Revolutionizing Vim/Neovim Plugin Development

An In-Depth Look into Denops

About me - Plugins and Avatar

- vim-gita / gina.vim / gin.vim
  - Git manipulation plugins
- fern.vim
  - Tree viewer (File explorer)
- suda.vim
  - Allow sudo read/write on Neovim
- mr.vim
  - MRU/MRW/MRR
- kensaku.vim
  - Search Japanese in Roma-ji (migemo)
- denops.vim
  - Vim/Neovim plugin eco-system
- etc.
  - Over 200 repositories for Vim related works

Λlisue
(Ali sue・ありすえ)

About me - My secrets

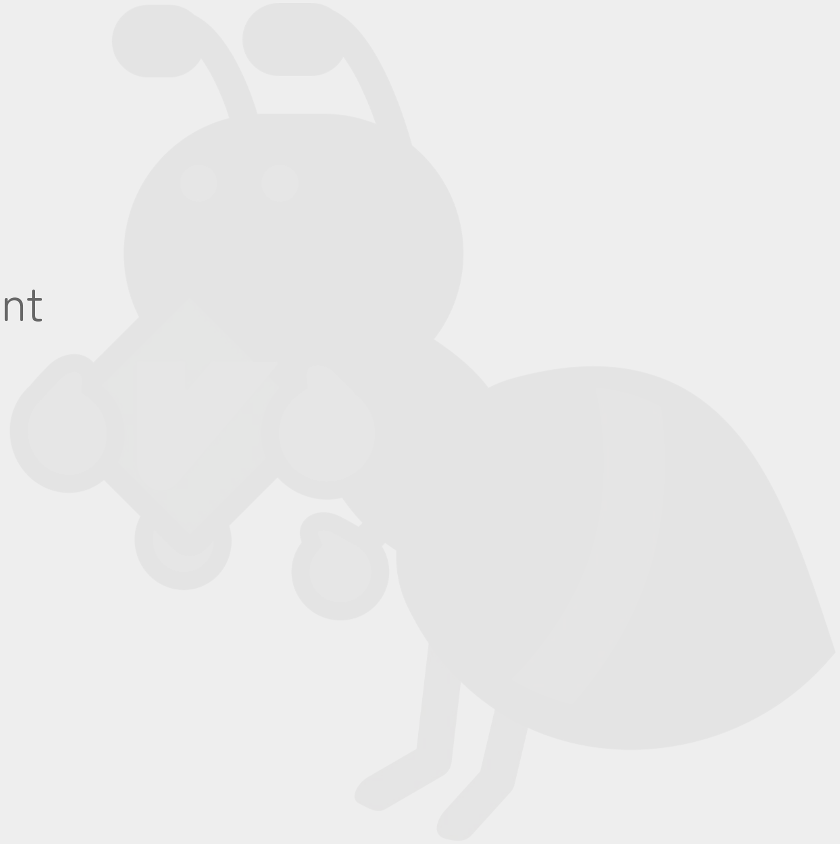
About me - My secrets

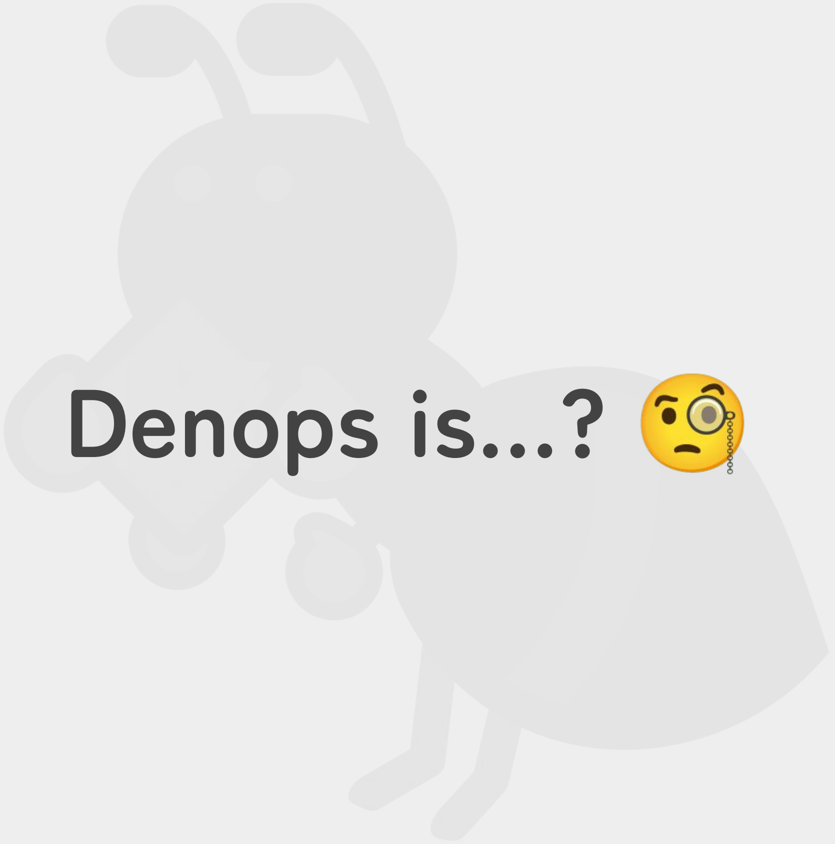
About me - My secrets

About me - My secrets

Agenda

1. Denops is...?
2. Mechanisms
3. Plugin development





Denops is...? 🧐

Denops is...? - Vim/Neovim plugin eco-system

- Developers can utilize TypeScript code to manage Vim/Neovim, allowing for the creation of Vim/Neovim plugins in TypeScript.
- It solely relies on Deno, a single binary, making installation and isolation from the system's Deno effortless.

Denops is...? - Pros. & Cons.

Pros.

- **Good for complex features**
  - High expressive power of JavaScript
  - Robust coding facilitated by TypeScript
- **Unrestricted access to external libraries**
  - No bundle
  - No library version conflicts
- **Enhanced development experience**
  - Supported by built-in LSP
  - Abundance of built-in tools
- **Strong performance**
  - Highly efficient V8 engine
  - Run on external process

Cons.

- **Need external dependencies**
  - Users must install deno and denops.vim
  - This must be done once but still
- **Slow startup**
  - Especially on Windows
  - Shared Server helps it but still
- **Overkill for simple plugins**
  - Extra point of failure
  - RPC-specific circumstances
- **Low name recognition**
  - Only few globally famous plugins
  - Only several talks in Reddit
  - Few articles or documentations

Denops is…? - When to use?

- **You would like to access external resources**
  - API access
  - Database access
  - Process handling
  - OS/File
- **You would like to build a plugin with complex features**
  - Mathematical simulation
  - Graph calculation
  - Image manipulation
- **You would like to handle massive data**
  - List / Streaming (Completion, Fuzzy Finder, etc.)
  - Time series (Metrics, etc.)
  - File analysis (Log data, PDF, etc.)

Denops is...? - When to use?

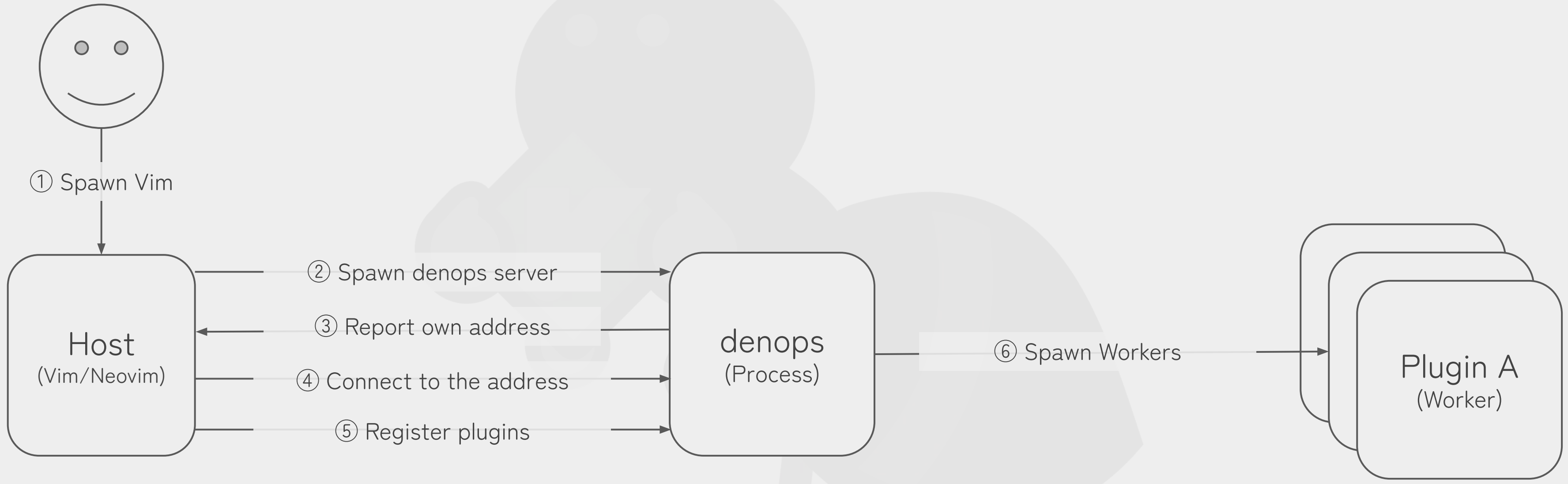
When

- **You would like to access external resources**
  - API access
  - Database acces
  - Process handlin
  - OS/File
- **You would like to build a plugin with complex feature**
  - Mathematical simulation
  - Graph calculation
  - Image manipulation
- **You would like to handle massive data**
  - List / Streaming (Completion, Fuzzy Finder, etc.)
  - Time series (Metrics, etc.)
  - File analysis (Log data, PDF, etc.)

