We Have Built Nice Things

Ossification is not Stagination

Wang, Hao (she / her)

Lead Software Architect @ Graveflex | ms-jpq @ github | hola@bigly.dog

Correction

Wrong employer on speakers list

Lead Engineer @ ***Sun****Cloud* not ***Sound****Cloud*

$NEW_JOB

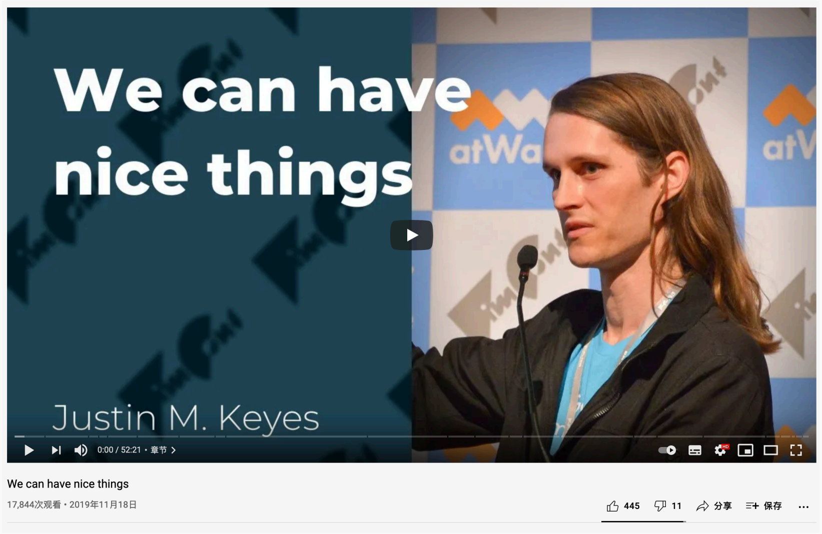
Lead Software Architect @ Graveflex

about:me

- CHADTree
  File manager
- COQ.nvim
  Auto completion
- sad
  Batch regex edits (TUI)
- lua async await
  Concurrency library

Inspiration

We can have nice things - *Justin M. Keyes*



We can have nice things

17,844次观看 • 2019年11月18日

Ossification -> Innovation

Ossofication is not Stagnation

- Nvim has comes with a historied legacy
- Consequences of legacy
- Gifts of legacy

Ossification of Ecosystem

(Existing Plugins)

Vim ”philosophy”

Driver of ossification

- unix-y
- ad hoc
- macro driven
- minimalist
- extensible
- composible
- conservative
- worse is better

Weight of History

- slowiness
- jankiness
- ossification
- interlocking
- beginner hostile
- dx hostile

Interoperation during runtime -> exposure of implementations

Ossification of User Habits

Inertia

Reverence for the *status quo*

People want a faster horse.

Faster horse not good enough to break habits.

Need ***significant value proposition***.

Tolerance for Heresy

User mindset

- Fork the codebase
- Fork the community
- Fork the culture

Have Your Cake and Eat It Too

Don't expose implementations

Interoperability still possible

AOT Static Linking

~~Web~~ Vim Scrapping

"Zero runtime cost" – ~~rust~~ JSON

1. Compile existing plugins for artifacts
2. Dump them into a static file

Ossification -> Unofficial API

"Compiler" can perform validation too!

- CHADTree
  - 3 icon themes
  - 9 colour themes
  - 700+ language colours
- COQ.nvim
  - 13,000+ snippets
  - **zero runtime parse errors**

Dynamic Linking

Can't just AOT everything

What about runtime interface?

Ossification Kills

Notice a pattern here:

- nvim-completion-manager -> ncm2
- neocomplete.vim -> deoplete.nvim -> ddc.vim
- nvim-compe -> nvim-cmp
- completion-nvim -> ~~still ok!~~ *archived*

Too big to rewrite:

- YouCompleteMe
- coc.nvim

Move Fast Without Breaking Things

Protocols Are Meant to Be Ossified

Implementations Are Not

Some protocols are *accidental* or *emergent*

- vim.bo.omnifunc
- :h complete-items
- **LSP**

Zero Public Code

How coq.nvim pulls in thirdparty completion results:

---

1. Third party code **CAN NOT** call into coq.nvim
2. They register callbacks at well known locations (Vim Tradition)
3. Callbacks communicate via LSP

