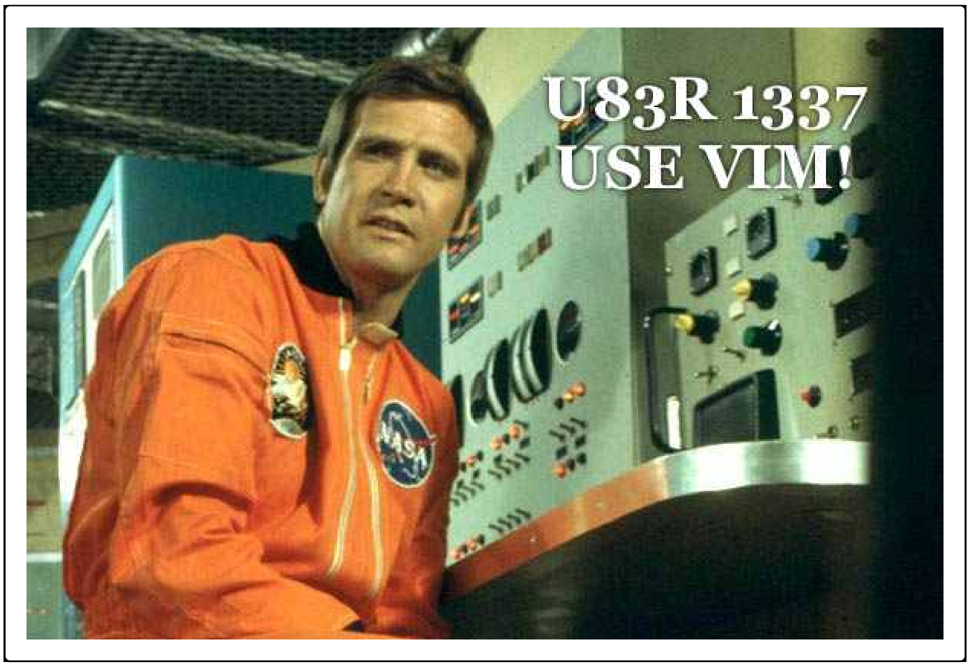
French ↓ Menu ↓ Change Theme

# Learn Vim Progressively



*TL;DR[*]: You want to teach yourself vim (the best text editor known to human kind) in the fastest way possible. This is my way of doing it. You start by learning the minimal to survive, then you integrate all the tricks slowly.*

## *Vim*[†] the Six Billion Dollar editor

> *Better, Stronger, Faster.*

Learn *vim*[†] and it will be your last text editor. There isn't any better text editor that I know of. It is hard to learn, but incredible to use.

I suggest you teach yourself Vim in 4 steps:

1. Survive
2. Feel comfortable
3. Feel Better, Stronger, Faster
4. Use superpowers of vim

By the end of this journey, you'll become a vim superstar.

But before we start, just a warning. Learning vim will be painful at first. It will take time. It will be a lot like playing a musical instrument. Don't expect to be more efficient with vim than with another editor in less than 3 days. In fact it will certainly take 2 weeks instead of 3 days.

## 1. 1st Level – Survive

0. Install vim[†]
1. Launch vim
2. DO NOTHING! Read.

In a standard editor, typing on the keyboard is enough to write something and see it on the screen. Not this time. Vim is in *Normal* mode. Let's go to *Insert* mode. Type the letter `i`.

You should feel a bit better. You can type letters like in a standard editor. To get back to *Normal* mode just press the `ESC` key.

You now know how to switch between *Insert* and *Normal* mode. And now, here are the commands that you need in order to survive in *Normal* mode:

- `i` → Insert *mode. Type* `ESC` *to return to Normal mode.*
- `x` → *Delete the char under the cursor*
- `:wq` → *Save and Quit (*`:w` *save,* `:q` *quit)*
- `dd` → *Delete (and copy) the current line*
- `p` → *Paste*

*Recommended:*

- `hjkl` *(highly recommended but not mandatory) → basic cursor move (←↓↑→). Hint:* `j` *looks like a down arrow.*
- `:help <command>` → *Show help about* `<command>`*. You can use* `:help` *without a* `<command>` *to get general help.*

Only 5 commands. That is all you need to get started. Once these command start to become natural (maybe after a day or so), you should move on to level 2.