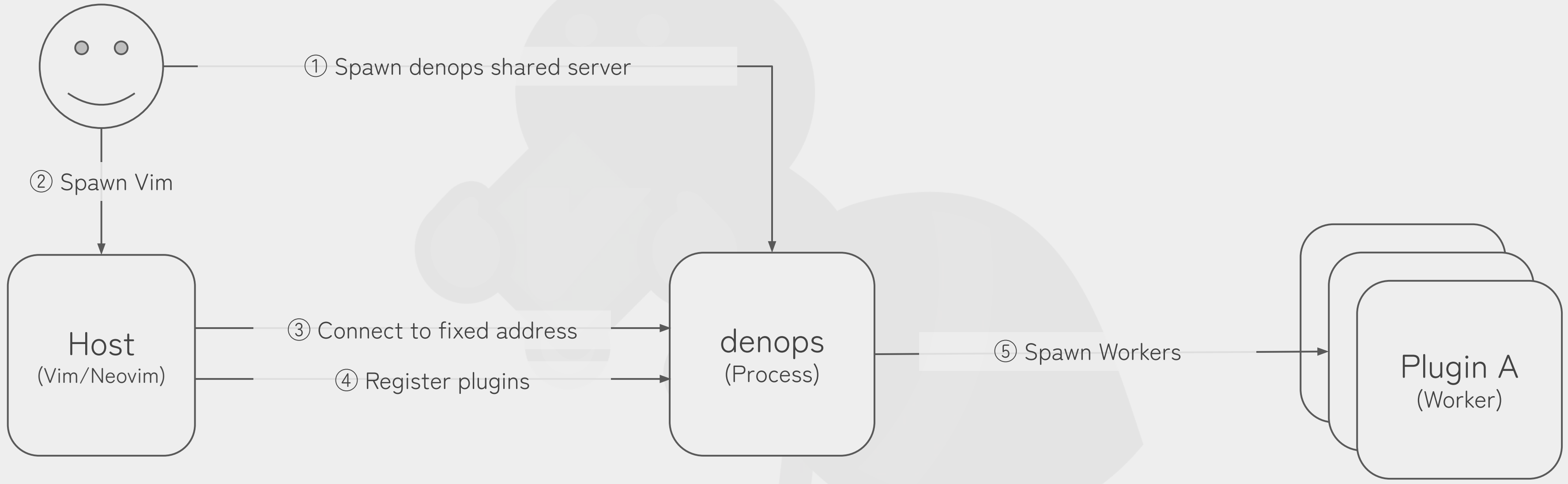
When your ghost whispers!

Mechanisms 💡

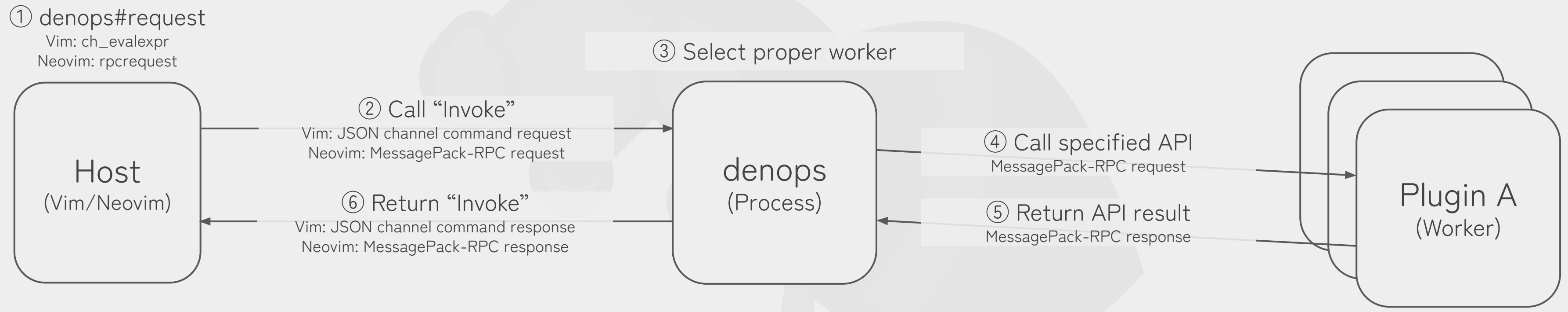
Mechanisms - Start up (Local Server)



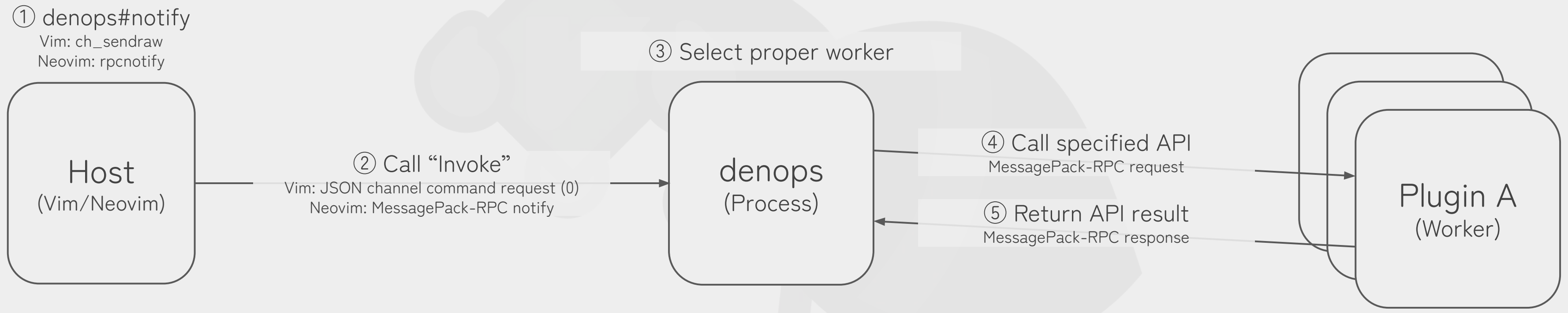
Mechanisms - Start up (Shared Server)



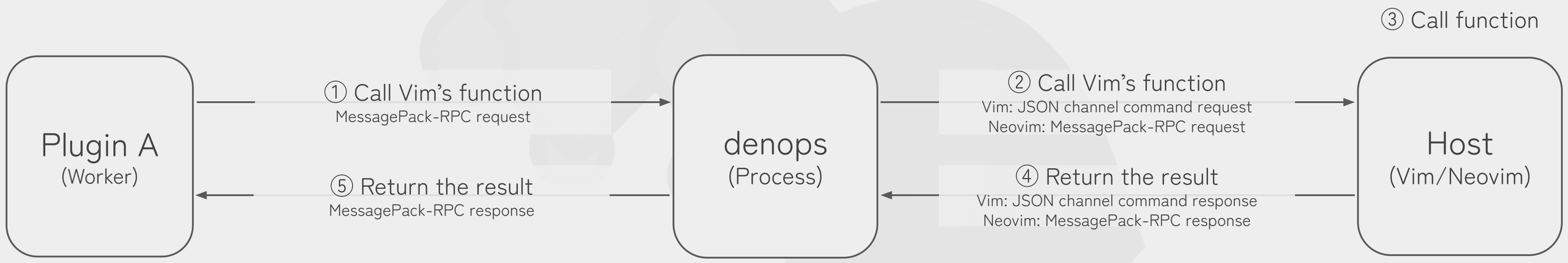
Mechanisms - Plugin API call (request)



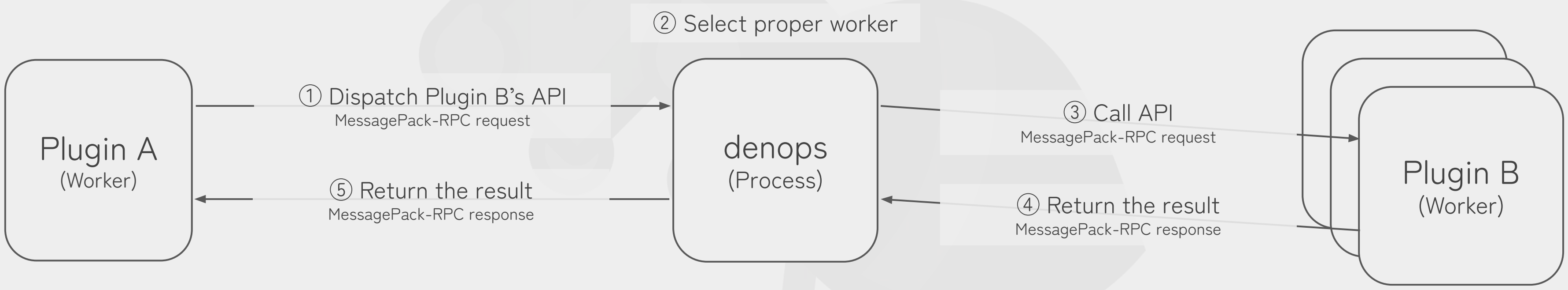
Mechanisms - Plugin API call (notify)

