`, `su` for split up， `sd` for split down

`:vspilt` split left and right

In `vimrc-noplugin`, `sl` for split left `sr` for split right

Move the cursor position manually `ctrl+w+h/j/k/l`

4. Change backgroud (Do you want to be manuel)

   `:color` +tab

   Go back to default `:color default`

With Plugin

See -> `Vim_wksp_2024/vimrc-withplugin/vimrc`

We strongly suggest you do this in wksp:

1. Install Vundle

```
cd ~/.vim
mkdir bundle
cd bundle
git clone https://github.com/VundleVim/Vundle.vim.git
```

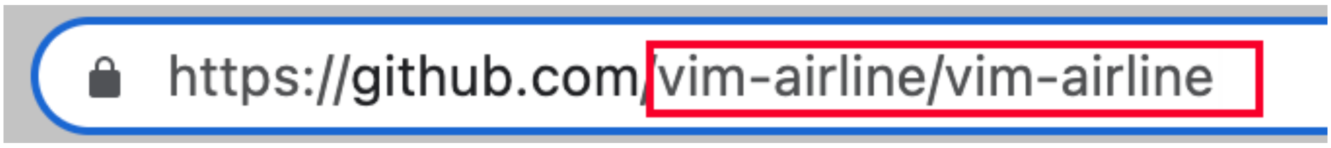
Windows users may need to refer to this -> https://github.com/VundleVim/Vundle.vim/wiki/Vundle-for-Windows

2. Write `vimrc`

You don't need to write now because I have prepared it for you. All you need to do now is to understand!

```
set nocompatible              " be iMproved, required
filetype off
" set the runtime path to include Vundle and initialize
set rtp+=~/.vim/bundle/Vundle.vim
call vundle##begin('~/.vim/bundle')
" alternatively, pass a path where Vundle should install plugins
"call vundle##begin('~/some/path/here')

" let Vundle manage Vundle, required



call vundle##end() " required

filetype on " required
filetype indent on  " required
filetype plugin on  " required
filetype plugin indent on " required
" To ignore plugin indent changes, instead use:
"filetype plugin on
"
" Brief help
" :PluginList       - lists configured plugins
" :PluginInstall    - installs plugins; append `!` to update or just :PluginUpdate
" :PluginSearch foo - searches for foo; append `!` to refresh local cache
" :PluginClean      - confirms removal of unused plugins; append `!` to auto-approve
removal
"
" see :h vundle for more details or wiki for FAQ
" Put your non-Plugin stuff after this line
```

3. Find and install plugin

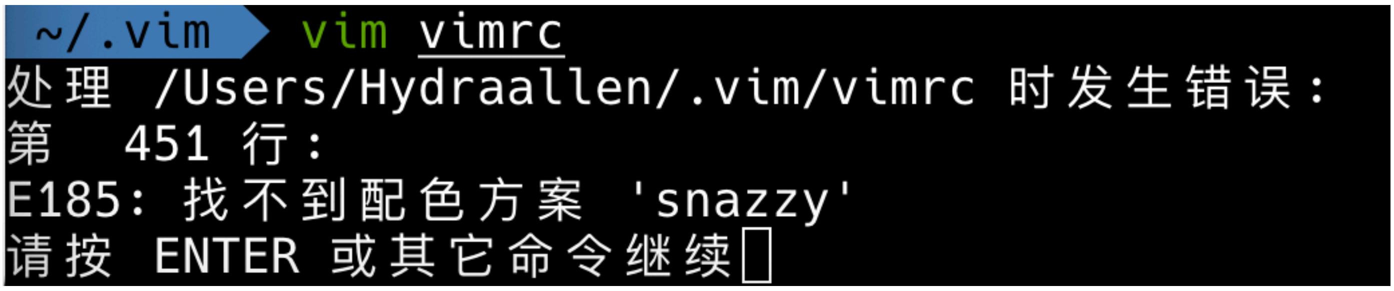
- Search in Github
- 
https://github.com/vim-airline/vim-airline
- write in vimrc `Plugin 'vim-airline/vim-airline'`
- plug install `:PluginInstall`或`:BundleInstall`
  plug update `:PluginUpdate`
  plug uninstall `:PluginClean`

4. Time to have a try!

We strongly suggest you do this in wksp:

```
cd ~/
git clone git@github.com:TechJI-2023/Vim_wksp_2024.git
mkdir .vim
cd .vim
cp ~/Vim_wksp_2024/vimrc-withplugin/vimrc ~/.vim/vimrc
vim vimrc
```