But first, just a little remark about *Normal mode*. In standard editors, to copy you have to use the `Ctrl` key (

`Ctrl-c` generally). In fact, when you press `Ctrl`, it is as if all of your keys change meaning. Using vim in normal mode is a bit like having the editor automatically press the `Ctrl` key for you.

A last word about notations:

- instead of writing `Ctrl-λ`, I'll write `<C-λ>`.
- commands starting with `:` end with `<enter>`. For example, when I write `:q`, I mean `:q<enter>`.

## 2. 2nd Level – Feel comfortable

You know the commands required for survival. It's time to learn a few more commands. These are my suggestions:

1. Insert mode variations:

   > - `a` → *insert after the cursor*
   > - `o` → *insert a new line after the current one*
   > - `O` → *insert a new line before the current one*
   > - `cw` → *replace from the cursor to the end of the word*

2. Basic moves

   > - `0` → *go to the first column*
   > - `^` → *go to the first non-blank character of the line*
   > - `$` → *go to the end of line*
   > - `g_` → *go to the last non-blank character of line*

- `/pattern` → *search for* `pattern`

3. Copy/Paste

- `P` → *paste before, remember* `p` *is paste after current position.*
- `yy` → *copy the current line, easier but equivalent to* `ddP`

4. Undo/Redo

- `u` → *undo*
- `<C-r>` → *redo*

5. Load/Save/Quit/Change File (Buffer)

- `:e <path/to/file>` → *open*
- `:w` → *save*
- `:saveas <path/to/file>` → *save to* `<path/to/file>`
- `:x`, `ZZ` *or* `:wq` → *save and quit (*`:x` *only save if necessary)*
- `:q!` → *quit without saving, also:* `:qa!` *to quit even if there are modified hidden buffers.*
- `:bn` *(resp.* `:bp`*)* → *show next (resp. previous) file (buffer)*

Take the time to learn all of these command. Once done, you should be able to do every thing you are able to do in other editors. You may still feel a bit awkward. But follow me to the next level and you'll see why vim is worth the extra work.

## 3. 3rd Level – Better. Stronger. Faster.

Congratulation for reaching this far! Now we can start with the interesting stuff. At level 3, we'll only talk about commands which are compatible with the old vi editor.

### 3.1. Better

Let's look at how vim could help you to repeat yourself:

1. `.` → (dot) will repeat the last command,
2. N<command> → will repeat the command N times.

Some examples, open a file and type:

- `2dd` → *will delete 2 lines*
- `3p` → *will paste the text 3 times*
- `100idesu [ESC]` → *will write "desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu*

*desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu desu”*

- `.` → *Just after the last command will write again the 100 “desu”.*
- `3.` → *Will write 3 “desu” (and not 300, how clever).*

## 3.2. Stronger

Knowing how to move efficiently with vim is *very* important. Don’t skip this section.

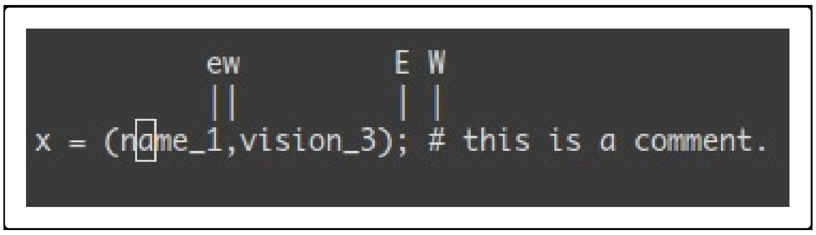
1. N`G` → Go to line N
2. `gg` → shortcut for `1G` - go to the start of the file
3. `G` → Go to last line

4. Word moves:

    1. `w` → *go to the start of the following word,*
    2. `e` → *go to the end of this word.*

    *By default, words are composed of letters and the underscore character. Let’s call a WORD a group of letter separated by blank characters. If you want to consider WORDS, then just use uppercase characters:*

    1. `W` → *go to the start of the following WORD,*
    2. `E` → *go to the end of this WORD.*



```
          ew         E W
          ||         | |
x = (name_1,vision_3); # this is a comment.
```

Now let's talk about very efficient moves:

> - `%` : *Go to the corresponding* `(`, `{`, `[`.
> - `*` *(resp.* `#`*) : go to next (resp. previous) occurrence of the word under the cursor*

Believe me, the last three commands are gold.

