From `hjkl` To a platform for plugins

Bram Moolenaar

VimConf Tokyo - November 2018

Plugin support in Vim

The first books about Vim explained all the commands and how to use them.

Plugin support in Vim

The first books about Vim explained all the commands and how to use them.

The latest book tells you what plugins to install [1].

[1] Hands-on Text Processing with Vim 8 - Ruslan Osipov

It all started with Vi

It all started with Vi

You probably forgot (or never knew) what it could do:

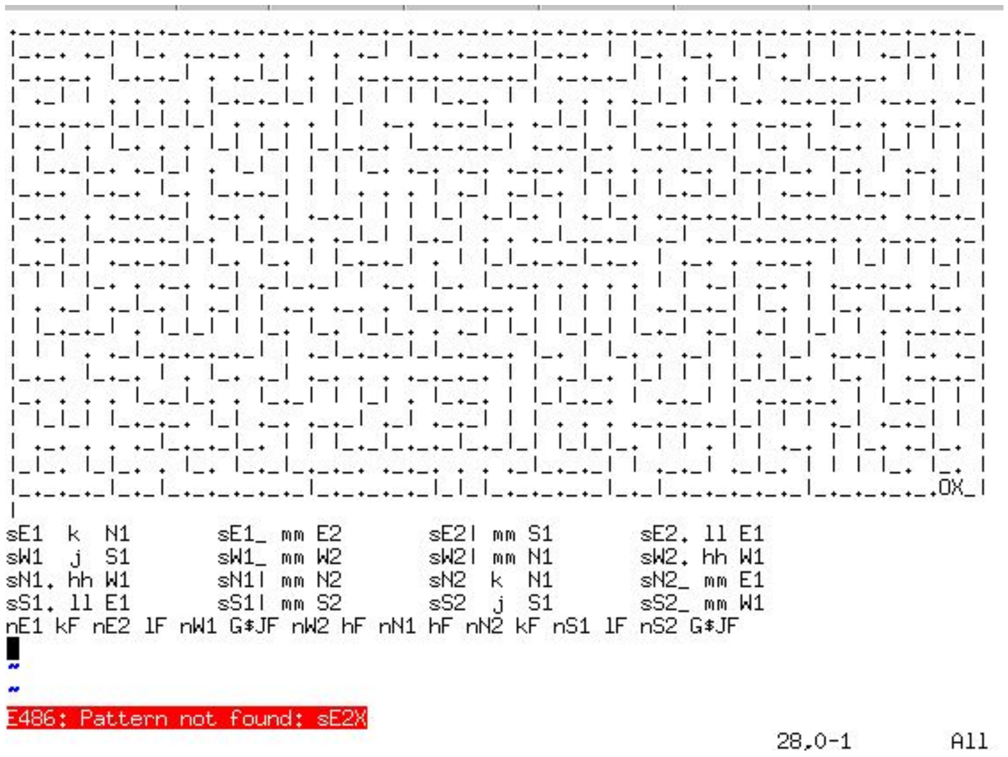
1. `.exrc` for setup - also in current directory
2. Undo - one level
3. Swap file - file bigger than memory, crash recovery
4. Execute a register with `@r`, repeat with `@@`

It all started with Vi

You probably forgot (or never knew) what it could do:

1. `.exrc` for setup - also in current directory
2. Undo - one level
3. Swap file - file bigger than memory, crash recovery
4. Execute a register with `@r`, repeat with `@@`
5. Recursive mappings that can solve a maze

Vi solves a maze



```
[illegible]
|_.__.__|__.__|__.__.__.__.__|__.__.__.__|_|_|__.__.__.__|___|__.__.__.__.__|__.__.__.__.OX_|
|
sE1  k  N1       sE1_ mm E2       sE2| mm S1       sE2. ll E1
sW1  j  S1       sW1_ mm W2       sW2| mm N1       sW2. hh W1
sN1. hh W1       sN1| mm N2       sN2  k  N1       sN2_ mm E1
sS1. ll E1       sS1| mm S2       sS2  j  S1       sS2_ mm W1
nE1 kF nE2 lF nW1 G$JF nW2 hF nN1 hF nN2 kF nS1 lF nS2 G$JF

~
~
E486: Pattern not found: sE2X
                                                  28,0-1        All
```

It all started with Vi

You probably forgot (or never knew) what it could do:

1. `.exrc` for setup - also in current directory
2. Undo - one level
3. Swap file - file bigger than memory, crash recovery
4. Execute a register with `@r`, repeat with `@@`
5. Recursive mappings that can solve a maze
6. Jump around with marks (within one file only)
7. Modelines (with gaping security hole)
8. `:source` to load settings and clever mappings

What Vi could **not** do

1. :if statement, expressions, variables, ...
2. Multiple windows, buffers
3. Remember state between sessions (Viminfo)
4. Highlighting
5. Completion
6. :make and parsing error messages
7. Etc.
8. Etc.