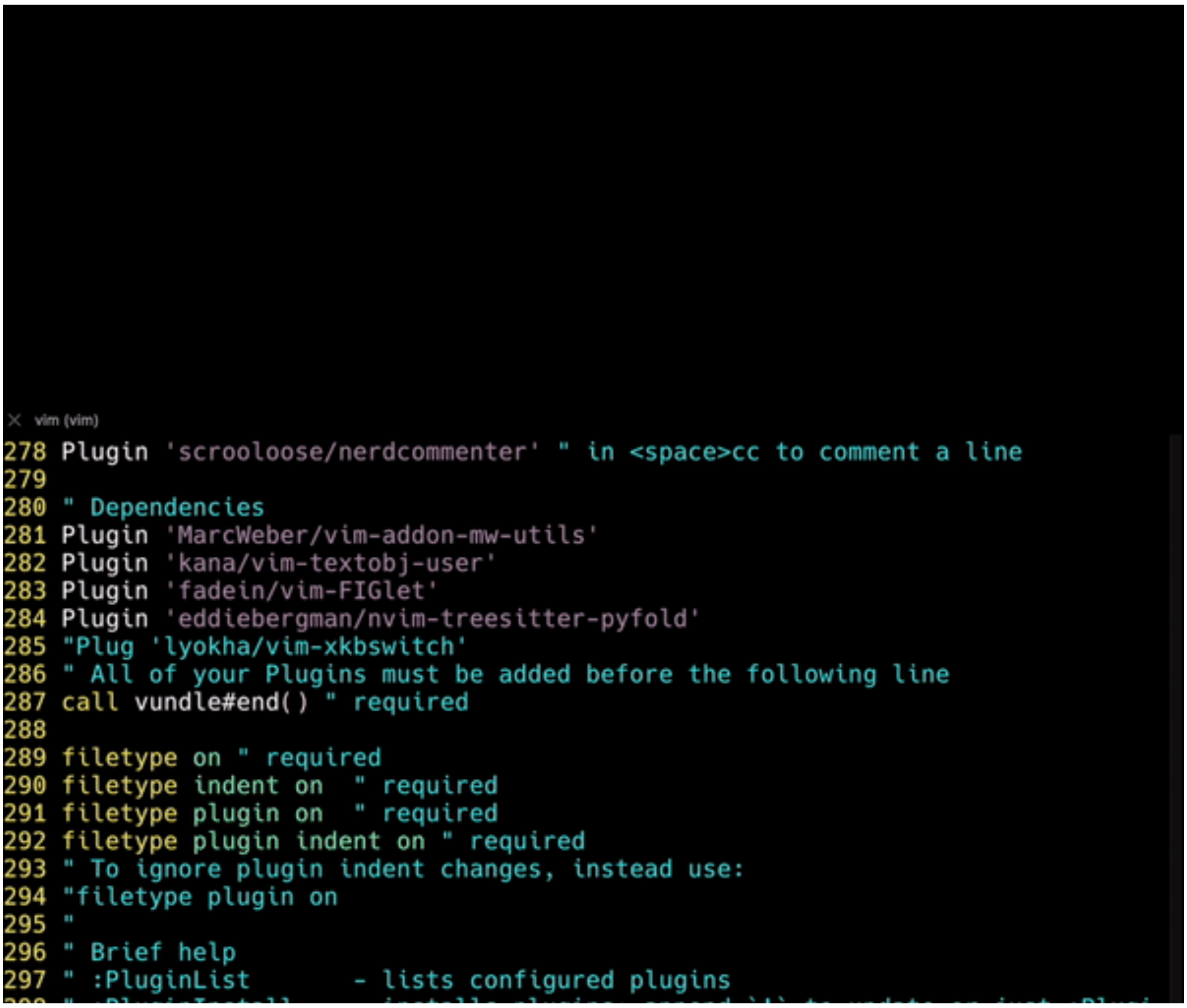
There will be some warnings or errors. But **DO NOT PANIC**, just press `enter`