### 3.3. Faster

Remember about the importance of vi moves? Here is the reason. Most commands can be used using the following general format:

`<start position><command><end position>`

For example : `0y$` means

- `0` → go to the beginning of this line
- `y` → yank from here
- `$` → up to the end of this line

We also can do things like `ye`, yank from here to the end of the word. But also `y2/foo` yank up to the second occurrence of “foo”.

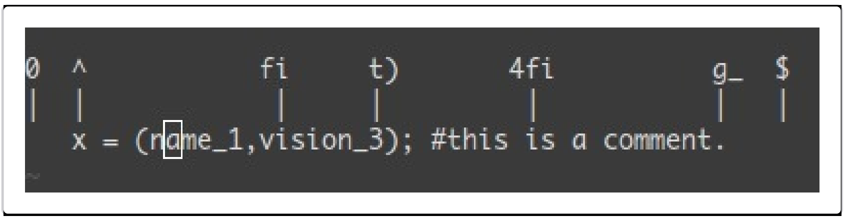
But what was true for `y` (yank), is also true for `d` (delete), `v` (visual select), `gU` (uppercase), `gu` (lowercase), etc…

## 4. 4th Level – Vim Superpowers

With all preceding commands you should be comfortable using vim. But now, here are the killer features. Some of these features were the reason I started to use vim.

### 4.1. Move on current line: `0` `^` `$` `g_` `f` `F` `t` `T` `,` `;`

- `0` → *go to column 0*
- `^` → *go to first character on the line*
- `$` → *go to the last column*
- `g_` → *go to the last character on the line*
- `fa` → *go to next occurrence of the letter* `a` *on the line.* `,` *(resp.* `;`*) will find the next (resp. previous) occurrence.*
- `t,` → *go to just before the character* `,`*.*
- `3fa` → *find the 3rd occurrence of* `a` *on this line.*
- `F` *and* `T` → *like* `f` *and* `t` *but backward.*



```
0  ^           fi    t)        4fi         g_  $
|  |            |     |         |           |   |
   x = (name_1,vision_3); #this is a comment.
~
```

A useful tip is: `dt"` → remove everything until the `"`.

### 4.2. Zone selection `<action>a<object>` or `<action>i<object>`

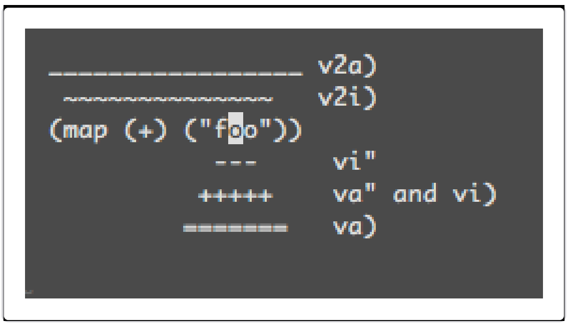
These command can only be used after an operator in visual mode. But they are very powerful. Their main pattern is:

`<action>a<object>` and `<action>i<object>`

Where action can be any action, for example, `d` (delete), `y` (yank), `v` (select in visual mode). The object can be: `w` a word, `W` a WORD (extended word), `s` a sentence, `p` a paragraph. But also, natural character such as `"`, `'`, `)`, `}`, `]`.

Suppose the cursor is on the first `o` of `(map (+) ("foo"))`.