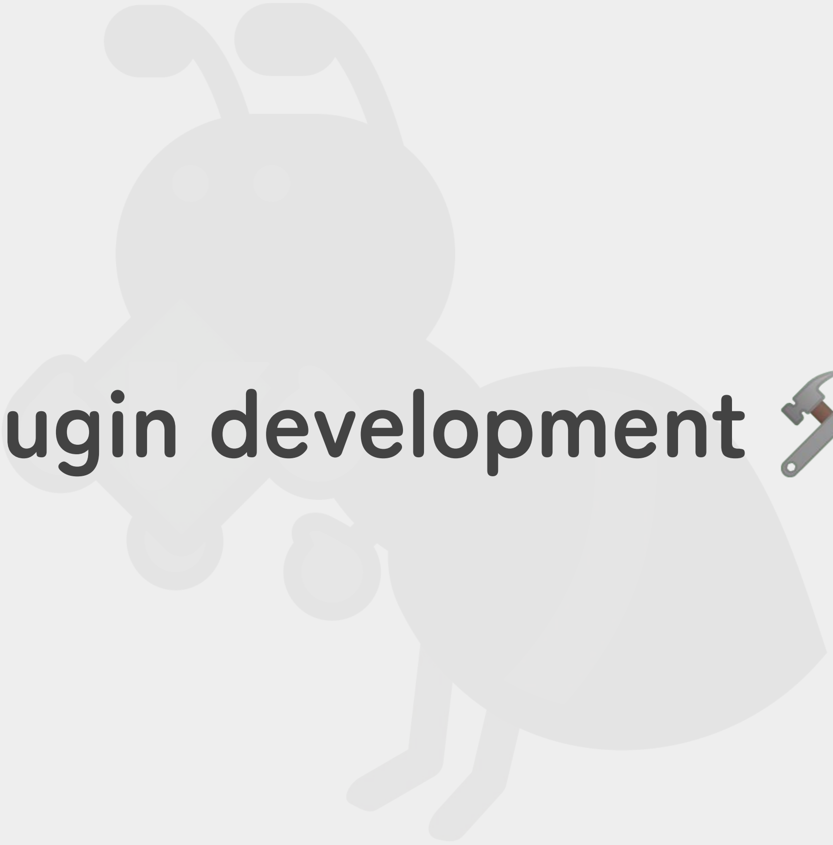


Mechanisms - Call Vim's function



Mechanisms - Dispatch other plugin’s API





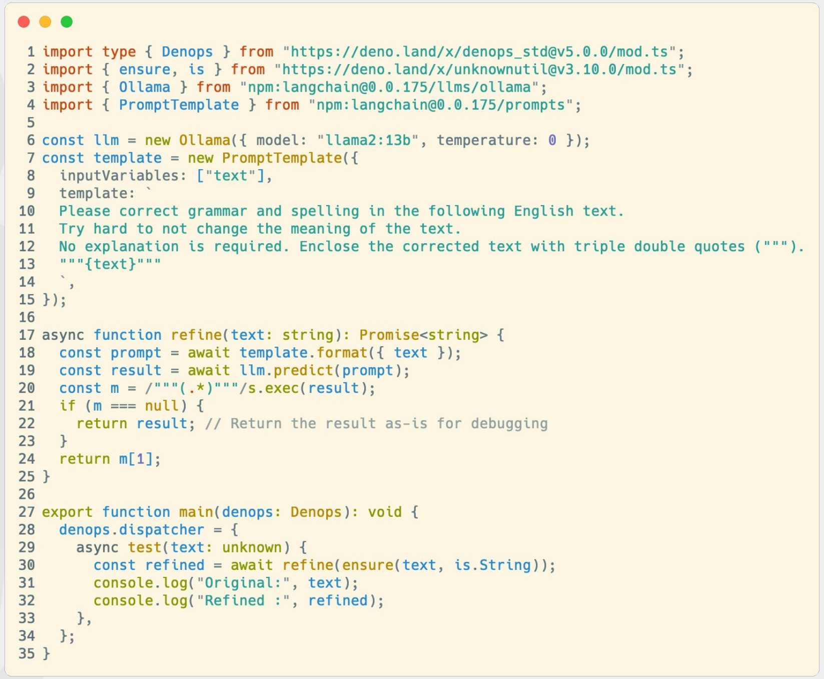
Plugin development 🛠️

Plugin development - Denops meets AI!

Deno's compatibility with Node.js libraries enables developers to utilize the LangChain.js to access LLM from their denops plugin.

We know that Open AI can do it so let's use Ollama (Local LLM) to create a Vim plugin that refines user's English like Grammarly.

First, create a **`test`** API that takes a string and print refined text.

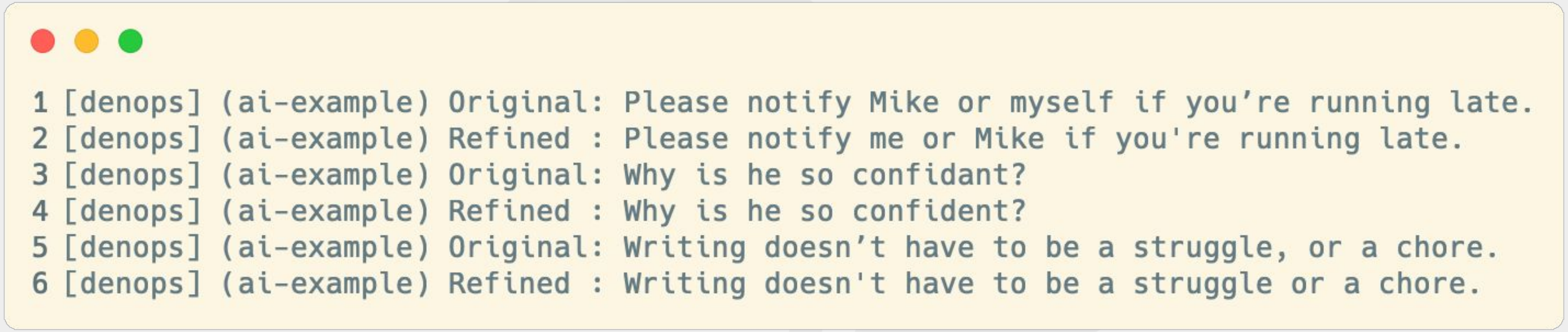


```typescript
import type { Denops } from "https://deno.land/x/denops_std@v5.0.0/mod.ts";
import { ensure, is } from "https://deno.land/x/unknownutil@v3.10.0/mod.ts";
import { Ollama } from "npm:langchain@0.0.175/llms/ollama";
import { PromptTemplate } from "npm:langchain@0.0.175/prompts";

const llm = new Ollama({ model: "llama2:13b", temperature: 0 });
const template = new PromptTemplate({
  inputVariables: ["text"],
  template: `
  Please correct grammar and spelling in the following English text.
  Try hard to not change the meaning of the text.
  No explanation is required. Enclose the corrected text with triple double quotes (""").
  """{text}"""
  `,
});

async function refine(text: string): Promise<string> {
  const prompt = await template.format({ text });
  const result = await llm.predict(prompt);
  const m = /"""(.*)"""/s.exec(result);
  if (m === null) {
    return result; // Return the result as-is for debugging
  }
  return m[1];
}

export function main(denops: Denops): void {
  denops.dispatcher = {
    async test(text: unknown) {
      const refined = await refine(ensure(text, is.String));
      console.log("Original:", text);
      console.log("Refined :", refined);
    },
  };
}
```

The code image is powered by skanehira/denops-sillicon.vim

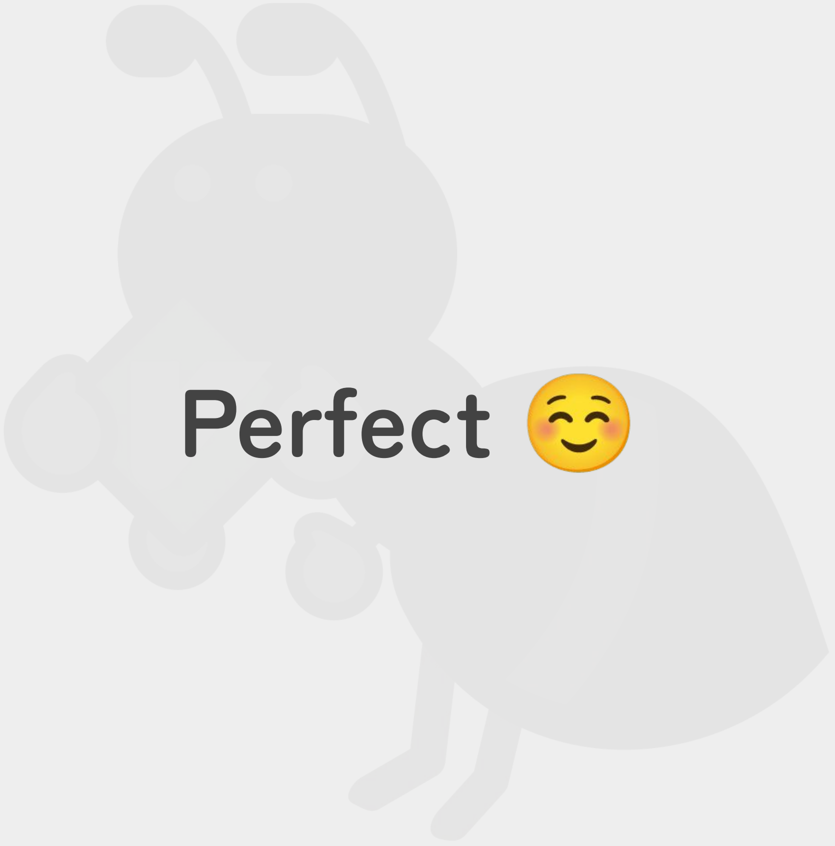
Plugin development - Denops meets AI!

```
:call denops#request("ai-example", "test", ["..."])
```



```
[denops] (ai-example) Original: Please notify Mike or myself if you’re running late.
[denops] (ai-example) Refined : Please notify me or Mike if you're running late.
[denops] (ai-example) Original: Why is he so confidant?
[denops] (ai-example) Refined : Why is he so confident?
[denops] (ai-example) Original: Writing doesn’t have to be a struggle, or a chore.
[denops] (ai-example) Refined : Writing doesn't have to be a struggle or a chore.
```

