Your Vim is Only for You
mopp

GitHub

- mopp

Interesting

- Vim, Hobby OS
  Self-published techbooks

Job

- MC of VimConf2017/2018/2019
  Quipper Limited Japan Branch
  - Meguro, Tokyo
  - Elixir, Go, and Ruby on Rails
Introduction
Target of this talk

**Middle level Vimmer**

A person who

- has vimrc with hundreds of lines
- doesn’t hesitate to open vimrc
- is not so familiar with Vim script

**Vim level**

- You are Vim
- Vimmer emperor
- You can create plugin
- You can enhance your Vim by many plugins
- You can config Vim and plugins
- You can write code with Vim
- You can input something with Vim
- You can quite Vim
- Not Vimmer
How to be more productive with Vim
Ways to increase productivity

1. Master the default Vim
   - No configs and plugins
   - Powerful, but unhandy sometimes

2. Customize Vim by myself
   - You can send a patch
   - Not easy

3. **Customize Vim by vimrc**
   - Customizability of Vim is high

Theme of this talk
Theme of this talk

Increase productivity by vimrc

Overcome vimrc which has many copy-and-past

Like Vim more and more
mopp’s vimrc
You can find my vimrc

- `mopp/dotfiles`
- 1303 lines

I use Neovim
Policies

1. Shorter startup time is better
2. Portability: No errors without plugins
3. Compatibility: Works with Neovim/Vim
4. Put everything into one file
Policy 1: Shorter startup time is better

Why?
- **Shorter startup time doesn’t distract us**
- I open/close Vim many times in a day.

How?
- 108 plugins via **dein.nvim**
- 55 plugins with **lazy loading**
  - filetype, command, mapping, etc

```
nvim --startuptime stime
```

![Startup time comparison](chart.png)
Policy 2: Portability: No errors without plugins

Why?
- It’s handy immediately *after installing OS or copying vimrc*

How?
- My vimrc is composed of the two parts
- Pure Vim part
  - Only vim itself configs
- The checker
  - *terminates loading vimrc* if plugins not found

vimrc structure

Pure Vim part

Plugin part

Plugins exist?
Policy3: Compatibility: Works with Neovim/Vim

Why?

- I switch Neovim and Vim sometimes to use
  - New features
  - Plugins support either Vim or Neovim
- Neovim specific options exist
  - Please see :vim-differences for more details

How?

- Use `has('nvim')`
  - return 1 if Neovim, 0 otherwise

```bash
if has('nvim')
    set inccommand=split
    set scrollback=5000
    set wildoptions=
endif
```
Policy 4: Put everything into one file

Why?
- I simply prefer this style 😊
- Find somethings easily

How?
- Just put the all into vimrc
- Filter by denite.nvim
  - :Denite outline
  - / works well
1. `<Leader>` is space

2. `number + relativenumber`

3. `Work quickly by mappings`

4. `Extending plugin`

5. `Plugin combination`
<Leader> is space

Why?
- It’s easy to press
- Plugin prefixes <Leader> to default mappings

How?
- Just set the variable

```
" Set <Leader>
let g:mapleader = ''
```
number + relativenumber

🤔 Why?

- I can know how far from current line easily

🔧 How?

- Just set the configs

```
set number
set relativenumber
```

⚠️ Note

- It makes non-Vimmer confused
- I disable it during pair-programming
Work quickly by mappings

Why?
- To **reduce the time of operations**

How?
- Define mappings what you do many time in a day.

```
" Write buffer to file
nnoremap <silent> <Leader>w :<C-U>write<CR>

" Go to prev/next tab
nnoremap <Leader>j gT
nnoremap <Leader>k gt

" Open help by word under the cursor.
nnoremap <silent> <Leader>hh :<C-U>help <C-R><C-W><CR>
nnoremap <silent> <Leader>ht :<C-U>tab help <C-R><C-W><CR>
```

I save buffer after every input.

I use **tabs** heavily.

I want to refer help **quickly** when I would like to know something about Vim.
Which interface should we use?
- If you use many times -> Mapping
- If you use sometime -> Command
- Otherwise -> Function

Operation speed
- How easily we can input

Expressiveness
- How obvious what it can

Note
- Command/function can take arguments
Extending plugin: GrepBuffer by capture.vim

Why?

- To **analyze application logs easily**

How?