- `vi"` → *will select* `foo`.
- `va"` → *will select* `"foo"`.
- `vi)` → *will select* `"foo"`.
- `va)` → *will select* `("foo")`.
- `v2i)` → *will select* `map (+) ("foo")`
- `v2a)` → *will select* `(map (+) ("foo"))`

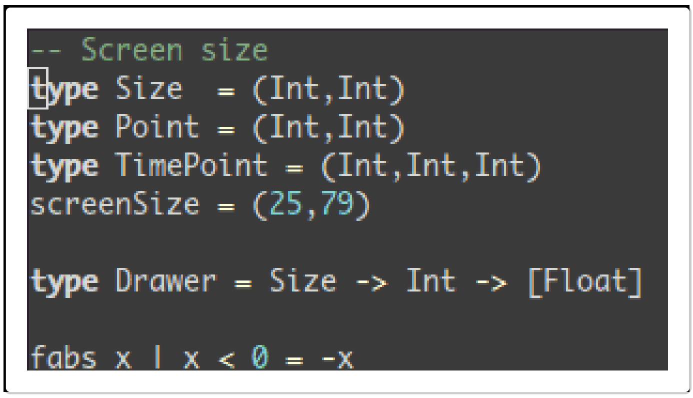


```
________________ v2a)
 ~~~~~~~~~~~~~~  v2i)
(map (+) ("foo"))
           ---    vi"
          +++++   va" and vi)
         =======  va)
```

### 4.3. Select rectangular blocks: <C-v>.

Rectangular blocks are very useful for commenting many lines of code. Typically: `0<C-v><C-d>I-- [ESC]`

- `^` → go to the first non-blank character of the line
- `<C-v>` → Start block selection
- `<C-d>` → move down (could also be `jjj` or `%`, etc...)
- `I-- [ESC]` → write `--` to comment each line

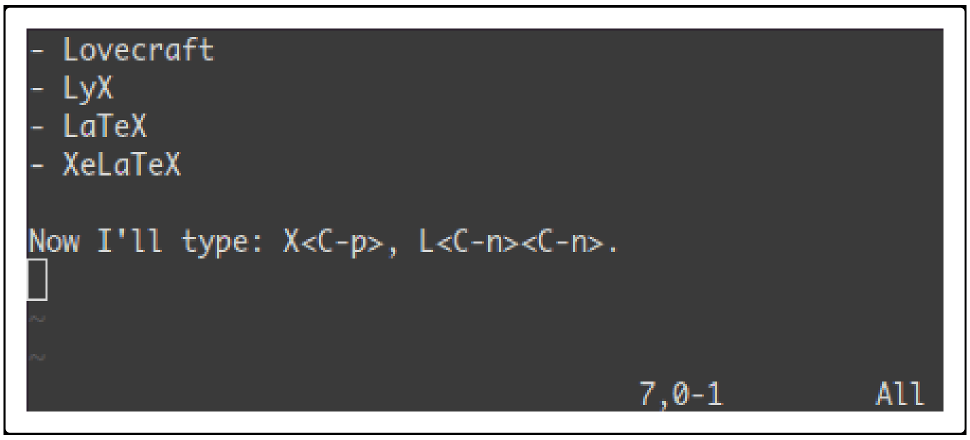


```
-- Screen size
type Size  = (Int,Int)
type Point = (Int,Int)
type TimePoint = (Int,Int,Int)
screenSize = (25,79)

type Drawer = Size -> Int -> [Float]

fabs x | x < 0 = -x
```

Note: in Windows you might have to use `<C-q>` instead of `<C-v>` if your clipboard is not empty.

### 4.4. Completion: <C-n> and <C-p>.

In Insert mode, just type the start of a word, then type `<C-p>`, magic...



```
- Lovecraft
- LyX
- LaTeX
- XeLaTeX

Now I'll type: X<C-p>, L<C-n><C-n>.

