vim-easymotion *by* EASYMOTION

4869 ★ 1978

Vim motions on speed!
(/plugin/vim-easymotion-fearless)

CREATED
4 years ago

UPDATED
24 days ago

VIM.ORG

GITHUB (HTTPS://GITHUB.COM/EASYMOTION

? Uncategorized (/?q=cat:uncategorized)

EDIT TAGS

INSTALL FROM

Vundle
NeoBundle
VimPlug
Pathogen

Place this in your `.vimrc`:

```
Plugin 'easymotion/vim-easymotion'
```

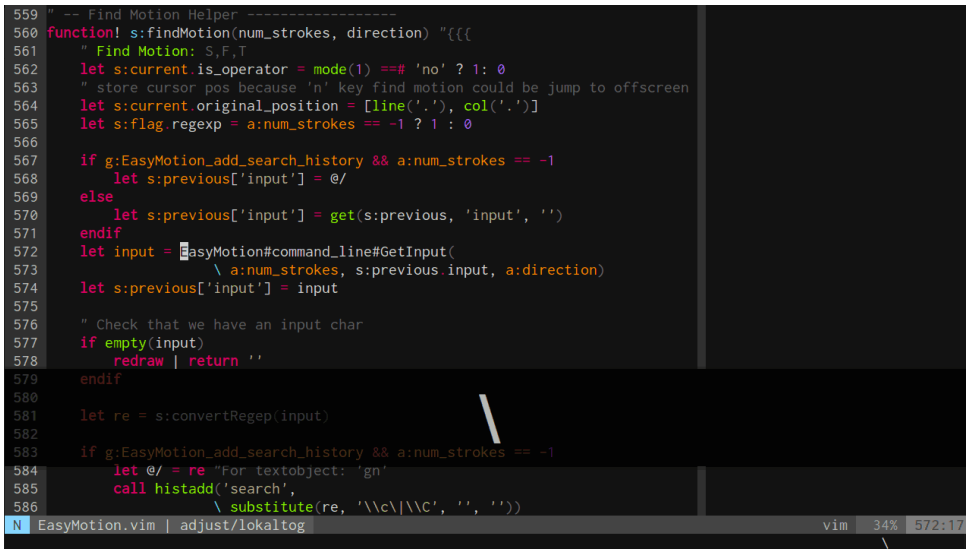
... then run the following in Vim:

```
:source %
:PluginInstall
```

For Vundle version < 0.10.2, replace `Plugin` with `Bundle` above.

VIM MOTION ON SPEED!

build passing (https://travis-ci.org/easymotion/vim-easymotion)



ABOUT THE AUTHORS

Authors

Kim Silkebækken https://github.com/Lokaltog (https://github.com/Lokaltog)
haya14busa https://github.com/haya14busa (https://github.com/haya14busa)

THE EASYMOTION PROJECT, REVIVED!

Starting from version 2.0 haya14busa (https://github.com/haya14busa) will be taking over the project from Lokaltog (https://github.com/Lokaltog). He's improved the default motions, implemented many useful new features, and fixed some bugs.

EasyMotion is now completely

- **Well-behaved**: It's consistent with the default motions of Vim and works well in all modes. And it now supports repeating with the dot operator.
- **Configurable**: You can easily configure its behavior and map it to any key
- **Sophisticated**: Provide flawless, smooth and fast motions with minimal keystrokes

Even though some default behaviors were modified and many new features were added, I carefully considered backward compatibility. So those of you updating from older versions can do so without worry and start benefitting immediately from all the new features!

INTRODUCTION

EasyMotion provides a much simpler way to use some motions in vim. It takes the `<number>` out of `<number>w` or `<number>f{char}` by highlighting all possible choices and allowing you to press one key to jump directly to the target.

When one of the available motions is triggered, all visible text preceding or following the cursor is faded, and motion targets are highlighted.

EasyMotion is triggered by the provided mappings. This readme only covers the basics; please refer to `:help easymotion.txt` (https://github.com/easymotion/vim-easymotion/blob/master/doc/easymotion.txt#L86) to see all the available mappings.