What Vi could **not** do

Relevant improvements in Vim

Why the “im” in Vim means IMproved.

Features to support users better.

(no plugins yet, but used by plugins later)

Autocommands

Added in Vim 4.0

Hooks to allow the user to execute commands depending on the file name.

Autocommands

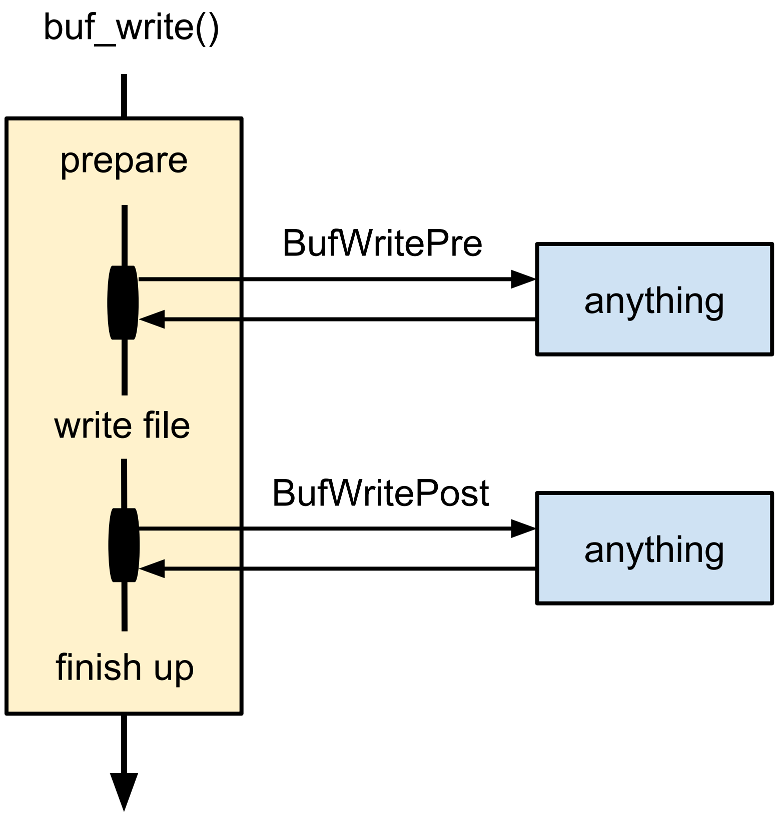
Added in Vim 4.0

Hooks to allow the user to execute commands depending on the file name.