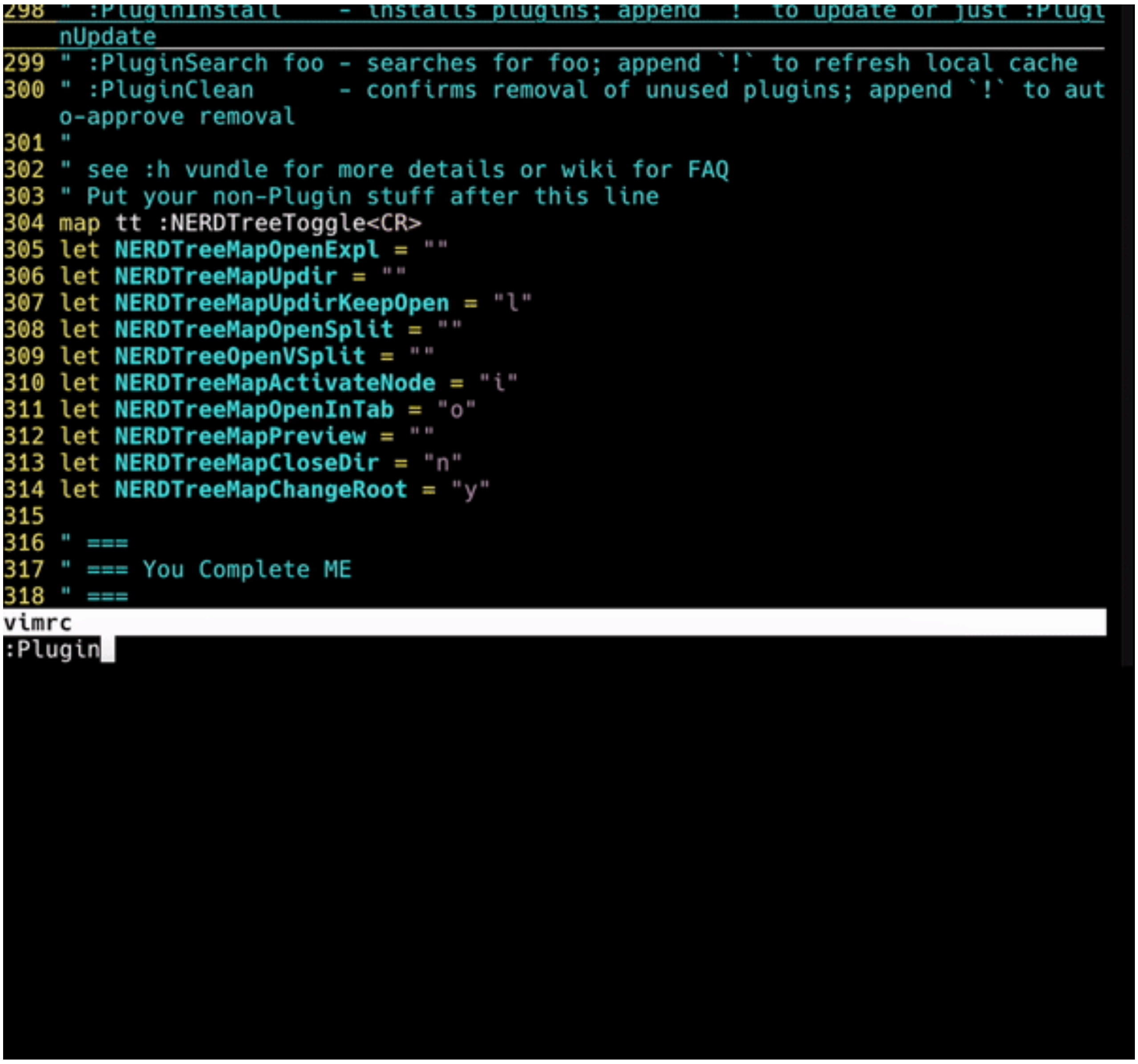
```
 ~/.vim  vim vimrc
处理 /Users/Hydraallen/.vim/vimrc 时发生错误：
第  451 行：
E185: 找不到配色方案 'snazzy'
请按 ENTER 或其它命令继续
```

Now install some plugin `:PluginInstall`

(In vimrc-withplugin, we included `airline, snazzy,ale,nerdtree`)