The sentence above is referred from Grammarly’s official example



Perfect 😊

Plugin development - Improve usability

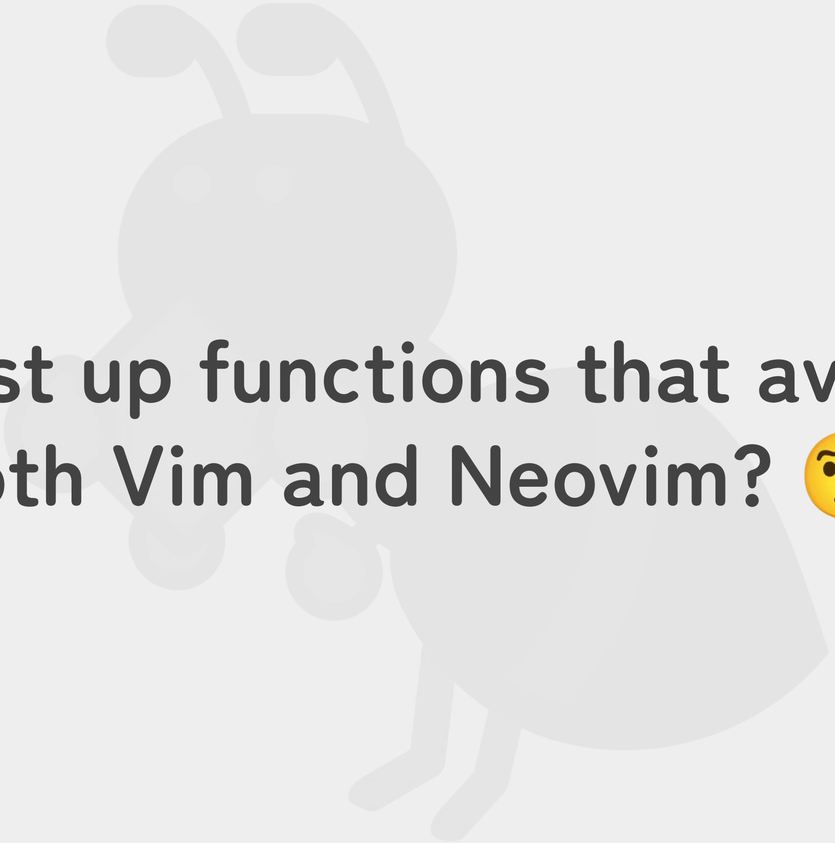
Problems

1. It's too much trouble to type strings as function arguments every time.
2. Results output to the Echo area cannot be copied, etc., making it difficult to use.
3. In actual use, text before modification is not required as an output result if it can be restored to its original state

Design

1. Get visually selected text
2. Get AI refined text of the selected text
3. Replace visually selected text to the AI refined one

Before start, study time 📝



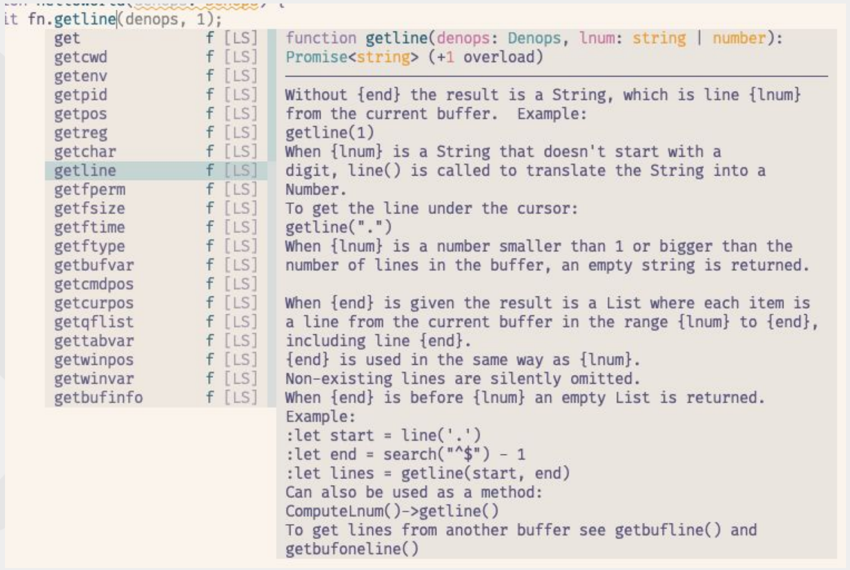
Can you list up functions that available on both Vim and Neovim? 🧐

Plugin development - function module

```
import * as fn from "https://deno.land/x/denops_std@v5.0.1/function/mod.ts";
```

Provides all Vim/Neovim common functions with type hints and document comments

- Generated from help files of supported versions of Vim/Neovim
- Vim specific functions are exposed under `vim/mod.ts`
- Neovim specific functions are exposed under `nvim/mod.ts`



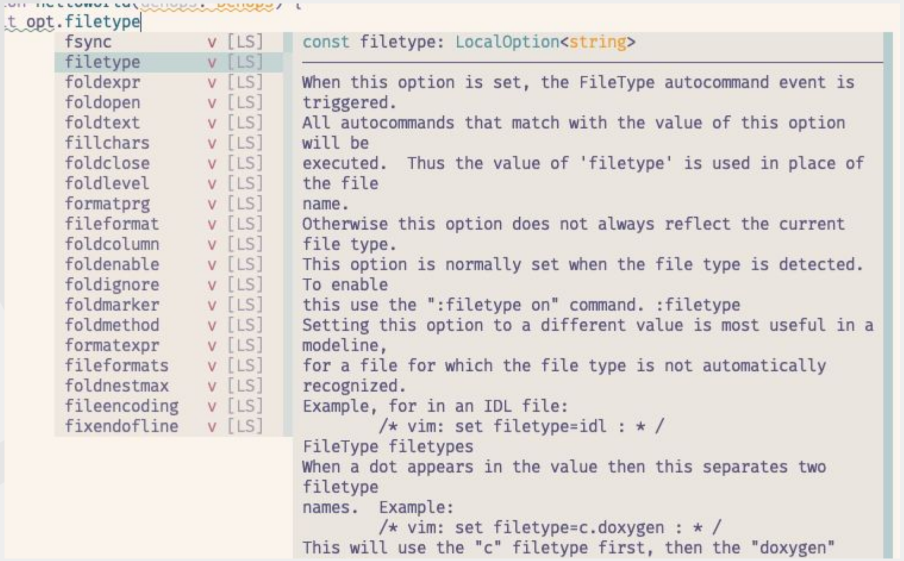
```
it fn.getline(denops, 1);
    get            f [LS]   function getline(denops: Denops, lnum: string | number):
    getcwd         f [LS]   Promise<string> (+1 overload)
    getenv         f [LS]   ──────────────────────────────────────────────────────────
    getpid         f [LS]   Without {end} the result is a String, which is line {lnum}
    getpos         f [LS]   from the current buffer.  Example:
    getreg         f [LS]   getline(1)
    getchar        f [LS]   When {lnum} is a String that doesn't start with a
    getline        f [LS]   digit, line() is called to translate the String into a
    getfperm       f [LS]   Number.
    getfsize       f [LS]   To get the line under the cursor:
    getftime       f [LS]   getline(".")
    getftype       f [LS]   When {lnum} is a number smaller than 1 or bigger than the
    getbufvar      f [LS]   number of lines in the buffer, an empty string is returned.
    getcmdpos      f [LS]
    getcurpos      f [LS]   When {end} is given the result is a List where each item is
    getqflist      f [LS]   a line from the current buffer in the range {lnum} to {end},
    gettabvar      f [LS]   including line {end}.
    getwinpos      f [LS]   {end} is used in the same way as {lnum}.
    getwinvar      f [LS]   Non-existing lines are silently omitted.
    getbufinfo     f [LS]   When {end} is before {lnum} an empty List is returned.
                            Example:
                            :let start = line('.')
                            :let end = search("^$") - 1
                            :let lines = getline(start, end)
                            Can also be used as a method:
                            ComputeLnum()->getline()
                            To get lines from another buffer see getbufline() and
                            getbufoneline()
```

Plugin development - option module

```
import * as opt from “https://deno.land/x/denops_std@v5.0.1/option/mod.ts”;
```