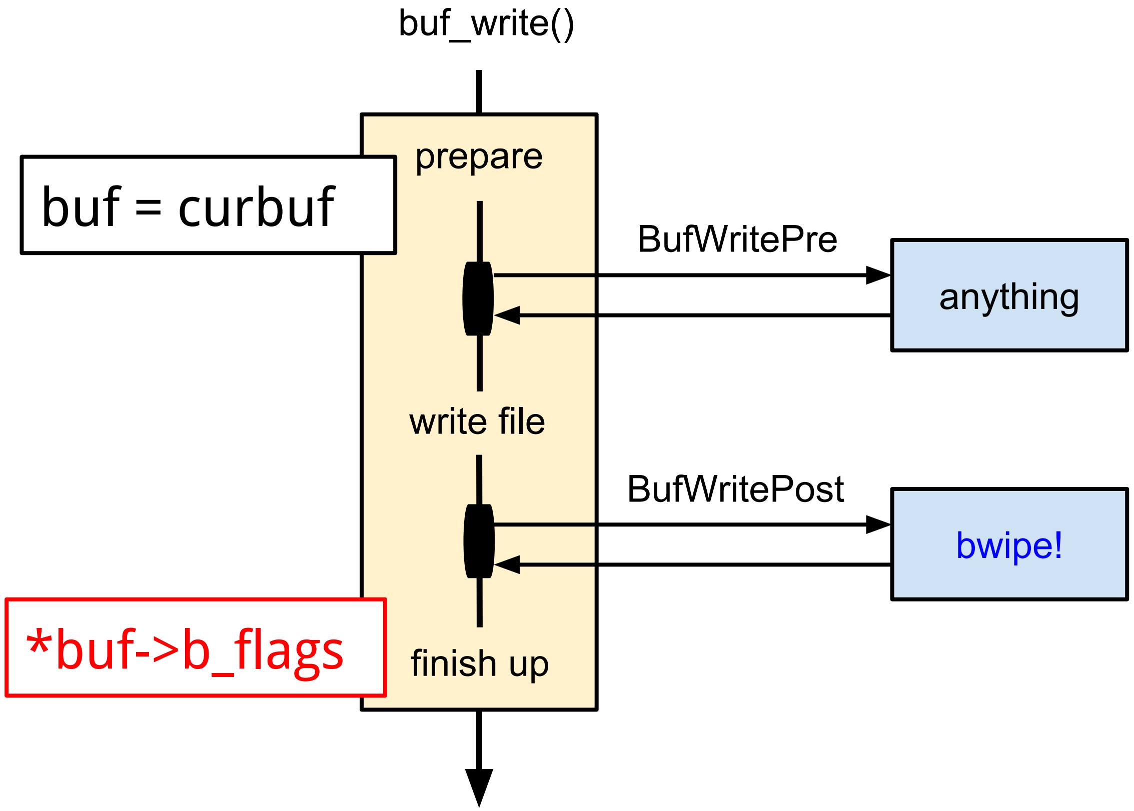
Often used for filetype specific settings:

```
:au BufRead *.c    set tw=78 cindent sw=4
:au BufRead *.java source ~/.vim/java.vim
```

Autocommands



Autocommands



Autocommands

Current solutions to prevent a crash:

1. Allow the command, check buffer pointer is still valid

Autocommands

Current solutions to prevent a crash:

1. Allow the command, check buffer pointer is still valid
2. Disallow the command (with `buf->b_locked`)

Autocommands

Current solutions to prevent a crash:

1. Allow the command, check buffer pointer is still valid
2. Disallow the command (with `buf->b_locked`)
3. Instead of actually deleting the buffer, take it out of the buffer list and only free the memory when the reference count goes to zero. (not actually used for buffers currently)

Autocommands

Current solutions to prevent a crash:

1. Allow the command, check buffer pointer is still valid
2. Disallow the command (with `buf->b_locked`)
3. Instead of actually deleting the buffer, take it out of the buffer list and only free the memory when the reference count goes to zero. (not actually used for buffers currently)

Still have to deal with the buffer disappearing...

start of Vim script

Added in Vim 5.0, defines the syntax

:if
:while
expressions
variables (only numbers and strings)
User functions (added in Vim 5.2)

start of Vim script

Added in Vim 5.0

:if
:while
expressions
variables (only numbers and strings)
User functions (added in Vim 5.2)

Automatic memory management:
- Reference counting for most things
- Garbage collection for cycles (later)

start of Vim script

Used for syntax highlighting, later ftplugin and indenting.

start of Vim script

Used for syntax highlighting, later ftplugin and indenting.

Each user still needs to edit their .vimrc to source specific script files and setup autocommands.
A lot of copy - pasting.

plugins!

Added in Vim 6.0

plugins!

Added in Vim 6.0

Drop a script file in the right place and it gets loaded.

plugins!

Added in Vim 6.0

Drop a script file in the right place and it gets loaded.

‘runtimepath’ option - search multiple directories
Load `.../plugins/*.vim` on startup

plugins!

Added in Vim 6.0

Drop a script file in the right place and it gets loaded.

‘runtimepath’ option - search multiple directories
Load `.../plugins/*.vim` on startup

Distributed with Vim initially: explorer, gzip, netrw, rrhelper

improved plugin support

Data types added over the years:

- Number
- Float
- String
- List
- Dict - can be used as an Object
- Funcref, Partial
- Special
- Job
- Channel

improved plugin support

Builtin functions growing over the years:

- Vim 5.0: 28
- Vim 6.0: 119
- Vim 7.0: 213
- Vim 8.0: 350
- Vim now: 402

plugin performance

plugin performance

Profiling plugins:

```
:profile file {pattern}
:profile func {pattern}
```

Slow startup? Find out why:

```
vim --startuptime {fname}
```

plugin performance

How to make plugins faster?