~
~
                                       7,0-1          All
```

## 4.5. Macros : `qa` do something `q`, `@a`, `@@`

`qa` record your actions in the *register* `a`. Then `@a` will replay the macro saved into the register `a` as if you typed it. `@@` is a shortcut to replay the last executed macro.

Example

*On a line containing only the number 1, type this:*

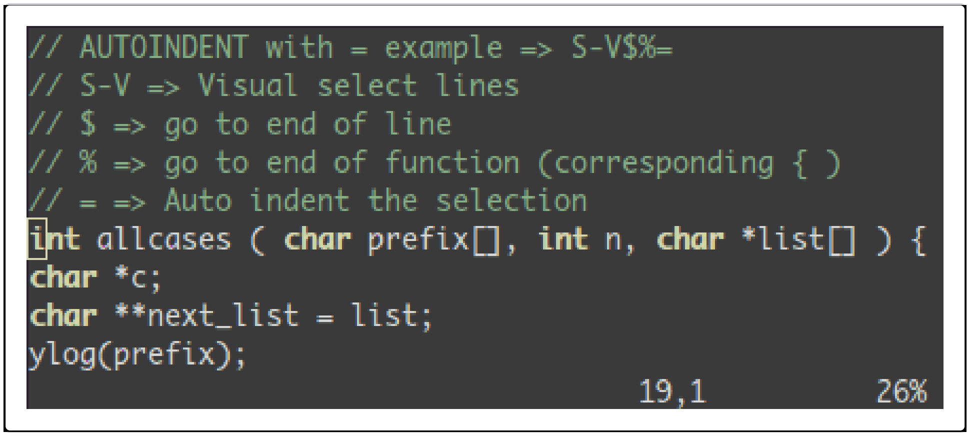
- `qaYp<C-a>q` → 
  - `qa` *start recording.*
  - `Yp` *duplicate this line.*
  - `<C-a>` *increment the number.*
  - `q` *stop recording.*
- `@a` → *write 2 under the 1*
- `@@` → *write 3 under the 2*
- *Now do* `100@@` *will create a list of increasing numbers until 103.*

```
1
~
~
~
~
~
~
~
~
                                                  1,1
```

## 4.6. Visual selection: v,V,<C-v>

We saw an example with `<C-v>`. There is also `v` and `V`. Once the selection has been made, you can:

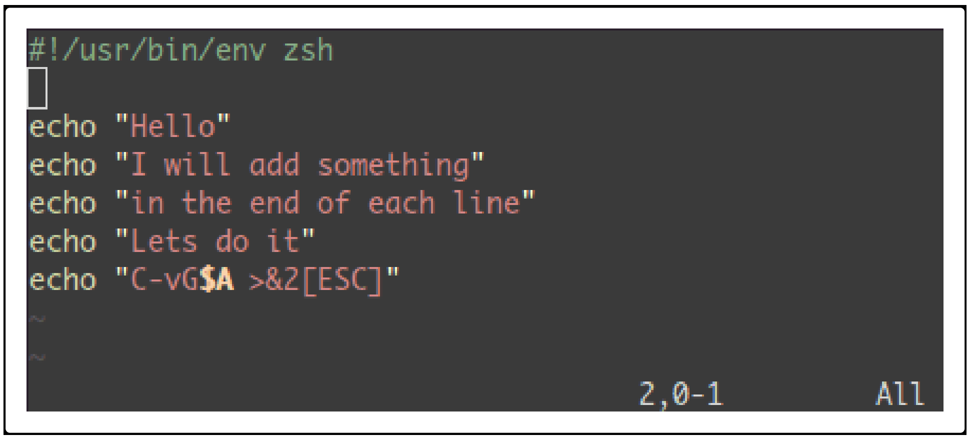
- `J` → join all the lines together.
- `<` (resp. `>`) → indent to the left (resp. to the right).
- `=` → auto indent