IMPORTANT NOTES

Default bindings

The default leader has been changed to `<Leader><Leader>` to avoid conflicts with other plugins you may have installed. This can easily be changed back to pre-1.3 behavior by rebinding the leader in your vimrc:

```
map <Leader> <Plug>(easymotion-prefix)
```

All motions will then be triggered with `<Leader>` by default, e.g. `<Leader>s`, `<Leader>gE`.

For users of the forked version

SelectLines and SelectPhrase are not actually *motions*, so I've moved them into separate plugins.

- https://github.com/haya14busa/vim-easyoperator-line (https://github.com/haya14busa/vim-easyoperator-line)
- https://github.com/haya14busa/vim-easyoperator-phrase (https://github.com/haya14busa/vim-easyoperator-phrase)

USAGE EXAMPLE FOR THE BASE FEATURES

```
<cursor>Lorem ipsum dolor sit amet.
```

Type `<Leader><Leader>w` (`<Plug>(easymotion-w)`) to trigger the word motion `w`. When the motion is triggered, the text is updated (no braces are actually added, the text is highlighted in red by default):

```
<cursor>Lorem {a}psum {b}olor {c}it {d}met.
```

Press `c` to jump to the beginning of the word "sit":

```
Lorem ipsum dolor <cursor>sit amet.
```

Similarly, if you're looking for an "o", you can use the `f` motion. Type `<Leader><Leader>fo`, and all "o" characters are highlighted:

```
<cursor>L{a}rem ipsum d{b}l{c}r sit amet.
```

Press `b` to jump to the second "o":

```
Lorem ipsum d<cursor>olor sit amet.
```

Jeffrey Way of Nettuts+ has also written a tutorial (http://net.tutsplus.com/tutorials/other/vim-essential-plugin-easymotion/) about EasyMotion.

NEW FEATURES IN VERSION 2.0

Two key highlighting

When EasyMotion runs out of single characters to highlight movement targets, it now shows you immediately the keys you have to press.

In previous versions you could not see the next character you would need to press until you entered the first one. This made movement over long distances less fluid. Now you can see at a glance exactly which characters to select to get to your destination.

Bidirectional motions

All motions now come in a bidirectional variants (e.g. `<Plug>(easymotion-s)`, `<Plug>(easymotion-bd-w)` and so forth). By default, you can already jump forward or backward with `<Leader>s`. A useful trick is to map `nmap s <Plug>(easymotion-s)` to use `s` instead and save one keystroke!

2-character search motion

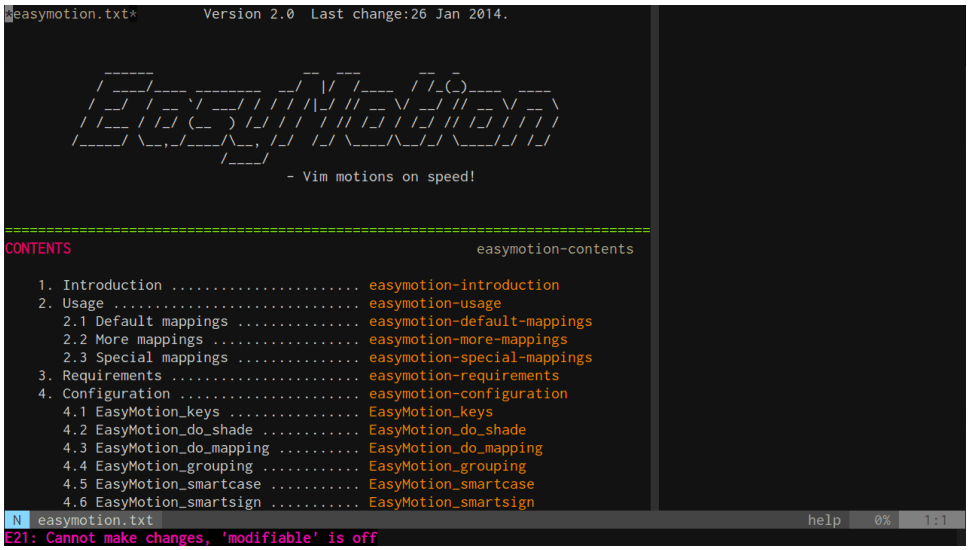
You can now also perform a 2-character search, similar to vim-seek (https://github.com/goldfeld/vim-seek)/vim-sneak (https://github.com/justinmk/vim-sneak) with `<Plug>(easymotion-s2)`. For example you can highlight all words that start with `fu`.


```
" Gif config
nmap s <Plug>(easymotion-s2)
nmap t <Plug>(easymotion-t2)
```