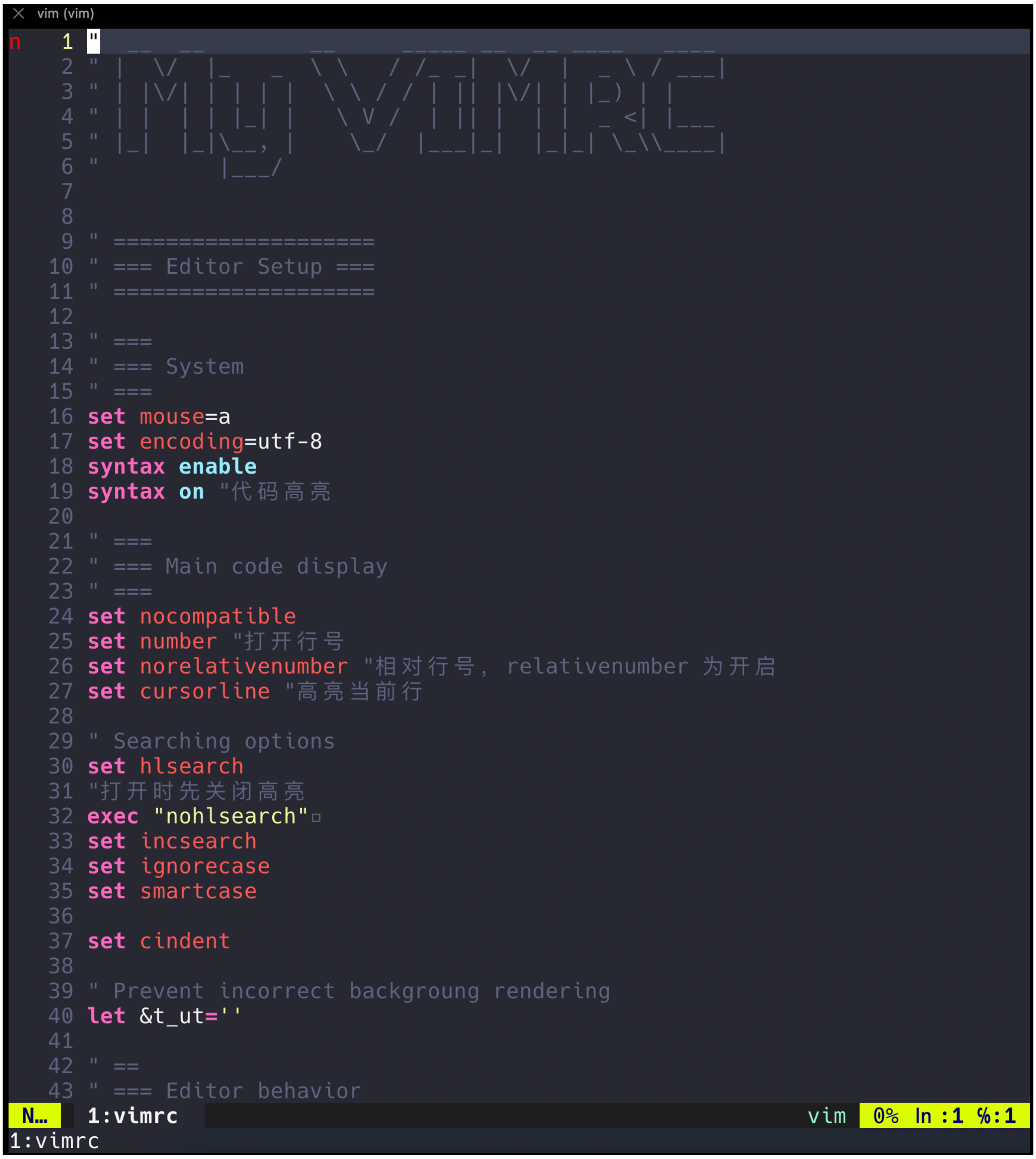


```
Plugin 'scrooloose/nerdcommenter' " in <space>cc to comment a line

" Dependencies
Plugin 'MarcWeber/vim-addon-mw-utils'
Plugin 'kana/vim-textobj-user'
Plugin 'fadein/vim-FIGlet'
Plugin 'eddiebergman/nvim-treesitter-pyfold'
"Plug 'lyokha/vim-xkbswitch'
" All of your Plugins must be added before the following line
call vundle#end() " required

filetype on " required
filetype indent on  " required
filetype plugin on  " required
filetype plugin indent on " required
" To ignore plugin indent changes, instead use:
"filetype plugin on
"
" Brief help
" :PluginList       - lists configured plugins
```



```
" :PluginInstall    - installs plugins; append `!` to update or just :Plugi
nUpdate
" :PluginSearch foo - searches for foo; append `!` to refresh local cache
" :PluginClean      - confirms removal of unused plugins; append `!` to aut
o-approve removal
"
" see :h vundle for more details or wiki for FAQ
" Put your non-Plugin stuff after this line
map tt :NERDTreeToggle<CR>
let NERDTreeMapOpenExpl = ""
let NERDTreeMapUpdir = ""
let NERDTreeMapUpdirKeepOpen = "l"
let NERDTreeMapOpenSplit = ""
let NERDTreeOpenVSplit = ""
let NERDTreeMapActivateNode = "i"
let NERDTreeMapOpenInTab = "o"
let NERDTreeMapPreview = ""
let NERDTreeMapCloseDir = "n"
let NERDTreeMapChangeRoot = "y"

" ===
" === You Complete ME
" ===
vimrc
:Plugin
```