1. Loading time
2. Execution time

plugin performance

How to make plugins faster?

1. Loading time
2. Execution time

Get a faster computer!

plugin loading time

Reduce loading time with autoload (added in Vim 7.0):

- Plugin file defines:
  - user commands
  - mappings
  - autocommands

plugin loading time

Reduce loading time with autoload (added in Vim 7.0):

- Plugin file defines:
  - user commands
  - mappings
  - autocommands
- The main code is under $VIMRUNTIME/autoload/ auto-loaded only when used

plugin loading time

Autoload for the netrw plugin:

Plugin size:	10 Kbyte
Autoload size:	500 Kbyte

plugin loading time

A one-file plugin: only an autoload file.

Example: Vim-plug: drop plug.vim in ~/.vim/autoload/
Then trigger the auto-load from your .vimrc file:

```
call plug#begin()
   Plug 'junegunn/vim-easy-align'
   …
call plug#end()
```

plugin loading time

Can also use optional sub-plugins, e.g.

```
if has(‘win32’)
   call myplug_unix#func()
else
   call myplug_win#func()
endif
```

plugin loading time

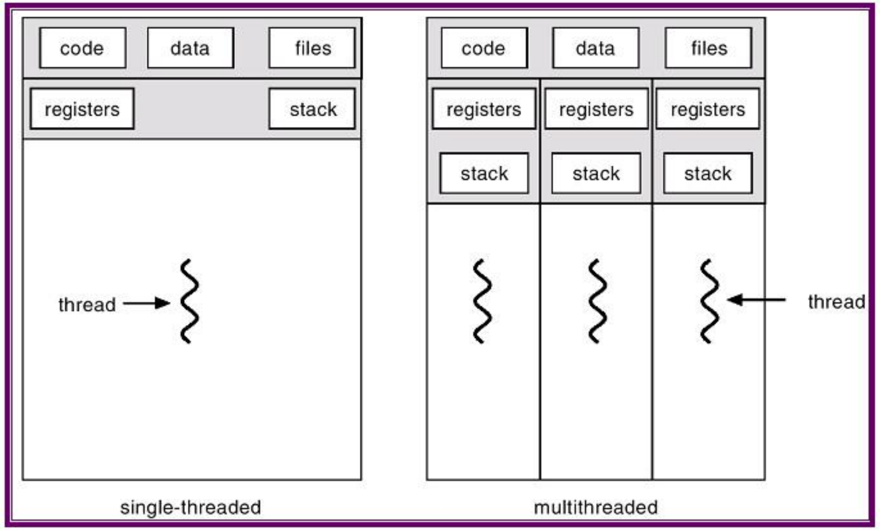
Improve parsing speed. How?

The boring and tedious way:

- Find hot spots with profiling

plugin loading time

Future: Multi-threading: load a plugin in a separate thread.



plugin loading time

Future: Multi-threading: load a plugin in a separate thread.

Must be isolated from the main thread: Cannot add a user command or mapping any time without causing trouble.

Add everything under an autoload namespace, then add that namespace atomically. Need a function to wait on that.

plugin loading time

Future: Multi-threading: load a plugin in a separate thread.

Must be isolated from the main thread: Cannot add a user command or mapping any time without causing trouble.

Add everything under an autoload namespace, then add that namespace atomically. Need a function to wait on that.

Some day...

plugin performance

Instead of trying to make Vim script faster, use an existing language that is fast.

plugin performance

Instead of trying to make Vim script faster, use an existing language that is fast.

- Python - most popular scripting language, but not used much for plugins
- Perl - hardly ever used
- Ruby, Tcl, Scheme - ?
- Lua - Rare

plugin performance

Instead of trying to make Vim script faster, use an existing language that is fast.

- Python - most popular scripting language, still not used much for plugins
- Perl - hardly ever used
- Ruby, Tcl, Scheme - ?
- Lua - Rare

Interfaces can be improved, but does it really help?

plugin performance

Instead of trying to make Vim script faster, use an existing language that is fast.

- Python - most popular scripting language, still not used much for plugins
- Perl - hardly ever used
- Ruby, Tcl, Scheme - ?
- Lua - Rare

Interfaces can be improved, but does it really help?