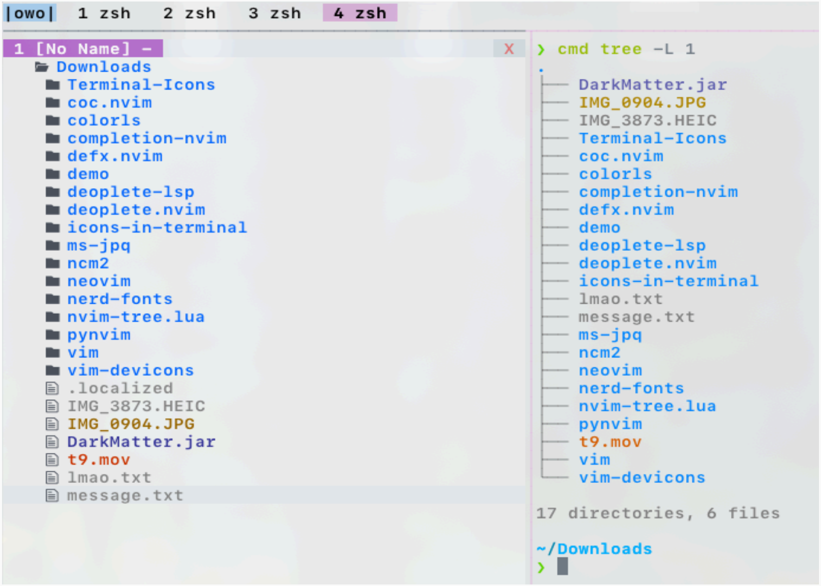
PaaS

What is the platform

- Your app is not the **platform**
- Nvim is the **platform**
- Ecosystem is the **platform**

Nvim :: Browser for the TUI

$LS_COLORS



```
|owo|  1 zsh   2 zsh   3 zsh   4 zsh

1 [No Name] -                                   X    ❯ cmd tree -L 1
  Downloads                                         .
    Terminal-Icons                                  ├── DarkMatter.jar
    coc.nvim                                        ├── IMG_0904.JPG
    colorls                                         ├── IMG_3873.HEIC
    completion-nvim                                 ├── Terminal-Icons
    defx.nvim                                       ├── coc.nvim
    demo                                            ├── colorls
    deoplete-lsp                                    ├── completion-nvim
    deoplete.nvim                                   ├── defx.nvim
    icons-in-terminal                               ├── demo
    ms-jpq                                          ├── deoplete-lsp
    ncm2                                            ├── deoplete.nvim
    neovim                                          ├── icons-in-terminal
    nerd-fonts                                      ├── lmao.txt
    nvim-tree.lua                                   ├── message.txt
    pynvim                                          ├── ms-jpq
    vim                                             ├── ncm2
    vim-devicons                                    ├── neovim
    .localized                                      ├── nerd-fonts
    IMG_3873.HEIC                                   ├── nvim-tree.lua
    IMG_0904.JPG                                    ├── pynvim
    DarkMatter.jar                                  ├── t9.mov
    t9.mov                                          ├── vim
    lmao.txt                                        └── vim-devicons
    message.txt
                                                    17 directories, 6 files

                                                    ~/Downloads
                                                    ❯
```

When to Break Traditions

Worse is Not Always Better

Worse is better for **software** not **wetware**

- Computers fast
- Meatbags slow

~~Implementations Simplicity~~ Humans are not computers

Humans Suck

Designing for meatbags.

Observations

- Computers are *fast*, humans are *slow*
- Humans can't typo good
- Humans have low working memory

Example: Drop requirement for exact prefix match for completions

Data Driven Completion

Link to 2 stage algorithm (Filtering -> Ranking)

Robust against 2 character typos[1]

https://github.com/ms-jpq/coq_nvim/blob/coq/docs/FUZZY.md

Borrow ideas from ML's data processing

Weights, normalization & sigmoid func adjust for user inputs

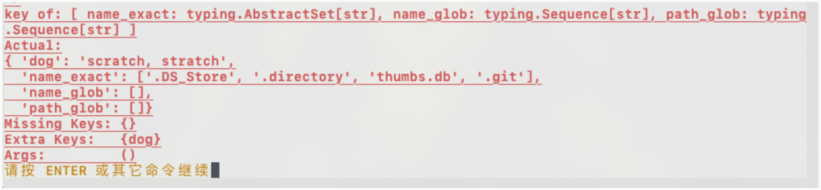
[1] except in first 2 characters

Showcase

(if we have time)