Provides all Vim/Neovim common options with document comments

- Generated from help files of supported versions of Vim/Neovim
- Vim specific options are exposed under `vim/mod.ts`
- Neovim specific options are exposed under `nvim/mod.ts`



```
t opt.filetype
      fsync          v [LS]   const filetype: LocalOption<string>
      filetype       v [LS]   ────────────────────────────────────────────────────
      foldexpr       v [LS]   When this option is set, the FileType autocommand event is
      foldopen       v [LS]   triggered.
      foldtext       v [LS]   All autocommands that match with the value of this option
      fillchars      v [LS]   will be
      foldclose      v [LS]   executed.  Thus the value of 'filetype' is used in place of
      foldlevel      v [LS]   the file
      formatprg      v [LS]   name.
      fileformat     v [LS]   Otherwise this option does not always reflect the current
      foldcolumn     v [LS]   file type.
      foldenable     v [LS]   This option is normally set when the file type is detected.
      foldignore     v [LS]   To enable
      foldmarker     v [LS]   this use the ":filetype on" command. :filetype
      foldmethod     v [LS]   Setting this option to a different value is most useful in a
      formatexpr     v [LS]   modeline,
      fileformats    v [LS]   for a file for which the file type is not automatically
      foldnestmax    v [LS]   recognized.
      fileencoding   v [LS]   Example, for in an IDL file:
      fixendofline   v [LS]           /* vim: set filetype=idl : * /
                              FileType filetypes
                              When a dot appears in the value then this separates two
                              filetype
                              names.  Example:
                                      /* vim: set filetype=c.doxygen : * /
                              This will use the "c" filetype first, then the "doxygen"
```

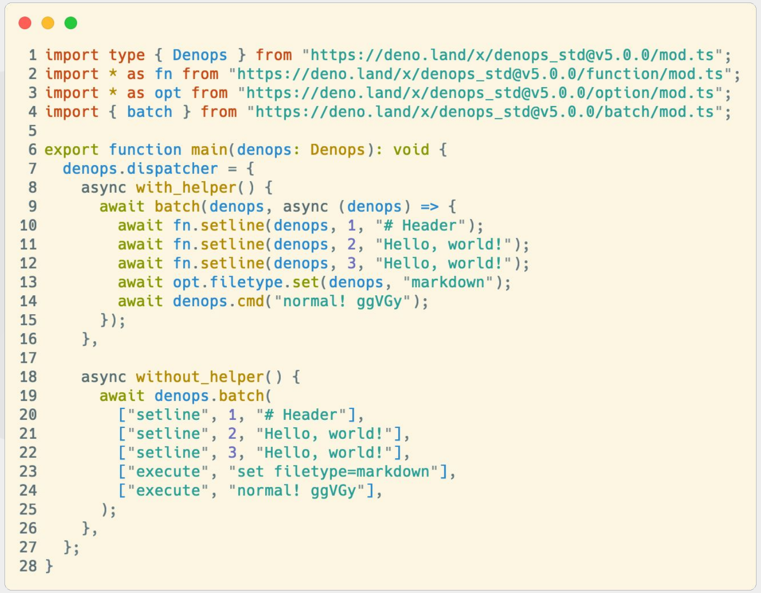
Reducing the number of RPC calls 🏃

Plugin development - batch/batch function

```
import { batch } from “https://deno.land/x/denops_std@v5.0.1/batch/mod.ts”;
```

Helper function for writing codes to execute multiple functions **without** return values in batch

- function, option modules are available
  - Document comments
  - Type annotation & guard
- Nestable
- Return values are not available



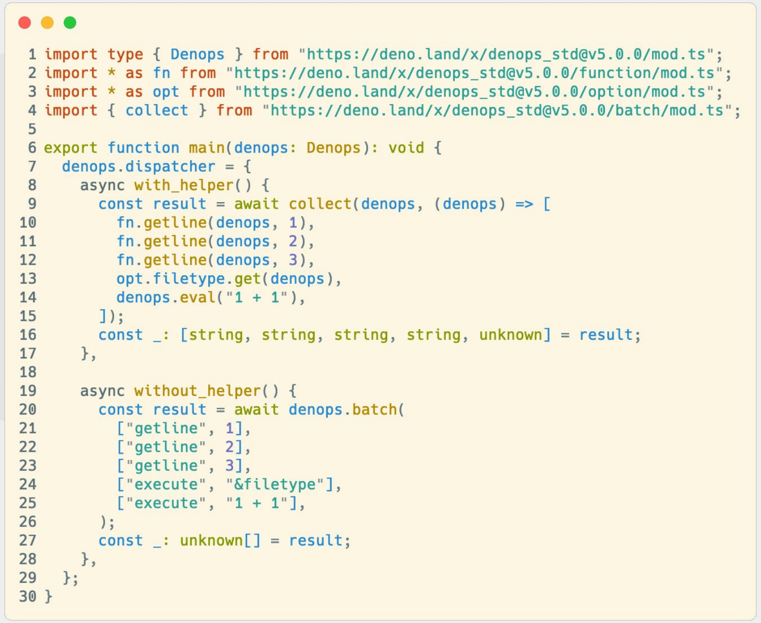
```
import type { Denops } from "https://deno.land/x/denops_std@v5.0.0/mod.ts";
import * as fn from "https://deno.land/x/denops_std@v5.0.0/function/mod.ts";
import * as opt from "https://deno.land/x/denops_std@v5.0.0/option/mod.ts";
import { batch } from "https://deno.land/x/denops_std@v5.0.0/batch/mod.ts";

export function main(denops: Denops): void {
  denops.dispatcher = {
    async with_helper() {
      await batch(denops, async (denops) => {
        await fn.setline(denops, 1, "# Header");
        await fn.setline(denops, 2, "Hello, world!");
        await fn.setline(denops, 3, "Hello, world!");
        await opt.filetype.set(denops, "markdown");
        await denops.cmd("normal! ggVGy");
      });
    },

    async without_helper() {
      await denops.batch(
        ["setline", 1, "# Header"],
        ["setline", 2, "Hello, world!"],
        ["setline", 3, "Hello, world!"],
        ["execute", "set filetype=markdown"],
        ["execute", "normal! ggVGy"],
      );
    },
  };
}
```

Plugin development - batch/collect function

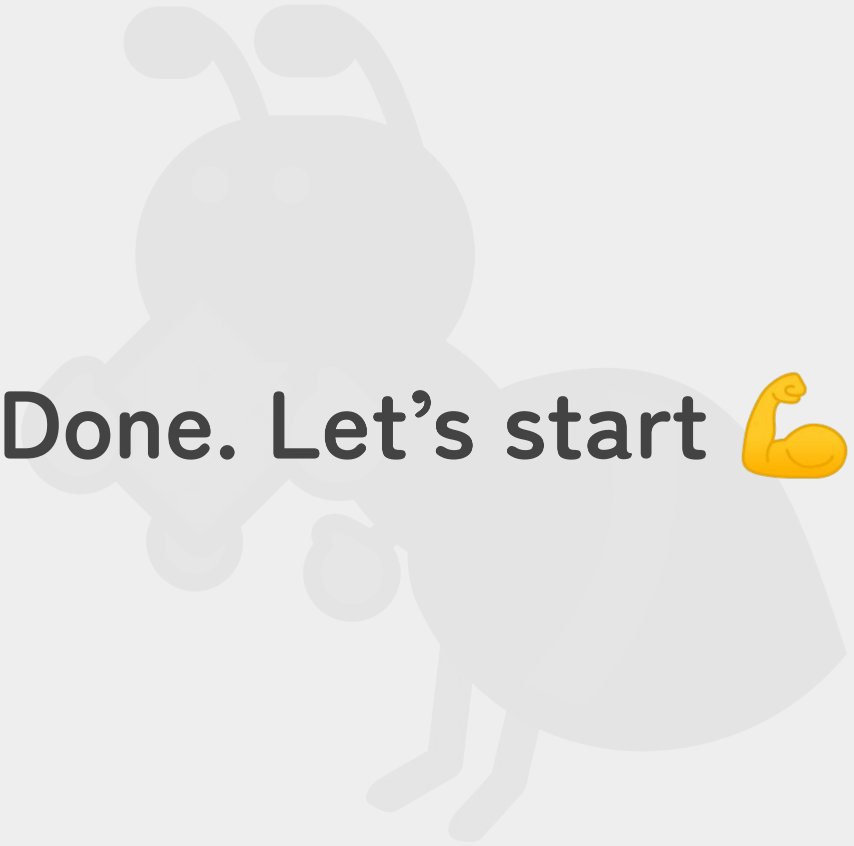
```
import { collect } from “https://deno.land/x/denops_std@v5.0.1/batch/mod.ts”;
```

Helper function for writing code to execute multiple functions **with** return values in batch

- function, option modules are available
  - Document comments
  - Type annotation & guard
- Return values are available
- No branching
- Non nestable



```
import type { Denops } from "https://deno.land/x/denops_std@v5.0.0/mod.ts";
import * as fn from "https://deno.land/x/denops_std@v5.0.0/function/mod.ts";
import * as opt from "https://deno.land/x/denops_std@v5.0.0/option/mod.ts";
import { collect } from "https://deno.land/x/denops_std@v5.0.0/batch/mod.ts";

export function main(denops: Denops): void {
  denops.dispatcher = {
    async with_helper() {
      const result = await collect(denops, (denops) => [
        fn.getline(denops, 1),
        fn.getline(denops, 2),
        fn.getline(denops, 3),
        opt.filetype.get(denops),
        denops.eval("1 + 1"),
      ]);
      const _: [string, string, string, string, unknown] = result;
    },

    async without_helper() {
      const result = await denops.batch(
        ["getline", 1],
        ["getline", 2],
        ["getline", 3],
        ["execute", "&filetype"],
        ["execute", "1 + 1"],
      );
      const _: unknown[] = result;
    },
  };
}
```



Done. Let’s start 💪

Plugin development - get/set selected text

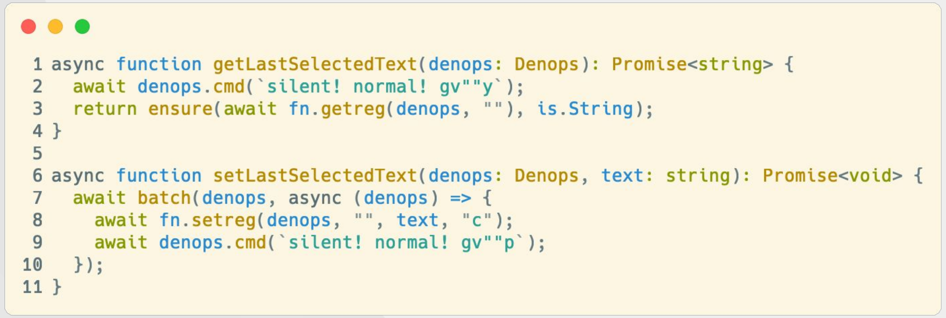
Vim does not have functions to get/set selected text.

