My dark plugins development history ~ over 10 years ~

Shougo
"My presentation may be the last. So I need to speak the development history."
Profile

- Shougo= 暗黒美無王
- “Dark Vim Master”
  (pronunciation: Uncock Vim Awe)
- Text editor is the world
VimConf 2019’s theme

• “How to be more productive with Vim?”
“more productive with Vim” is ...

- To develop dark plugins is my productive method!

Pinned

<table>
<thead>
<tr>
<th>deoplete.nvim</th>
<th>dein.vim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark powered asynchronous completion framework for neovim/Vim8</td>
<td>Dark powered Vim/Neovim plugin manager</td>
</tr>
<tr>
<td>Python 4.5k 257</td>
<td>Vim script 2.4k 137</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>denite.nvim</th>
<th>deol.nvim</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dark powered asynchronous unite all interfaces for Neovim/Vim8</td>
<td>Dark powered shell interface for NeoVim and Vim8.</td>
</tr>
<tr>
<td>Python 1.5k 196</td>
<td>Vim script 170 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>deoppet.nvim</th>
<th>defx.nvim</th>
</tr>
</thead>
<tbody>
<tr>
<td>The dark powered snippet plugin for neovim</td>
<td>The dark powered file explorer implementation</td>
</tr>
<tr>
<td>Python 141 1</td>
<td>Vim script 428 29</td>
</tr>
</tbody>
</table>
My theme is ...

- My dark plugins development history over 10 years
- “Why does Shougo discard Vim script and choose Python3 instead?”
What is “dark plugins”? (1)

- I want to do everything in Vim(neovim) like Emacs
- It is not allowed by “:help design-not” in Vim

VIM IS... NOT
design-not

- Vim is not a shell or an Operating System. It does provide a terminal window, in which you can run a shell or debugger. E.g. to be able to do this over an ssh connection. But if you don't need a text editor with that it is out of scope (use something like screen or tmux instead).
  A satirical way to say this: "Unlike Emacs, Vim does not attempt to include everything but the kitchen sink, but some people say that you can clean one with it. ;-)"
  To use Vim with gdb see: http://www.agide.org and http://clewn.sf.net.
- Vim is not a fancy GUI editor that tries to look nice at the cost of being less consistent over all platforms. But functional GUI features are welcomed.
What is “dark plugins”? (2)

- It is not the light way to use Vim
- I have defined it as the dark way

My plugins are dark powered!
Zero generation plugins

- Autocomplpop.vim (acp.vim)
- FuzzyFinder.vim (fzf.vim)
- Shell.vim
Why did I use the plugins?

- I used IDEs in Windows
- I wanted to use IDE features
Why did I quit to use zero generation plugins?

- The plugins are not customizable fully
- Does not exist plugins for me
- I can create the plugins!
1\textsuperscript{st} generation plugins

- vimshell.vim (2009)
- neocomplcache.vim (2009)
- neosnippet.vim (2012)
Vimshell.vim

```vim

1 ~/Code/dotfiles> git status
2 # On branch master
3 # Your branch is ahead of 'origin/master' by 1 commit.
4 #
5 # Changes to be committed:
6 # modified: .gitconfig
7 # modified: .vimrc
8 # modified: vim/bundles.vim
9 #
10 ~/Code/dotfiles> cat ~/.yankring_history_v2.txt
11
12 16.577: sourcing /Users/onenamstartup,v
13 colorscheme base16-tomorrow*Bset background=dark*V
14 colorscheme base16-tomorrow
15 236.144 000.450 000.450: sourcing /usr/local/Cellar/macVim/7.4-70/MacVim.app/Contents/Resources/vim/runtime/syn-
16 411.454 022.914 022.914: sourcing /Users/onenamstartup/.vim/bundle/vim-smartinput/autoload/smartinput.vim
17 487.033 076.336 002.734: sourcing /Users/onenamstartup/.vim/bundle/vim-easyltags/plugin/easyltags.vim,
18 # ctrl-p conflict mapping,v
19 /Users/onenamstartup/.vim/bundle/brolink.vim/plugin/brolink.vim,v
20 :let g:session_autosave = 'no',v
21 vim-eisc,v
22 xolox/vim-shell,v
23 macinkl/vim-neatstatus,v
24 ~/Code/dotfiles> echo

```
Neocomplcache.vim
Neosnippet.vim
1st generation features

- Customizable
- Many functions
- Pluggable
2\textsuperscript{nd} generation plugins

- \texttt{unite.vim} (2010)
- \texttt{vimfiler.vim} (2010)
- \texttt{neobundle.vim} (2011)
- \texttt{neocomplete.vim} (2013)
2nd generation features

- Use if_lua
- Plugins combination
I choose *if_lua* because...