Both::Config Parser

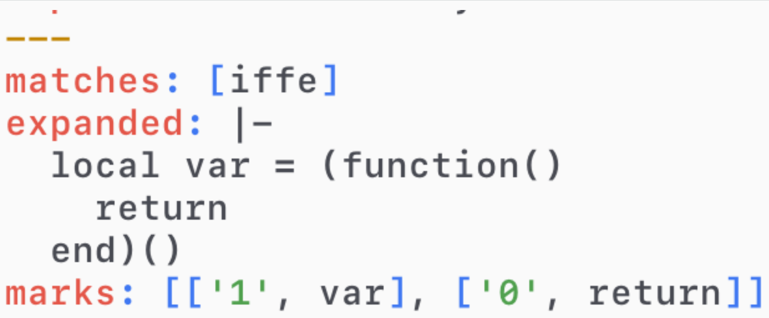
Type checker for plugin configuration



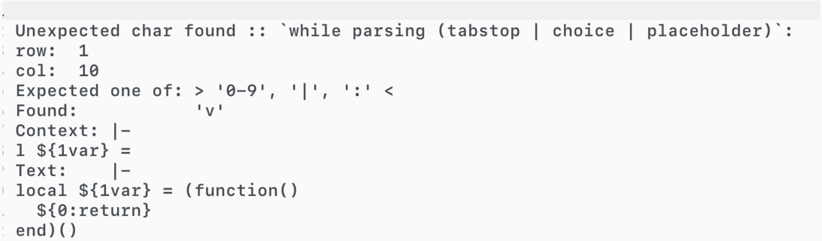
```
key of: [ name_exact: typing.AbstractSet[str], name_glob: typing.Sequence[str], path_glob: typing
.Sequence[str] ]
Actual:
{ 'dog': 'scratch, stratch',
  'name_exact': ['.DS_Store', '.directory', 'thumbs.db', '.git'],
  'name_glob': [],
  'path_glob': []}
Missing Keys: {}
Extra Keys:    {dog}
Args:          ()
请按 ENTER 或其它命令继续
```

Warns for unnecessary ”dog” key

COQ::Snippet REPL



```
---
matches: [iffe]
expanded: |-
  local var = (function()
    return
  end)()
marks: [['1', var], ['0', return]]
```

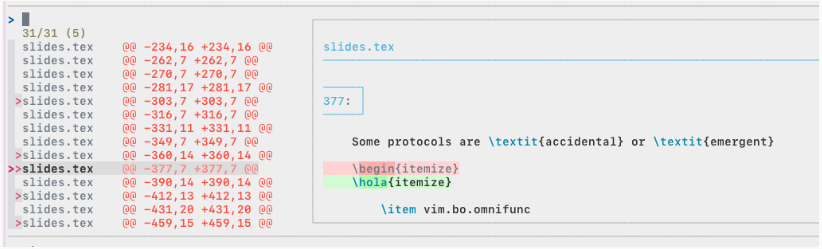


```
Unexpected char found :: `while parsing (tabstop | choice | placeholder)`:
row:  1
col:  10
Expected one of: > '0-9', '|', ':' <
Found:           'v'
Context: |-
l ${1var} =
Text:    |-
local ${1var} = (function()
  ${0:return}
end)()
```

COQ::Stats

	Avg Duration	Q0 Duration	Q50 Duration	Q95 Duration	Q100 Duration
3P	19ms	6ms	18ms	24ms	204ms
BUF	17ms	4ms	17ms	26ms	66ms
LSP	18ms	6ms	18ms	23ms	64ms
PATH	11ms	5ms	9ms	19ms	55ms
SNIP	19ms	9ms	19ms	25ms	54ms
TAG	19ms	9ms	18ms	24ms	66ms
TMUX	23ms	15ms	22ms	29ms	105ms
TS	13ms	5ms	14ms	20ms	53ms

Sad::Substitution



```
> ▌
  31/31 (5)
  slides.tex    @@ -234,16 +234,16 @@     slides.tex
  slides.tex    @@ -262,7 +262,7 @@
  slides.tex    @@ -270,7 +270,7 @@
  slides.tex    @@ -281,17 +281,17 @@     377:
 >slides.tex    @@ -303,7 +303,7 @@
  slides.tex    @@ -316,7 +316,7 @@
  slides.tex    @@ -331,11 +331,11 @@
  slides.tex    @@ -349,7 +349,7 @@         Some protocols are \textit{accidental} or \textit{emergent}
 >slides.tex    @@ -360,14 +360,14 @@
>>slides.tex    @@ -377,7 +377,7 @@          \begin{itemize}
  slides.tex    @@ -390,14 +390,14 @@         \hola{itemize}
 >slides.tex    @@ -412,13 +412,13 @@
  slides.tex    @@ -431,20 +431,20 @@             \item vim.bo.omnifunc
 >slides.tex    @@ -459,15 +459,15 @@
```

Tying the Knot

$$\frac{\text{Technical Factors} \cup \text{Cultural Factors}}{\text{Nice things}}$$

Q & A

---

ms-jpq @ github | hola@bigly.dog