How to get a selected text?

1. Select previous selection with **gv**
2. Yank selected text with **“”y**
3. Get yanked text with **getreg** function

How to set a selected text?

1. Yank text with **setreg** function
2. Select previous selection with **gv**
3. Overwrite selected text with **“”p**



```
async function getLastSelectedText(denops: Denops): Promise<string> {
  await denops.cmd(`silent! normal! gv""y`);
  return ensure(await fn.getreg(denops, ""), is.String);
}

async function setLastSelectedText(denops: Denops, text: string): Promise<void> {
  await batch(denops, async (denops) => {
    await fn.setreg(denops, "", text, "c");
    await denops.cmd(`silent! normal! gv""p`);
  });
}
```

Plugin development - get/set selected text

Previous code implicitly overwrote unnamed register.

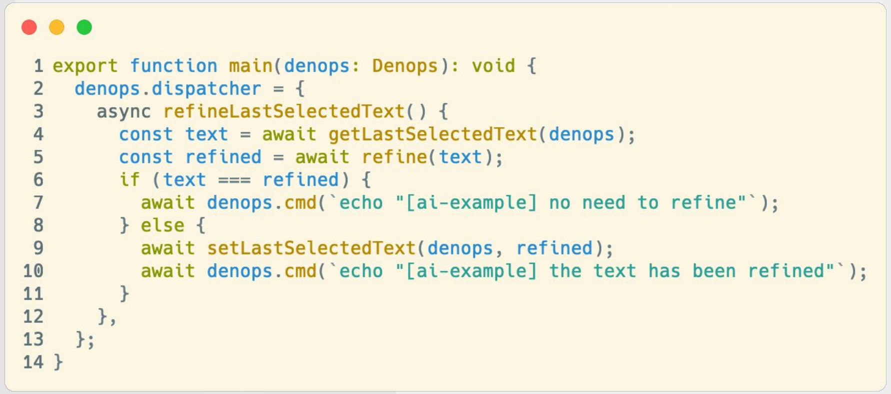
How to avoid this implicit overwrite?

1. Save values in the register to variables
2. Execute an internal function
3. Restore the register with the saved values

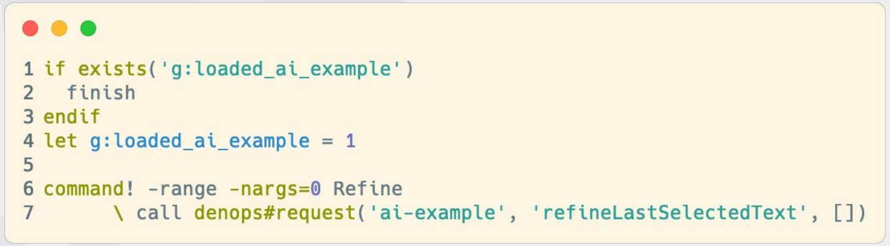
```typescript
async function guardRegister<T>(
  denops: Denops,
  func: () => Promise<T>,
): Promise<T> {
  const [reg, regtype] = await collect(denops, (denops) => [
    fn.getreg(denops, "", 1),
    fn.getregtype(denops, ""),
  ]);
  try {
    return await func();
  } finally {
    await fn.setreg(denops, "", reg, regtype);
  }
}

function getLastSelectedText(denops: Denops): Promise<string> {
  return guardRegister(denops, async () => {
    await denops.cmd(`silent! normal! gv""y`);
    return ensure(await fn.getreg(denops, ""), is.String);
  });
}

function setLastSelectedText(denops: Denops, text: string): Promise<void> {
  return guardRegister(denops, async () => {
    await batch(denops, async (denops) => {
      await fn.setreg(denops, "", text, "c");
      await denops.cmd(`silent! normal! gv""p`);
    });
  });
}
```

Plugin development - Tie up

1. Rewrite **`main`** function to use previously defined functions
   a. refine
   b. getLastSelectedText
   c. setLastSelectedText
2. Add **`plugin/ai-example.vim`** to define command that invoke defined denops API
   a. Refine

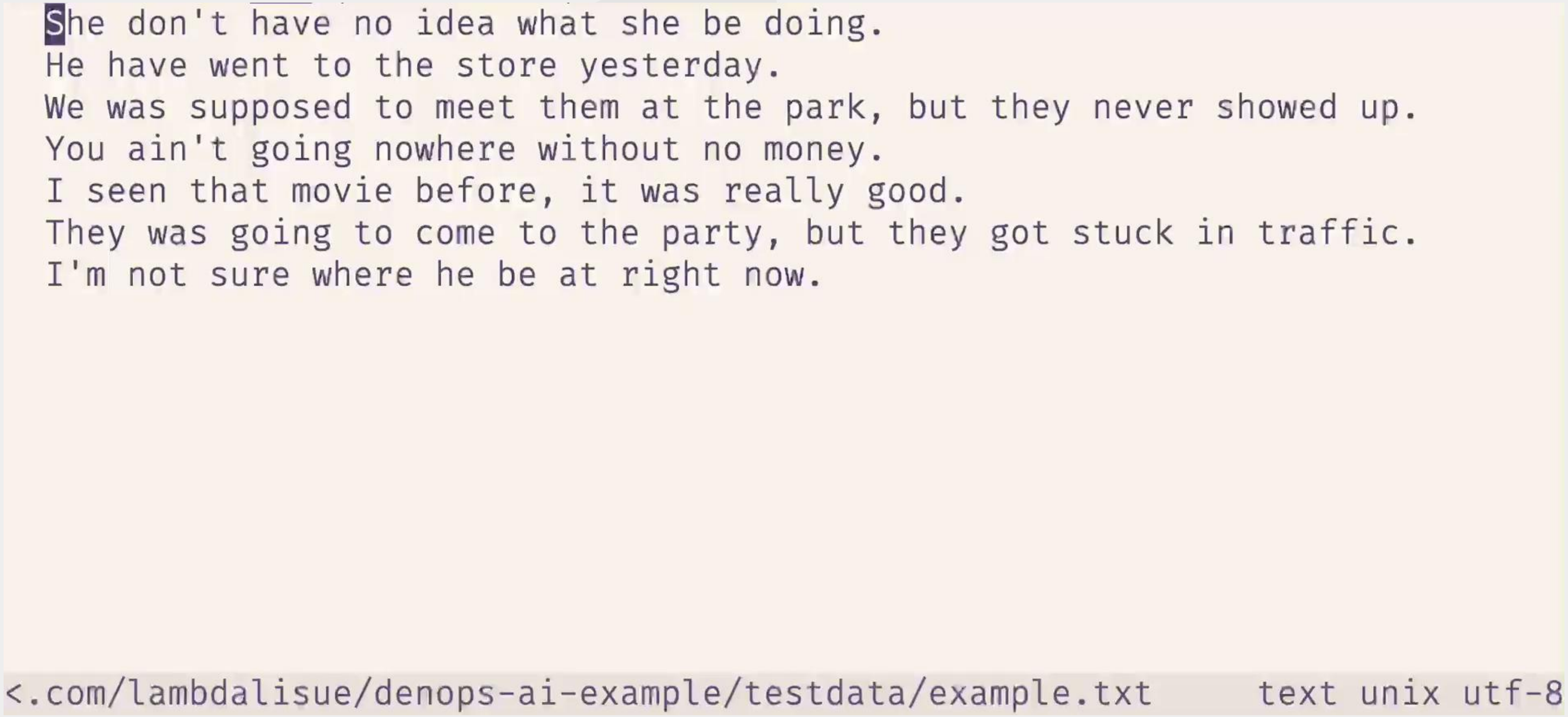


```typescript
export function main(denops: Denops): void {
  denops.dispatcher = {
    async refineLastSelectedText() {
      const text = await getLastSelectedText(denops);
      const refined = await refine(text);
      if (text === refined) {
        await denops.cmd(`echo "[ai-example] no need to refine"`);
      } else {
        await setLastSelectedText(denops, refined);
        await denops.cmd(`echo "[ai-example] the text has been refined"`);
      }
    },
  };
}
```



```vim
if exists('g:loaded_ai_example')
  finish
endif
let g:loaded_ai_example = 1

command! -range -nargs=0 Refine
      \ call denops#request('ai-example', 'refineLastSelectedText', [])
```