```
// AUTOINDENT with = example => S-V$%=
// S-V => Visual select lines
// $ => go to end of line
// % => go to end of function (corresponding { )
// = => Auto indent the selection
int allcases ( char prefix[], int n, char *list[] ) {
char *c;
char **next_list = list;
ylog(prefix);
                                    19,1          26%
```

Add something at the end of all visually selected lines:

- `<C-v>`
- go to desired line (`jjj` or `<C-d>` or `/pattern` or `%` etc...)
- `$` go to the end of the line
- `A`, write text, `ESC`.

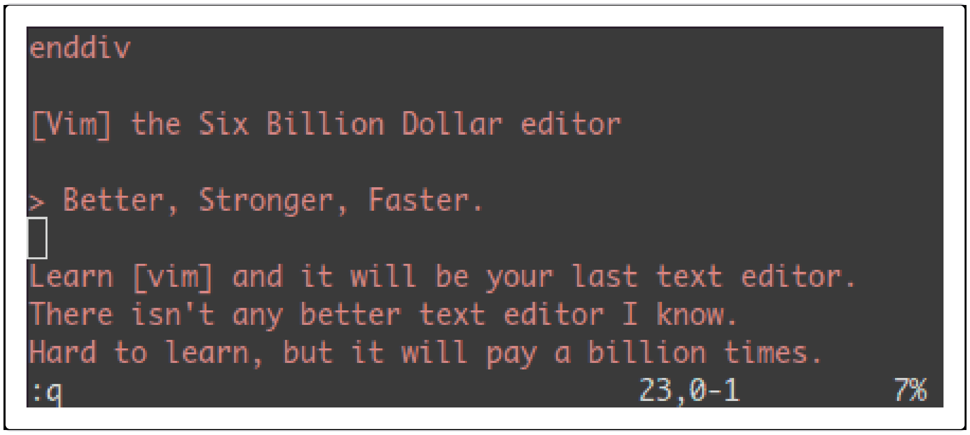


```
#!/usr/bin/env zsh

echo "Hello"
echo "I will add something"
echo "in the end of each line"
echo "Lets do it"
echo "C-vG$A >&2[ESC]"
~
~
                                              2,0-1          All
```

## 4.7. Splits: `:split` and `vsplit`.

These are the most important commands, but you should look at `:help split`.

- *`:split` → create a split (`:vsplit` create a vertical split)*
- *`<C-w><dir>` : where dir is any of `hjkl` or ←↓↑→ to change the split.*
- *`<C-w>_` (resp. `<C-w>|`) : maximise the size of the split (resp. vertical split)*
- *`<C-w>+` (resp. `<C-w>-`) : Grow (resp. shrink) split*



```
enddiv

[Vim] the Six Billion Dollar editor

> Better, Stronger, Faster.