And now your vim should be look like this:


vim (vim)

```
"  __  __        __     _____ __  __ ____   ____
" |  \/  |_   _  \ \   / /_ _|  \/  |  _ \ / ___|
" | |\/| | | | |  \ \ / / | || |\/| | |_) | |
" | |  | | |_| |   \ V /  | || |  | |  _ <| |___
" |_|  |_|\__, |    \_/  |___|_|  |_|_| \_\\____|
"         |___/


" ====================
" === Editor Setup ===
" ====================

" ===
" === System
" ===
set mouse=a
set encoding=utf-8
syntax enable
syntax on "代码高亮

" ===
" === Main code display
" ===
set nocompatible
set number "打开行号
set norelativenumber "相对行号, relativenumber 为开启
set cursorline "高亮当前行

" Searching options
set hlsearch
"打开时先关闭高亮
exec "nohlsearch"□
set incsearch
set ignorecase
set smartcase

set cindent

" Prevent incorrect backgroung rendering
let &t_ut=''

" ==
" === Editor behavior
```

```
N…  1:vimrc                                   vim  0%  ln :1  %:1
1:vimrc
```

If you want use more plugin and don't want to spend too much time, you may refer to `Vim-Wksp/vimrc-sample/vimrc`

Vim Macro

You can use vim macro to complete the repetition work.

```
1. Enter `q\<char\>` to start recording the macro.
2. Edit one line
3. Enter `q` to stop recording
4. Move to the line you modify(You can include this step in recording the macro)
5. Enter `@\<char\>`
```

Now exercise!

Write in `empty.txt`

```
wallpaper_1.jpg
wallpaper_2.jpg
......
wallpaper_100.jpg
```

Hint: `Ctrl+a` integer+1； `Ctrl+x` integer-1

Suggestions

1. If you want use vim more smoothly, you may need to change `esc` and `Capslock`

In macOS, you may use karabiner

In WIndows, you may use Powertoys

In Linux, I believe you can find one by yourself!

2. plugin

Here are some recommendations:

插件list（推荐）

- airline 状态栏
- snazzy 配色

  ```
  colorscheme snazzy
  ```

- nerdtree 文件树
- tagbar 显示函数列表
- ale 错误检查
- YouCompleteMe 代码自动补全
- markdown-preview