Plugin development - Demo



```
  She don't have no idea what she be doing.
  He have went to the store yesterday.
  We was supposed to meet them at the park, but they never showed up.
  You ain't going nowhere without no money.
  I seen that movie before, it was really good.
  They was going to come to the party, but they got stuck in traffic.
  I'm not sure where he be at right now.









<.com/lambdalisue/denops-ai-example/testdata/example.txt     text unix utf-8
```

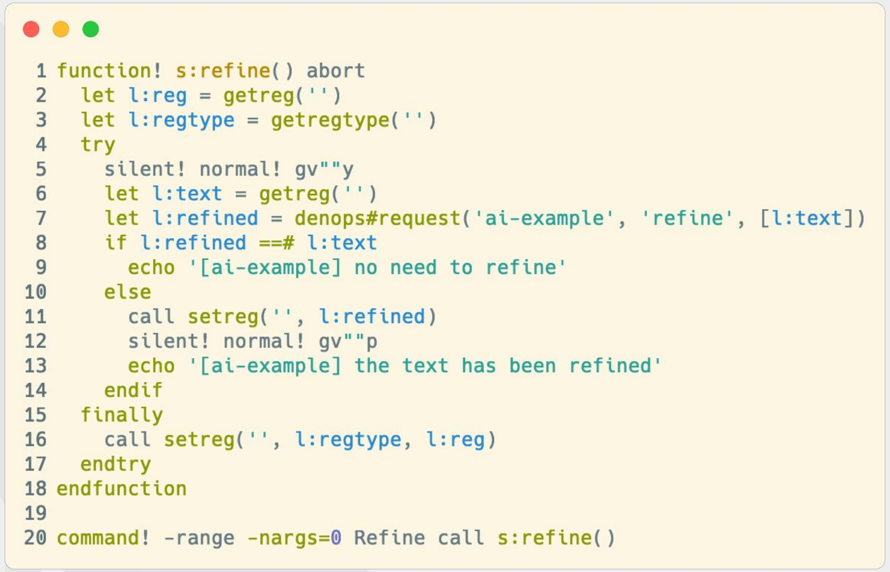
Plugin development - Diff

She don't have no idea what she be doing.	She **doesn't** have **any** idea what she**'s** doing.
He have went to the store yesterday.	He **went** to the store yesterday.
We was supposed to meet them at the park, but they never showed up.	We **were** supposed to meet them at the park, but they never showed up.
You ain't going nowhere without no money.	You **can't go anywhere** without **any** money.
I seen that movie before, it was really good.	I **saw** that movie before**;** it was really good.
They was going to come to the party, but they got stuck in traffic.	They **were** going to come to the party, but they got stuck in traffic.
I'm not sure where he be at right now.	I'm not sure where he **is** right now.

Plugin development - Don’t be afraid Vim script

This example demonstrates writing most of the code in denops. However, **it is not recommended to write all code in denops**.

Developers should not hesitate to use Vim script when necessary. Always consider complexity, performance, and maintainability when deciding which approach to use.



```vim
function! s:refine() abort
  let l:reg = getreg('')
  let l:regtype = getregtype('')
  try
    silent! normal! gv""y
    let l:text = getreg('')
    let l:refined = denops#request('ai-example', 'refine', [l:text])
    if l:refined ==# l:text
      echo '[ai-example] no need to refine'
    else
      call setreg('', l:refined)
      silent! normal! gv""p
      echo '[ai-example] the text has been refined'
    endif
  finally
    call setreg('', l:regtype, l:reg)
  endtry
endfunction

command! -range -nargs=0 Refine call s:refine()
```

Is the code sufficiently straightforward? How about its performance? Does the maintainability appear satisfactory?

Summary

Summary

Denops is venry

Thanks for listening! Want to be my GitHub sponsor?

https://github.com/sponsors/lambdalisue

Thanks for listening! Want to be my GitHub sponsor?

https://github.com/sponsors/lambdalisue

Thanks for listening! Want to be my GitHub sponsor?

https://github.com/sponsors/lambdalisue

Thanks for listening! Want to be my GitHub sponsor?

https://github.com/sponsors/lambdalisue

History 📚

History - In 2016

In 2016, the Language Server Protocol (LSP) was introduced by Microsoft’s VSCode team.

- prabirshrestha/vim-lsp v0.1.0 on Aug 6, 2017
  - Pure Vim script
  - Still active and the top choise for all Vim users
- autozimu/LanguageClient-neovim v0.1.0 on Nov 30, 2017
  - Started with Python + Vim script, then transitioned to Rust + Vim script
  - Not very active recently
- etc.
  - I am certain that there are more clients, but I cannot recall their names at the moment

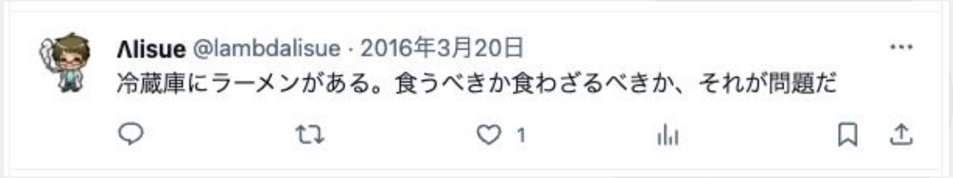
History - In 2018

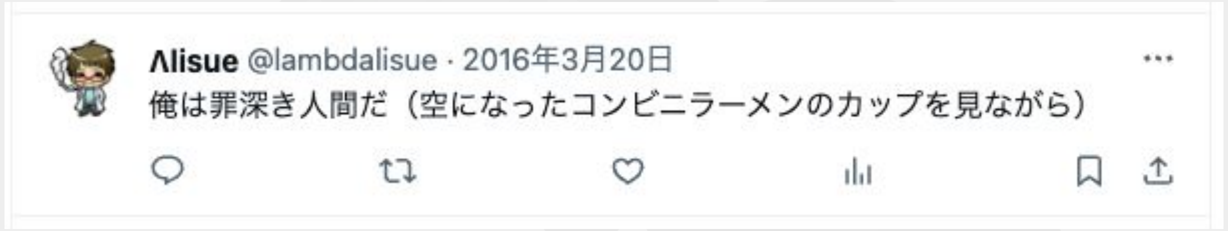
In 2018, neoclide/coc.nvim v0.0.1 was released.

- A Completion & LSP framework that reuses VSCode extensions
  - It is powered by Node.js
- It is compatible with both Vim and Neovim
  - Unlike other ".nvim" suffixed plugins, coc.nvim can function with both
- It is designed to be user-friendly
  - Complexities are skillfully concealed
  - Just install the "coc-xxxxx" plugins and you're good to go
  - Opt-out