Instead: Make Vim script better and faster.

plugin performance

plugin execution speed

How to make Vim script run faster?
Avoid parsing the same command line over and over again.

plugin execution speed

How to make Vim script run faster?
Avoid parsing the same command line over and over again.

Lines in a loop or a function:

1. Parse once, convert to intermediate form.
2. Execute the intermediate form, several times

plugin execution speed

How to make Vim script run faster?
Avoid parsing the same command line over and over again.

Lines in a loop or a function:

1. Parse once, convert to intermediate form.
2. Execute the intermediate form, several times

Could also store the intermediate form in a .vic file.
Like in Python .py is compiled into a .pyc file.
Can first do this internally, later decide if storing the intermediate form in a file is useful.

use an intermediate form

Add the intermediate form to each remembered command line:

- original line: string
- command index: enum
- parsed range and count: number, mark, pattern, ...
- parsed arguments: string, expression, ...
- Any remaining text: string

use an intermediate form

Add the intermediate form to each remembered command line:

- original line: string
- command index: enum
- parsed range and count: number, mark, pattern, ...
- parsed arguments: string, expression, ...
- Any remaining text: string

Info depends on the command.
Start with just the line, gradually store more parsed info.

use an intermediate form

Add the intermediate form to each remembered command line:

- original line: string
- command index: enum
- parsed range and count: number, mark, pattern, ...
- parsed arguments: string, expression, ...
- Any remaining text: string

Info depends on the command.
Start with just the line, gradually store more parsed info.

This is a lot of work....

use an intermediate form

plugin management

Manually:
- Get files, drop in $VIMRUNTIME.
- May need to unpack an archive.

Updating: Overwrite the files.

Problem: Files of one plugin mixed with files from other plugins.

plugin management

Added package support in Vim 8.0.

```
:packadd
```

Every plugin lives in a separate directory:

- Clone a plugin from github; update with “git pull”.
- Unpacking an archive; update by deleting + unpacking again.

plugin management

Added package support in Vim 8.0.

`:packadd`

Every plugin lives in a separate directory:

- Clone a plugin from github; update with “git pull”.
- Unpacking an archive; update by deleting + unpacking again.

Only helps for the unpacking and ‘runtimepath’ update.

plugin management

With a plugin manager plugin:

- Vimball (unpacking only)
- Pathogen
- Vundle
- VAM
- vim-plug

plugin management

With a plugin manager plugin:

- Vimball (unpacking only)
- Pathogen
- Vundle
- VAM
- vim-plug

Many depend on github repositories.

- Requires installing git.
- What if the repository is no longer available?
- What if you don’t like github? Or it is no longer free?

plugin dependencies

Still missing: specify plugin dependencies.

Plugins may share common parts, like a library.
How to express these dependencies?

plugin dependencies

Still missing: specify plugin dependencies.

Plugins may share common parts, like a library.
How to express these dependencies?

And then, how to specify versioning?
If it works with version 1.8, does it still work with 2.0?

plugin dependencies

Still missing: specify plugin dependencies.

Plugins may share common parts, like a library.
How to express these dependencies?

And then, how to specify versioning?
If it works with version 1.8, does it still work with 2.0?

Brute force method: Include the other plugin in your plugin, use git to keep it up-to-date. Test that the new version works.

plugin dependencies

Still missing: specify plugin dependencies.

Plugins may share common parts, like a library.
How to express these dependencies?

And then, how to specify versioning?
If it works with version 1.8, does it still work with 2.0?

Brute force method: Include the other plugin in your plugin, use git to keep it up-to-date. Test that the new version works.
Remaining problem: duplication.

plugin dependencies

The Solution

(Proposal)

plugin dependencies

Use a convention that plugin managers should support:

1. A plugin has a dependencies.vim file that lists dependencies with the PluginDepend() function.

plugin dependencies

Use a convention that plugin managers should support:

1. A plugin has a dependencies.vim file that lists dependencies with the PluginDepend() function.
2. A plugin uses PluginLoad() to load a dependency the moment it needs it.

plugin dependencies

dependencies.vim:

```
    let g:myplugin_vimproc = PluginDepend('github', 'Shougo/vimproc.vim')
    let g:myplugin_foolib = PluginDepend('zip',
            \ 'https://www.vim.org/scripts/download_script.php?src_id=1234')
```

