WHO AM I?

- Engineering Manager @MicrosoftFlow
- Author of vim-lsp, asyncomplete.vim, async.vim and quickpick.vim
- Author of typescript-language-server
RENAISSANCE

Renaissance was a period in European history, covering the span between the 14th and 17th centuries and marking the transition from the Middle Ages to modernity.

- wikipedia
VIM RENAISSANCE

Revival and rebirth of vim

- Neovim
- async jobs and channels
- embedded terminal
- floating window
MY VIM JOURNEY
LANGUAGE SERVER PROTOCOL (LSP)
HOW LSP WORKS

LSP Plugin

JSON RPC via stdio/sockets

JavaScript LangServer

TypeScript LangServer

Python LangServer
JSON RPC

REQUEST

```json
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "textDocument/definition",
  "params": {
    "textDocument": {
      "uri": "file:///code/helloworld/main.c"
    },
    "position": {
      "line": 3,
      "character": 12
    }
  }
}
```

RESPONSE

```json
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "uri": "file:///code/helloworld/main.c",
    "range": {
      "start": {
        "line": 0,
        "character": 4
      },
      "end": {
        "line": 0,
        "character": 11
      }
    }
  }
}
```
VIM-LSP

https://github.com/prabirshrestha/vim-lsp

- Modern yet minimal plugin and no legacy.
- Consistent across all language and OS.
- Async only. No freezing the editor
- Works on both vim8 and neovim
- Work on Linux/Mac/Windows all as first class citizen
- It should just work! No more battling with python2 vs python 3.
- Performance!
VIM-LSP V0.1.0

https://github.com/prabirshrestha/vim-lsp/tree/v0.1.0

- 3 years back
- client.vim (~200LOC) and example.vim (~500 LOC)
- tested with langserver-go
Do not merge. This is WIP.

After few months of break from LSP and working on asyncomplete.vim, it is time to give LSP a try again. Hopefully this time we have more servers that works on Linux, Mac and Windows and would not be disappointing like last year.

Privately I have re-implemented LSP (manager not client) multiple times but I have only made the first one open source which is what is seen in example.vim. Throughout this process I have actually learned a lot about Lsp and differences in different server implementations as well as os differences.

Here are some of the goals I have. (There are some missing features but I would like to get something usable that I can use it daily instead of trying to be a full featured LSP plugin first and never actually using it)
CURRENT STATE OF LSP IN VIM

Highest number of LSP plugins compared to any editors out there

- ale.vim
- coc.nvim
- LanguageClient-neovim
- vim-lsc
- vim-lsp
All problems in computer science can be solved by another level of indirection

```vim
function! s:handler(job_id, data, event_type)
    echo a:job_id . ' ' . a:event_type
    echo join(a:data, "\n")
endfunction

if has('win32') || has('win64')
    let argv = ['cmd', '/c', 'dir c:\ /b']
else
    let argv = ['bash', '-c', 'ls']
endif

let jobid = async#job#start(argv, {
    'on_stdout': function('s:handler'),
    'on_stderr': function('s:handler'),
    'on_exit': function('s:handler'),
})
```
Plan for cross platform from Day 1

```vimscript
if has('win32') || has('win64')
  function! lsp#utils#path_to_uri(path) abort
    if empty(a:path)
      return a:path
    else
      " You must not encode the volume information on the path if " present
      let l:end_pos_volume = matchstrpos(a:path, '\c[A-Z]:')[2]
      if l:end_pos_volume == -1
        let l:end_pos_volume = 0
      endif
      return s:encode_uri(substitute(a:path, '\', '/', 'g'), l:end_pos_volume, 'file:///')
  endif
endfunction
else
  function! lsp#utils#path_to_uri(path) abort
    if empty(a:path)
      return a:path
    else
      return s:encode_uri(a:path, 0, 'file://')
  endif
endfunction
endif
```
vim script in most likely not your bottleneck
TIPS FOR WRITING VIM-SCRIPT PLUGINS

If I can do it, you can do it.

60+ contributors of vim-lsp has proven it
Open Opensource! - https://openopensource.org/

Individuals making significant and valuable contributions are given commit-access to the project to contribute as they see fit. This project is more like an open wiki than a standard guarded open source project.

Special thanks to mattin and thomasfaingnaert
VIM’S NEXT REVOLUTION

- ECMAScript/JavaScript interface for vim
- WASM (WebAssembly)
QUESTIONS