Vim script is too slow!!!
Plugins combination examples

- Unite.vim + vimfiler.vim
- Neocomplete.vim + neosnippet.vim
- Unite.vim + neobundle.vim
3rd generation plugins

- deoplete.nvim (2015)
- denite.nvim (2016)
- deol.nvim (2016)
- dein.vim (2015)
3rd generation features

- Use Vim8/neovim features
- Use Python3 instead
- Asynchronous
Use Vim8/neovim features

- `:terminal`
- Job APIs
- Remote plugins
- `winid`
- And more...

- No more Vim 7!
I have chosen Python3 because...

Vim script is too Slow!!!
Python3 vs if_lua

if_lua

90%

10%

Python3

Lua
For example:

slow!
4th generation plugins

- deoplete.nvim (2019, ver. 5.1~)
- denite.nvim (2019, ver. 3.0~)
- deoppet.nvim (2017)
- defx.nvim (2018)
4th generation features

- Type definitions
- Non default key-mappings
- Remove configuration variables

mappings \[\rightarrow\] variables
Why is type definition important?

- It can detect some bugs
- Detect bugs is important
Why non default key-mappings?(1)

https://github.com/Shougo/defx.nvim/issues/5

Default key mappings #5

Closed kutsan opened this issue on 2 Feb 2018 · 11 comments

kutsan commented on 2 Feb 2018 · edited

I am sorry Shougo but I would like to say your default key mappings of choices were horrible for vimfiler. I know you can change it but it's not a great experience for first-time users. Like, who would have thought to press \texttt{Shift + K} to make a new directory? There is no mnemonic and it's not intuitive. So, if you don't mind, can we discuss this before its implementation? They should be easy to remember and well-thought. For example, \texttt{cf}, \texttt{[c]reate [f]ile or cd}, \texttt{[c]reate [d]irectory} something like that. Although this can be more intuitive with \texttt{dd}, cut file or directory under cursor and \texttt{yy}, copy file or directory under cursor.
Why non default key-mappings?(2)

- No best default key mappings
- Hard to define defaults
- Conflict with other plugins/config

Default mappings are not needed!
Why I have removed configuration variables?

• Functions can keep interfaces in multiple plugins

• Example:
  
call defx#custom#option('_', {})
call denite#custom#option('_', {})
Let’s create your plugins!

- This is just my history.
- You can choose other selection!
Your Vim is your text editor!
The extra: denite.nvim version 3

Shougo
Denite.nvim big changes

- Many issues are reported after upgrade
Denite.nvim version 2

- Steal your input
- Define original mappings
- Similar to ctrlp.vim
Denite.nvim version 2

Prompt
Prompt UI’s disadvantages

- Ignore user’s configuration
- Hard to work asynchronously
- Cannot combine with other plugins

- They fit with my policy???
My plugin development policy

- Customizable
- Combine with other plugins
- Asynchronous if possible
Denite.nvim version 3

- Replace prompt.nvim to normal buffer
- Edit buffer instead of command line
Denite.nvim version 3

Normal buffer

Edit buffer
New UI advantages

• Use user’s configuration
• Easy to work asynchronously
• Can combine with other plugins

• The features fit with my policy!
Denite.nvim version 3 Demo
The extra2: My development note

Shougo
My development style

I don’t use memo plugins!

- Memolist.vim
- QfixHowm
- Vimwiki
- ...

I use paper note for development

• Here
I use the paper note for...

- (Finished) task list
- Technology notes
- Survey for issues
Why do I use paper note?

- Not needed PC!
- Free style
The note is the strongest Vim for me!

• Thank you!
End

• Any questions?