Syntax:

```
    PluginDepend({type}, {location})
        {type}   is 'github' or 'zip', later other types
        {location} is the github repo name, URL of the zip file, etc.
```

plugin dependencies

dependencies.vim:

```
    let g:myplugin_vimproc = PluginDepend('github', 'Shougo/vimproc.vim')
    let g:myplugin_foolib = PluginDepend('zip',
        \ 'https://www.vim.org/scripts/download_script.php?src_id=1234')
```

Syntax:

```
    PluginDepend({type}, {location})
        {type}    is 'github' or 'zip', later other types
        {location} is the github repo name, URL of the zip file, etc.
```

myplugin.vim:

```
    call PluginLoad(g:myplugin_vimproc)
```

plugin dependencies

Use a convention that plugin managers should support:

1. A plugin has a dependencies.vim file that lists dependencies with the PluginDepend() function.
2. A plugin uses PluginLoad() to load a dependency the moment it needs it.
3. If a dependent plugin makes an incompatible change, it must use a different name: “v2”, “v3”, etc.

plugin dependencies

Use a convention that plugin managers should support:

1. A plugin has a dependencies.vim file that lists dependencies with the PluginDepend() function.
2. A plugin uses PluginLoad() to load a dependency the moment it needs it.
3. If a dependent plugin makes an incompatible change, it must use a different name: “v2”, “v3”, etc.
4. If the PluginLoad() function is missing the plugin may fall back to a builtin version or omit functionality.

builtin plugin management

A Vim builtin solution would not work differently or faster.

Instead, include a plugin manager with Vim?

How to support plugin authors

What is dearly needed?

How to support plugin authors

What is dearly needed?

And not too much work to implement!

more plugin support

Plugin authors want more functionality:

- Asynchronous processing
- Communicating with a server
- Being able to display more information
- More this...
- More that...

Quite often a very specific piece of functionality.

more plugin support

Plugin authors want more functionality:

- Asynchronous processing
- Communicating with a server
- Being able to display more information
- More this...
- More that...

Quite often a very specific piece of functionality.

How to generalize this, find a common solution?
How to decide what to add next?

more plugin support

Vim 8.0: Jobs and channels

Clear need from plugin authors for asynchronous support.

General idea: Be able to run a process and communicate with it.

- Process may run once (“:make”) or function as a daemon.
- Process may run already and serve many Vim instances
- Process may be written in any language - use JSON for portability

more plugin support

How do plugin authors test their plugin?

- Use the assert_ functions.
- Use feedkeys().
- Use screenshot tests (e.g. for highlighting, completion)

Recently added: Tests for indent plugins

plugin support poll

So, what else? Hold a poll!

plugin support poll

Most requested (16 Nov):

193	Popup window to show hints/message and pick an item
179	Store properties with text, used for highlighting et al.
96	Faster Vim script
65	LSP support (native or basic)
47	Popup for command-line mode
46	APIs for project-related information
41	autocommand for Visual selection
32	better API for signs, highlight, digraphs
30	API for the dot command
28	background thread
21	Multiple sign columns

plugin support poll

Most requested (16 Nov):

193	Popup window to show hints/message and pick an item
179	Store properties with text, used for highlighting et al.
96	Faster Vim script
65	LSP support (native or basic)
47	Popup for command-line mode
46	APIs for project-related information
41	autocommand for Visual selection
32	better API for signs, highlight, digraphs
30	API for the dot command
28	background thread
21	Multiple sign columns

plugin support poll

Most requested (16 Nov):

193	Popup window to show hints/message and pick an item
179	Store properties with text, used for highlighting et al.
96	Faster Vim script
65	LSP support (native or basic)
47	Popup for command-line mode
46	APIs for project-related information
41	autocommand for Visual selection
32	better API for signs, highlight, digraphs
30	API for the dot command
28	background thread
21	Multiple sign columns