Learn [vim] and it will be your last text editor.
There isn't any better text editor I know.
Hard to learn, but it will pay a billion times.
:q                                            23,0-1          7%
```

# 5. Conclusion

That was 90% of the commands I use every day. I suggest that you learn no more than one or two new commands per day. After two to three weeks you'll start to feel the power of vim in your hands.

Learning Vim is more a matter of training than plain memorization. Fortunately vim comes with some very good tools and excellent documentation. Run vimtutor until you are familiar with most basic commands. Also, you should read this page carefully: `:help usr_02.txt`.

Then, you will learn about `!`, folds, registers, plugins and many other features. Learn vim like you'd learn piano and all should be fine.

These social sharing links preserve your privacy

## 6. Comments

102 comments ★ 6

Join the discussion…

Best Community Share Login

Doron Linder • 2 years ago

I created an online game that teaches VIM commands. You play the cursor on a "Zelda meets text editing" adventure. The first couple of levels are free (hjkl and word navigation). The rest requires paying. Check it out - http://vim-adventures.com

21 • Reply • Share ›

yogsototh → Doron Linder • 2 years ago

I believe this is a great idea! This is genius!

5 • Reply • Share ›

johan → Doron Linder • 10 months ago

Dude this is awesome. Thanks!

1 • Reply • Share ›

Mark → Doron Linder • 11 months ago

It's a good game!

1 • Reply • Share ›

yen3 → Doron Linder • 2 months ago

It looks very interesting !

• Reply • Share ›

Neo → Doron Linder • 2 months ago

Good Job!

• Reply • Share ›

behemoth → Doron Linder • 7 months ago

very nice, thanks

• Reply • Share ›

Jean-Pierre de SOZA • 8 months ago

Have you heard of the "deep" command... Not super-useful though, it helps swap two words, when you're positioned before the first of them:
de -> delete to end of word
e -> move to the end of second word
p -> paste back the first word
Easy :)

5 • Reply • Share ›

Harbort • 2 years ago

A small error though: "t," will go just *before* &#039,&#039 and not just after.

Also very useful: range commands! You hints at one of them (i") but

you can use i (for inner) and a (for ... a) with:
s - sentence
w - word
p - paragraph
( - round brackets
[ - square brackets
{ - curly brackets

So if you type : "yi(" that would copy everything within the inner-most parenthesis.

You can also add a number to not get the inner most, but other levels:

"y2a(" ...

4 • Reply • Share ›

yogsototh ➜ Harbort • 2 years ago
Thanks for the comment! I fixed the error.

I&#039ll definitively add a text_objects subsection under in the super power section. And your example of "y2a(" is great.

• Reply • Share ›

Asif Sulikeri • 10 months ago
Learning VIM was never so wonderful..!! Thanks to you..!!
R.E.S.P.E.C.T..!!

2 • Reply • Share ›

ciferkey • 2 years ago
Thank you very much! This is the exact kind of guide I was looking for!

2 • Reply • Share ›

yogsototh ➜ ciferkey • 2 years ago
You&#039re welcome!

• Reply • Share ›

A.B • 4 months ago
I would highly encourage to disable arrow keys, before starting vim ;)

1 • Reply • Share ›

someone • 2 years ago
thank you!

1 • Reply • Share ›

Krishs • 2 years ago
Thanks a lot man... this is what i looking for.... also looking forward for the kind of articles. great job!!!
1 • Reply • Share ›

@wsaryoo • 2 years ago
thanks for VIM quick learning !! dividing the high learning curve to a few steps :)
1 • Reply • Share ›

Pramode • 2 years ago
Great article! Thanks for writing it!
1 • Reply • Share ›

Jian H. L. • 2 years ago
Good article!
I learnt <C v> and = from your article.
I tried <v> in the <C v> example, seems they&#039re not the same:)
1 • Reply • Share ›

@killa__bee • 2 years ago
Here&#039s an important tip for learning vim (that duplicates most of your admittedly nice efforts above): type "vimtutor" at the terminal.
1 • Reply • Share ›

yogsototh → @killa__bee • 2 years ago
Thanks for the reply, but I mention vimtutor in the conclusion as a way to train you (not exactly learn vim).
1 • Reply • Share ›

claytron • 2 years ago
I will be using this to convert new vim users, excellent way of easing in to things!
1 • Reply • Share ›

yogsototh → claytron • 2 years ago
Thanks for your kind comment! I greatly appreciate it.
1 • Reply • Share ›

Dennis • 2 years ago
Great intro!

Here&#039s an obscure trick I use pretty often...
:norm

Typing that plus some commands applies the commands to each selected line. So for example, select a bunch of lines that you want to add a semicolon to the end of, then type :norm A;

Invoking a macro inside your :norm really makes things fun.

1 ^ | ˅ • Reply • Share ›

yogsototh ➔ Dennis • 2 years ago

Thanks, I didn&#039t knew that one. I&#039ll try to remember it.

^ | ˅ • Reply • Share ›

Kwpolska • 2 years ago

Not "edition mode" and "insertion mode," but "Normal mode" and "Insert mode."

1 ^ | ˅ • Reply • Share ›

yogsototh ➔ Kwpolska • 2 years ago

Thanks another time, I fixed that too.

^ | ˅ • Reply • Share ›

Seren • 2 years ago