n-character search motion

You can also search for `n` characters, which basically can be used to replace the default search of Vim. It supports incremental highlighting and you can use `<Tab>` and `<S-Tab>` to scroll down/up a page. If you press `<CR>` you get the usual EasyMotion highlighting and can jump to any matching target destination with a single keystroke.

What sounds complicated should become clear if you look at the following examples.



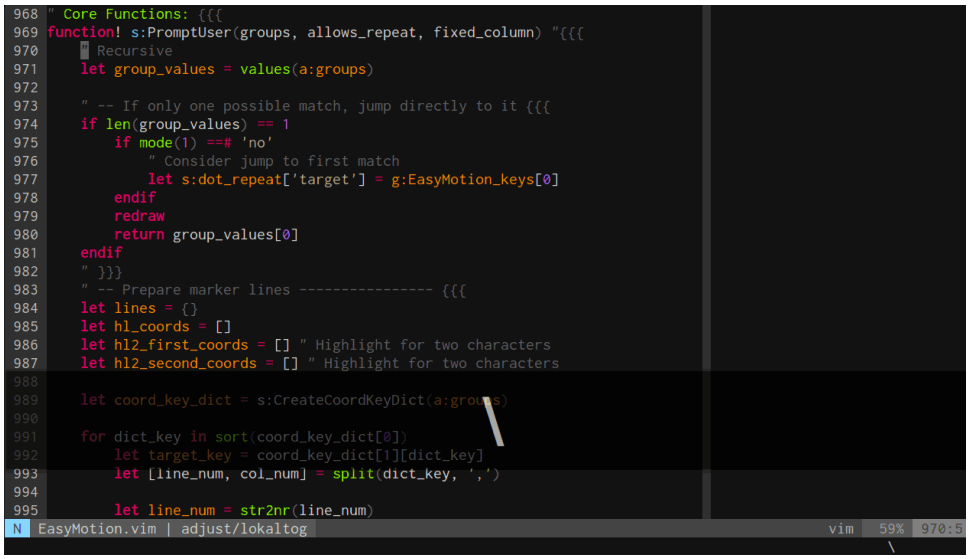

```
" Gif config
map  / <Plug>(easymotion-sn)
omap / <Plug>(easymotion-tn)

" These `n` & `N` mappings are options. You do not have to map `n` & `N` to EasyMotion.
" Without these mappings, `n` & `N` works fine. (These mappings just provide
" different highlight method and have some other features )
map  n <Plug>(easymotion-next)
map  N <Plug>(easymotion-prev)
```

Within line motion

Every motion also has variants that are restricted to just the current line (e.g. `<Plug>(easymotion-sl)`, `<Plug>(easymotion-bd-wl)`, etc...). This can be helpful if you find the full search distracting or slows down vim.

hjkl motions

EasyMotion can be configured to avoid repetitive use of the `h j k` and `l` keys.




```
" Gif config
map <Leader>l <Plug>(easymotion-lineforward)
map <Leader>j <Plug>(easymotion-j)
map <Leader>k <Plug>(easymotion-k)
map <Leader>h <Plug>(easymotion-linebackward)

let g:EasyMotion_startofline = 0 " keep cursor column when JK motion
```

Smartcase & Smartsign

This setting makes EasyMotion work similarly to Vim's `smartcase` option for global searches.

```
let g:EasyMotion_smartcase = 1
```

With this option set, `v` will match both `v` and `V`, but `V` will match `V` only. Default: 0.

```
let g:EasyMotion_use_smartsign_us = 1 " US layout
" or
let g:EasyMotion_use_smartsign_jp = 1 " JP layout
```

This applies the same concept, but for symbols and numerals. `1` will match `1` and `!`; `!` matches `!` only. Default: 0.