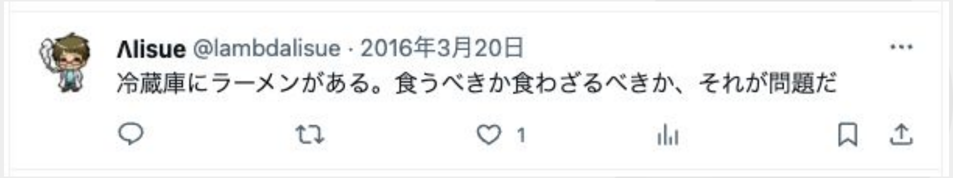
History - Me in 2016-2018

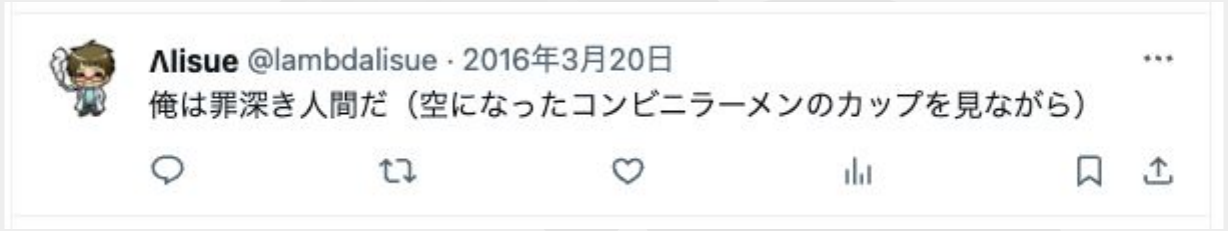
History - Me in 2016-2018

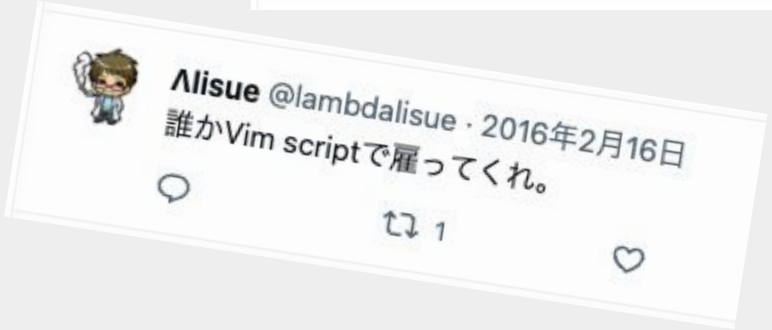

Λlisue @lambdalisue · 2016年3月20日
冷蔵庫にラーメンがある。食うべきか食わざるべきか、それが問題だ
1


Λlisue @lambdalisue · 2016年3月20日
俺は罪深き人間だ（空になったコンビニラーメンのカップを見ながら）

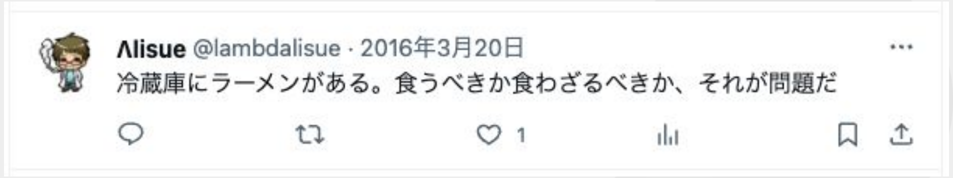
History - Me in 2016-2018

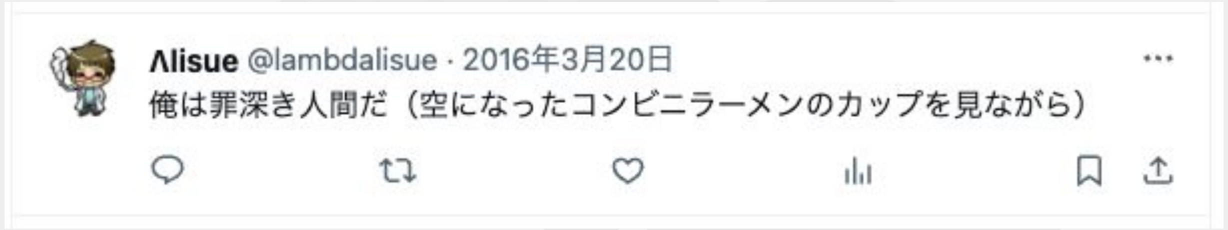

Λlisue @lambdalisue · 2016年3月20日
冷蔵庫にラーメンがある。食うべきか食わざるべきか、それが問題だ
1

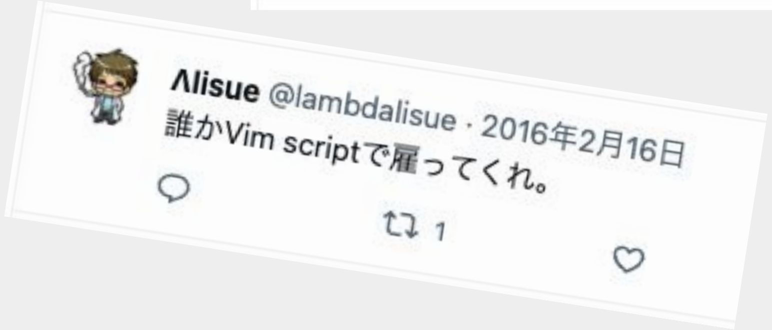

Λlisue @lambdalisue · 2016年3月20日
俺は罪深き人間だ（空になったコンビニラーメンのカップを見ながら）

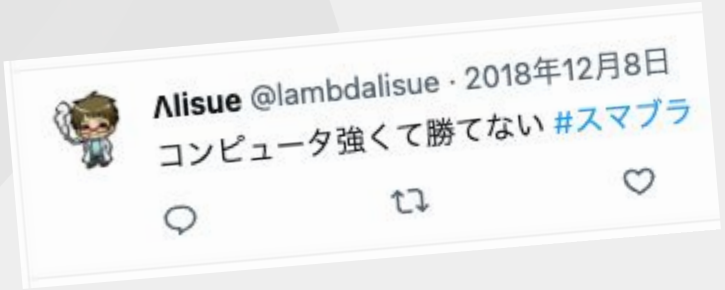

Λlisue @lambdalisue · 2016年2月16日
誰かVim scriptで雇ってくれ。
1

History - Me in 2016-2018

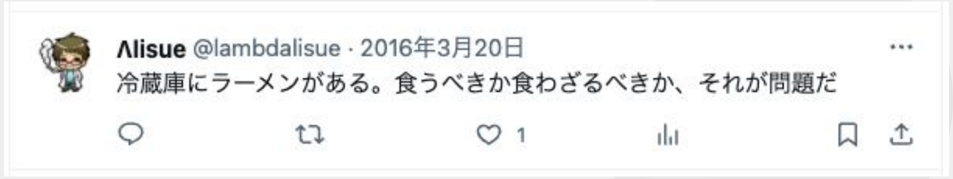

Λlisue @lambdalisue · 2016年3月20日
冷蔵庫にラーメンがある。食うべきか食わざるべきか、それが問題だ
1

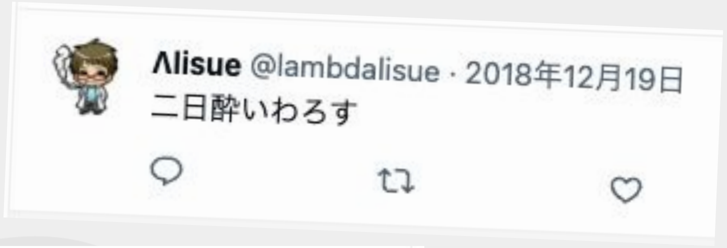

Λlisue @lambdalisue · 2016年3月20日
俺は罪深き人間だ（空になったコンビニラーメンのカップを見ながら）

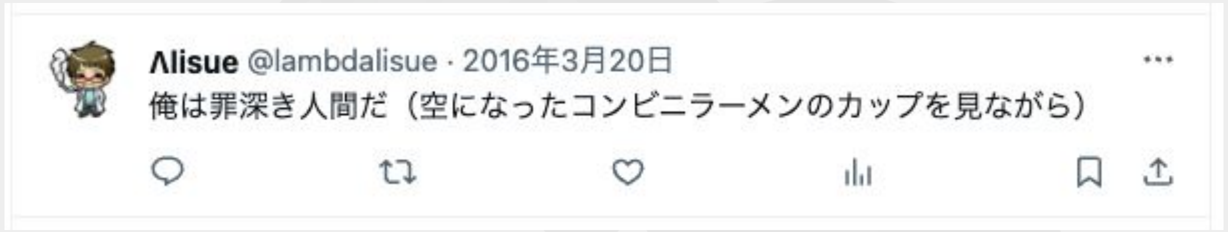

Λlisue @lambdalisue · 2016年2月16日
誰かVim scriptで雇ってくれ。
1

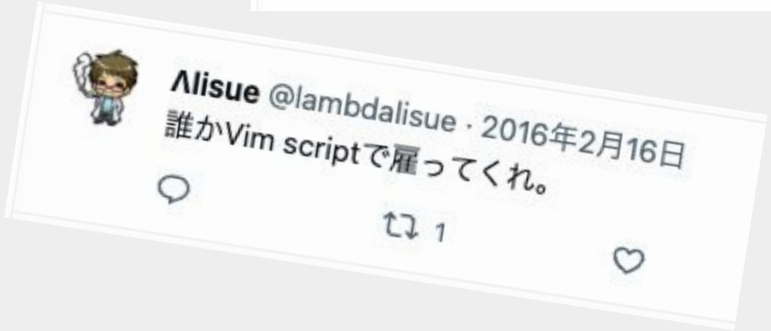

Λlisue @lambdalisue · 2018年12月8日
コンピュータ強くて勝てない #スマブラ

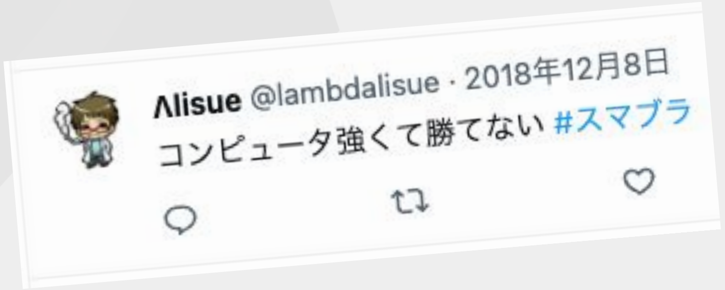
History - Me in 2016-2018


Λlisue @lambdalisue · 2016年3月20日
冷蔵庫にラーメンがある。食うべきか食わざるべきか、それが問題だ
1


Λlisue @lambdalisue · 2018年12月19日
二日酔いわろす


Λlisue @lambdalisue · 2016年3月20日
俺は罪深き人間だ（空になったコンビニラーメンのカップを見ながら）


Λlisue @lambdalisue · 2016年2月16日
誰かVim scriptで雇ってくれ。
1


Λlisue @lambdalisue · 2018年12月8日
コンピュータ強くて勝てない #スマブラ

History - In 2021

Feb 9, 2021, the development of denops.vim has started

- The idea had been brewing since the introduction of Deno in June 2018
- The launch of Deno 1.0 in May 2020 served as the catalyst for the decision to proceed with the project
- The development process progressed swiftly
  - Version 0.1 was released on Feb 14, 2021, merely five days after initiation
  - Version 1.0 followed suit on July 19, 2021, marking a significant milestone achieved within a short five-month period.
- Deno's official designer, @hashrock, crafted the official visual image
  - In the 10th Deno Study Group @ ONLINE
  - Alternative versions can be found at https://github.com/vim-denops/denops-logos

History - In 2023

The latest version of denops.vim is **v5.0.0**.

- Despite the appearance of significant changes in v5, there is only one small and low-impact disruptive change. The rest of the updates mainly involve testing and supporting newer versions of Vim, Neovim, and Deno
- This outcome is a direct consequence of strictly adhering to semantic versioning v2. Frankly, I believe it might have been a bit excessive and possibly even confusing (although I am unable to alter the versioning rules at this point).