3. Cheet Sheet

   https://devhints.io/vim

4. b站 @ TheCW

   https://space.bilibili.com/13081489

Other vim

- Neovim

NeoVim 项目地址: GitHub - neovim/neovim

Vim 原项目地址: GitHub - vim/vim

- lunarvim

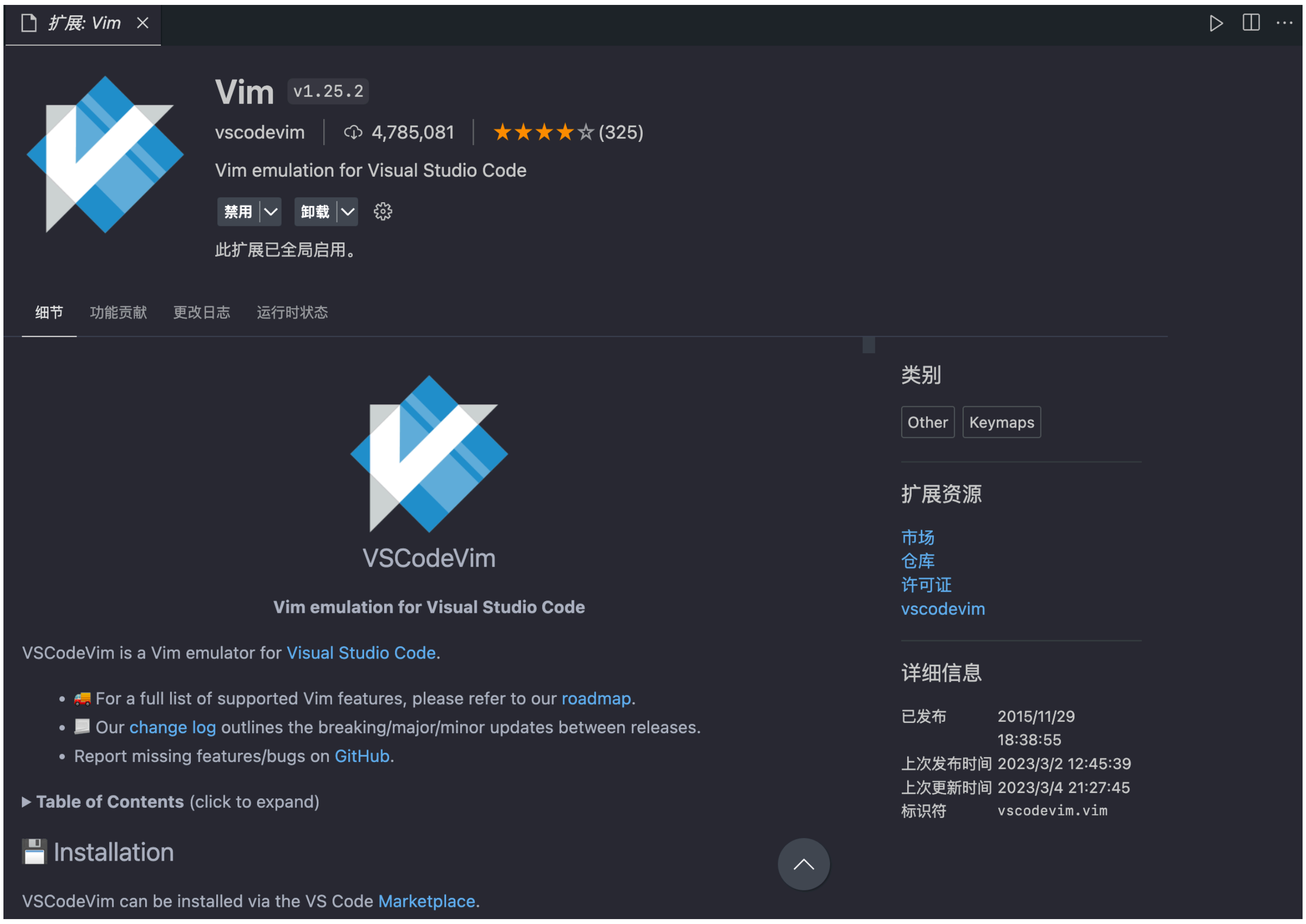
有非常详细的官方文档，可以参考`Vim_wksp_2024/vimrc-sample/lvim`或参考配置文件仓库

- astronvim
- supervim
- Manuel's neovim -> `Vim_wksp_2024/vimrc-sample/manuel-init.vim`
- emacs -> coming soon!

Throw Away Your Mouse!

- vscode

  You may refer to -> `Vim_wksp_2024/vimrc-sample/vimrc-vscode`


扩展: Vim

Vim v1.25.2

vscodevim | 4,785,081 | ★★★★☆ (325)

Vim emulation for Visual Studio Code

禁用 卸载

此扩展已全局启用。

细节 功能贡献 更改日志 运行时状态

VSCodeVim

Vim emulation for Visual Studio Code

VSCodeVim is a Vim emulator for Visual Studio Code.

- For a full list of supported Vim features, please refer to our roadmap.
- Our change log outlines the breaking/major/minor updates between releases.
- Report missing features/bugs on GitHub.

▶ **Table of Contents** (click to expand)

Installation

VSCodeVim can be installed via the VS Code Marketplace.

类别

Other Keymaps

扩展资源

市场
仓库
许可证
vscodevim

详细信息

已发布	2015/11/29 18:38:55
上次发布时间	2023/3/2 12:45:39
上次更新时间	2023/3/4 21:27:45
标识符	vscodevim.vim

- help/man
- Ranger -> available for macOS and Linux （Now, I'm using joshuto)

  Use vim key bindings to manage your files

  You may refer to -> `Vim_wksp_2024/Usage-reference/ranger.md` and `Vim_wksp_2024/vimrc-sample/ranger` and `Vim_wksp_2024/vimrc-sample/joshuto`.

  (If you use windows with powershell and posh, you can use lf)

- Use vim key bindings when surffing the Internet
  - vimium -> Google Chrome
  - vimari -> Safari
  - vimium c -> edge

You may refer to -> `Vim-Wksp/Usage-reference/vimium.md`

- sioyek -> available for macOS, Windows and Linux

  Use vim key bindings to view pdf files

  You may refer to -> `Vim-Wksp/Usage-reference/sioyek.md`

- lazygit
- tmux -> vim mode in tmux