Migemo feature (for Japanese user)

```
let g:EasyMotion_use_migemo = 1
```

Easymotion can match multibyte Japanese characters with alphabetical input. For example, `<Leader><Leader>sa` can search 'あ'. This feature doesn't require cmigemo because Easymotion includes regex patterns generated by cmigemo. However, installing `cmigemo` will make 2-character and n-character search motions to also support the migemo feature. Default:0

Repeat motions

Repeat the last motion

`<Plug>(easymotion-repeat)`

Repeat the last find motion

In a find motion (e.g. `<Plug>(easymotion-s)`), type `<CR>` without input characters to find the last motion again.

Jump to next/previous match (even on next/previous page)

- `<Plug>(easymotion-next)`
- `<Plug>(easymotion-prev)`

Support for dot repeat

This requires https://github.com/tpope/vim-repeat (https://github.com/tpope/vim-repeat).

You can use EasyMotion with operators and press `.` to repeat! It is well-behaved, and consistent with the default behavior of Vim.


```
" Gif config

" Require tpope/vim-repeat to enable dot repeat support
" Jump to anywhere with only `s{char}{target}`
" `s<CR>` repeat last find motion.
nmap s <Plug>(easymotion-s)
" Bidirectional & within line 't' motion
omap t <Plug>(easymotion-bd-tl)
" Use uppercase target labels and type as a lower case
let g:EasyMotion_use_upper = 1
 " type `l` and match `l`&`L`
let g:EasyMotion_smartcase = 1
" Smartsign (type `3` and match `3`&`#`)
let g:EasyMotion_use_smartsign_us = 1
```

Installation

Pathogen (https://github.com/tpope/vim-pathogen (https://github.com/tpope/vim-pathogen))

```
git clone https://github.com/easymotion/vim-easymotion ~/.vim/bundle/vim-easymotion
```

Vundle (https://github.com/gmarik/vundle (https://github.com/gmarik/vundle))

```
Plugin 'easymotion/vim-easymotion'
```

NeoBundle (https://github.com/Shougo/neobundle.vim (https://github.com/Shougo/neobundle.vim))

```
NeoBundle 'easymotion/vim-easymotion'
```

Minimal Configuration Tutorial

I recommend configuring and map keys by yourself if you are true Vimmer.

Please do not be satisfied with just installing vim-easymotion, configuring it yourself boost your productivity more and more!

Default `<Leader><Leader>` prefix isn't easy to press, and I leave them just for backwards compatibility. You should at least change prefix key like this `map <Leader> <Plug>(easymotion-prefix)`

Minimal but useful vimrc example:

```
let g:EasyMotion_do_mapping = 0 " Disable default mappings

" Bi-directional find motion
" Jump to anywhere you want with minimal keystrokes, with just one key binding.
" `s{char}{label}`
nmap s <Plug>(easymotion-s)
" or
" `s{char}{char}{label}`
" Need one more keystroke, but on average, it may be more comfortable.
nmap s <Plug>(easymotion-s2)

" Turn on case insensitive feature
let g:EasyMotion_smartcase = 1

" JK motions: Line motions
map <Leader>j <Plug>(easymotion-j)
map <Leader>k <Plug>(easymotion-k)
```

Now, all you need to remember is `s` and JK motions bindings, and it's good enough to boost your cursor speed!

`s` is bidirectional find motion, you can move to anywhere with it.

`<Leader>j` & **`<Leader>k`** make it easy to move to the lines.

Of course you can use any key you want instead of `s` such as `<Space>`, `<Leader>s`, etc...

If you want to use more useful mappings, please see :h easymotion.txt (https://github.com/easymotion/vim-easymotion/blob/master/doc/easymotion.txt) for more detail.