text properties

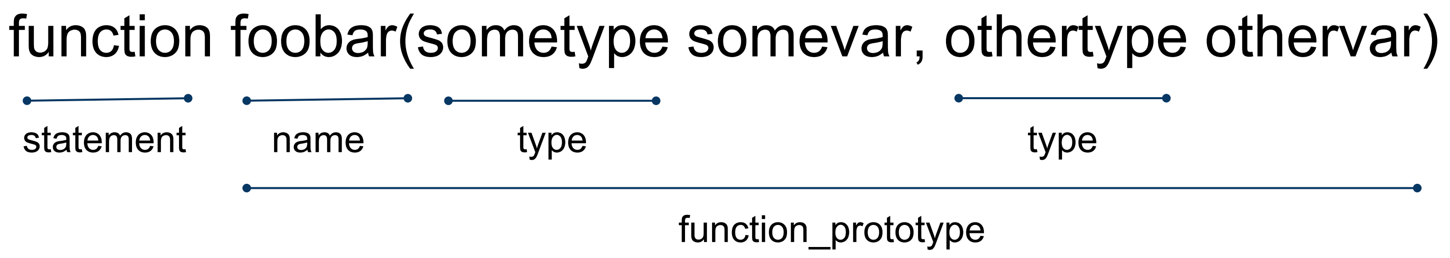
Attach properties to a text region.

- Start and end point
- Moves with the text on inserts and deletes
- Nesting, like syntax regions
- Specify what happens when text is inserted at the start and end point

text properties

Attach properties to a text region.

- Start and end point
- Moves with the text on inserts and deletes
- Nesting, like syntax regions
- Specify what happens when text is inserted at the start and end point



```
function foobar(sometype somevar, othertype othervar)
```

text properties

Can define property with:

- Unique ID
- Property type

Can associate a highlight group with a property type: e.g. Identifier

text properties

Can define property with:

- Unique ID
- Property type

Can associate a highlight group with a property type: e.g. Identifier

Should be possible to do (asynchronous) syntax highlighting with text properties.

text properties

Functions to:

- Set property
- Remove property
- Clear properties for a region
- Get properties at a character
- Find next property
- Find property by unique ID
- Find properties by type

text properties

Implementation: Store with the text line

- No need to update the line number when inserting lines
- Scales to large files

text properties

Implementation: Store with the text line

- No need to update the line number when inserting lines
- Scales to large files

Properties:

- start column, end column
- unique-ID
- property-type
- flags (span lines, what if text is inserted, etc.)

text properties

Implementation: Store with the text line

- No need to update the line number when inserting lines
- Scales to large files

Properties:

- start column, end column
- unique-ID
- property-type
- flags (span lines, what if text is inserted, etc.)

Property type stored as a number (can lookup the name).
Uses about 20 bytes (using 32 bit numbers).

text properties

When making changes: Vim keeps track of changed regions, so plugin can update them.

Plugin can install callback to get notified of changed regions.

When copy-pasting text: properties are lost.
No properties in a register.

popup window

Using text properties for a popup window:

- Defined as a list of lines with text properties.
- Property types map to highlighting

```
call show_popup({
      \ ‘line’: 20,
      \ ‘column’: 38,
      \ ‘text’: [‘foobar(string arg1, float arg2)’],
      \ ‘props’: [{‘col’: 1, ‘len’: 6, ‘type’: ‘functionName’},
      \            {‘col’: 8, ‘len’: 6, ‘type’: ‘varType’},
      \            {‘col’: 21, ‘len’: 5, ‘type’: ‘varType’}],
      \ ‘propdef’: b:popupPropDef,
      \ })
```

popup window

Support multiple (overlapping) windows?
Why not.

popup window

Support multiple (overlapping) windows?
Why not.

Use for a notification:

- Asynchronously show window with text “build done”.
- Remove after a few seconds.

popup window

Support multiple (overlapping) windows?
Why not.

Use for a notification:

- Asynchronously show window with text “build done”.
- Remove after a few seconds.

Use for picking an item:

- Show window where each line is an item
- Let user pick an item
- A bit like confirm() but much nicer

plugin support poll

Most requested:

193	Popup window to show hints/message and pick an item
179	Store properties with text, used for highlighting et al.
96	Faster Vim script
65	LSP support (native or basic)
47	Popup for command-line mode
46	APIs for project-related information
41	autocommand for Visual selection
32	better API for signs, highlight, digraphs
30	API for the dot command
28	background thread
21	Multiple sign columns

plugin support poll

Most requested:

193 Popup window to show hints/message and pick an item
179 Store properties wit xt, used for highlighting et al.
96 Faster Vim script
65 LSP support (na ic)
47 Popup for com de
46 APIs for proje ation
41 autocomma on
32 better API raphs
30 API for t
28 backgr
21 Multiple sign columns

The end

Questions?