You should add text objects to Vim superpowers. Check :help text-objects in Vim. It is one of the Vim superpower that is almost never mentioned.

(ciw to change a word under cursor, dip to delete current paragraph, etc )

1 ^ | ˅ • Reply • Share ›

yogsototh ➔ Seren • 2 years ago

Thanks for the suggestion. I only mentionned vi" , I should add some words about them.

1 ^ | ˅ • Reply • Share ›

xie • 16 days ago

Dude, thanks!

^ | ˅ • Reply • Share ›

krishraov • 3 months ago

Brilliant !!! This is such a detailed explanation of the basics. I have been wondering if I should learn Vi or Emacs but the bindings in emacs threw me off ... thanks for this tutorial!!

^ | ˅ • Reply • Share ›

tharun • 5 months ago
how to "saveas" :e saveas <path file="" to=""> ????????could you elaborate plz.....</path>
• Reply • Share ›

yogsototh Mod tharun • 5 months ago
To save as you just need to write:

:w filename

or

:saveas filename

:e is for open, and :w is for save using the current file name.
• Reply • Share ›

tharun yogsototh • 5 months ago
thanx....how abt saving in a different directory?
• Reply • Share ›

Phil Thompson • 5 months ago
Thanks, this post has been a great help in learning Vim.

I would move `u` and add `CTRL+Q` (turn on flow-control - if Vim appears to stop responding) to the Survive section. I've found that being able to get back to a sane state in Vim is important.
• Reply • Share ›

behemoth • 7 months ago
Useful
• Reply • Share ›

Tomas Pruzina • 7 months ago
Best ... vim .. profficiency guide... ever!
<3
• Reply • Share ›

ersran9 • 9 months ago
This is awesome. Thanks a lot!
• Reply • Share ›

Ben F • 11 months ago
I thought you should know your "Zone" actions section is a little incorrect. These "zones" or "text objects" work directly from normal

mode, you don't actually NEED to use visual mode for them to work.

yiw yanks the word just as well as yviw.

• Reply • Share ›

Keith Bowes • 11 months ago

Vim is the best text editor around, no doubt. I've tried others, like Emacs and nano, and they both have a bunch of long key sequences to do common tasks. Vim, though, is so easy! I can't imagine anyone thinking it's hard. It does of course come with vimtutor, which will get into step 1 in no time. Though, vim does come with gvim and evim. I can't imagine using evim, though. You might as well use Notepad or something. gvim is about making a transition from GUI text editors to efficiency.

• Reply • Share ›

Mark • 11 months ago

4.1 picture for illustration is not correct. 'fi' should be point directly to 'i', other than its left word.

• Reply • Share ›

yogsototh Mod → Mark • 11 months ago

Thanks I fixed it.

• Reply • Share ›

Phil Pirozhkov • 11 months ago

"4.3 Select rectangular blocks: " ^ → go to start of the line" is wrong.

• Reply • Share ›

yogsototh Mod → Phil Pirozhkov • 11 months ago

Yes thanks, I now use "first non-blank character of the line"

• Reply • Share ›

Fang • 2 years ago

I am an Objective C developer and am currently learning VIM with hopes of eventually being able to do all text editing and programming with VIM.

I also read your article describing briefly all the languages you have experience with and found that really interesting, and helpful for when

I choose my next programming language to learn.

Your website is really great and I will continue reading it. It is one of the best I have found on programming in general.

One question I have is about plugins. For iOS development, what all plugins if any do you use on your vim? (auto completion, compiler plugins, color & aesthetics). Thanks so much for your posts. They are really insightful.

• Reply • Share ›

yogsototh → Fang • 2 years ago

Thank you!

All my configuration is here. Sadly, it is much a spaghetti code for now, I don&#039t really use any system to organize it for now.
http://github.com/yogsototh/Vim-configuration

Mainly, I use solarize with dark background as coloscheme.
I use NERDTree.
I also used a lot of other plugin, but I don&#039t use them much.

Working in Objective-C I don&#039t use a 100% vim for now. I open Xcode and if I need to make big changes I use Vim, for very small updates, I generally use Xcode. I didn&#039t find an easy way to access documentation and run my program easily from vim. But if I find a way I&#039ll certainly blog about it.

• Reply • Share ›

Fang → yogsototh • 2 years ago

Ah, thank you. Very helpful.

• Reply • Share ›

Ryan • 2 years ago

Hi there! I&#039ve just written an interactive Vim tutorial that would be a good sequel to this article.
http://rypress.com/articles/utilities/vim-tutoria...

I&#039d love to hear your thoughts.

• Reply • Share ›

yogsototh → Ryan • 2 years ago

Hi Ryan! Sorry for the late reply.

I go throught you entire tutorial, but I was a bit fast. This is a great work.

My only complaint is the number of section which require the user to remember a list of command before comming back to the tutorial. You might want to start by the split. This way, it will be easier to have the instruction of your tutorial always visible.

Very good work though!

• Reply • Share ›

↑ Menu ↑

Published on 2011-08-25
Follow @yogsototh
Yann Esposito©
Done with Vim & ~~nanoc~~ Hakyll