- Filter current buffer by given word
- **Capture.vim** shows command result on new buffer
- GrepBuffer! works like grep -v

```vim
command! -nargs=1 -bang GrepBuffer
  
  :execute printf(':Capture! global%s/%s/print',
  
  expand('<bang>'),
  
  <q-args>)
```
Plugin combination: defx.nvim + vim-choosewin

Why?
● To open file at target buffer easily

How?
● defx.nvim + vim-choosewin
● Filer and window selector

My standard buffer layout

```vimscript
function! DefxChoosewin(context) abort
    let l:winrs = find_winrs() " Modified for slide
    let l:opts = {'auto_choose': 1, 'hook_enable': 0}
    for filename in a:context.targets
        call choosewin#start(l:winrs, l:opts)
        execute 'edit' filename
    endfor
endfunction

nnoremap <silent><buffer><expr><nowait> w
    \
defx#do_action('call', 'DefxChoosewin')
```

About 10 lines

NOTE: I usually 3 windows with vsplit
Outline

Introduction

mopp’s vimrc

How to grow vimrc

Summary
How to grow vimrc
Writing codes

First step is to use Vim.

Notice inconvenience

- Examples
  - Cannot align well
  - Many key inputs are required

Keypoint: DRY

- Don’t Repeat Yourself rule
- If it distracted your attention, It is things to be improved
Notice inconvenience

Keep your idea in your mind

- Take a note
  - Google Keep, Evernote, Scrapbox
- Post brief messages
  - Twitter, Slack, IRC
  - #vim channel is good practice.

vimrc growing flowchart

1. Writing codes
2. Enough?
   - Yes
   - No
   - Keep in mind
     - Find Plugins/Configs
       - Found?
         - Yes
         - Configure
         - No
         - Write Vim script
         - No
         - Find Plugins/Configs
         - No
         - Write Vim script
   - No
     - Enough?
       - Yes
       - Writing codes
       - No
Find plugins/configs

Search config or plugins on Internet

Ask somebody
- Colleagues on Slack
  - #vim channel is good practice
- Followers on Twitter

Writing codes

Enough?

Yes

Keep in mind

Find Plugins/Configs

Found?

Yes

Configure

No

Write Vim script
When you find, setup

Do not just copy and paste

Read docs for 30 seconds at least

- Vim itself is well documented.
  - You can also know related configs
- Check what **interfaces** plugin has
  - Examples
    - :help gitgutter-commands
    - :help lexima-key-mappings
When you don’t get satisfied?

Why?

- No configs, No plugins or
  The interface does not fit use-case

It’s chance to grow vimrc

- Write the all into vimrc at first
  - You don’t need plugin at first
- Extend/combine exist plugins
- Keep dogfooding/improving
  - add new features if you want

vimrc growing flowchart

1. Writing codes
2. Enough?
   - Yes
   - No
     - Keep in mind
3. Find Plugins/Configs
   - Found?
     - Yes
     - No
     - Write Vim script
   - No
     - Configure
**When you don’t get satisfied?**

**Tips for improving quickly**

- Use Git/GitHub to edit/share codes easily
  
  - a.k.a dotfiles
  
  - anishathalye/dotbot

---

```
“Open vimrc
nnoremap <silent> <Leader>ev :<C-U>tab drop $MYVIMRC<CR>

“Reload vimrc
command! ReloadVimrc :source $MYVIMRC
```

---

**vimrc growing flowchart**

```
Writing codes

Enough?

Yes

Keep in mind

Find Plugins/Configs

Found?

Yes

Configure

No

Write Vim script
```

---

```
Writing codes

Enough?

Yes

Keep in mind

Find Plugins/Configs

Found?

Yes

Configure

No

Write Vim script
```
From vimrc to plugin

Iterate this cycle

- vimrc will getting large

Vim plugin comes from your vimrc
Cleanup vimrc every 6 months

Remove unused configs/plugins

- Pruning is necessary for vimrc as well as plants

Hold a meetup

- The purpose is clear
- 1~3 times is best to be done
- Good opportunity for sharing your knowledge
Outline

- Introduction
- mopp’s vimrc
- How to grow vimrc
- Summary
Summary
Summary: What you take

Optimize Vim for you
- Your Vim is only for you
- Find out your policy
- What you want for Vim

Keep improving
- Prepare tools to improve
- Share your knowledge
- I didn’t discover the all by myself
- Output is important
Thank you
License and Acknowledgement

Noto Emoji
- Apache license, version 2.0

Plugins and tool
- Shougo/dein.vim
- Shougo/denite.nvim
- t9md/vim-choosewin
- tyru/capture.vim
- airblade/vim-gitgutter
- cohama/lexima.vim
- anishathalye/dotbot