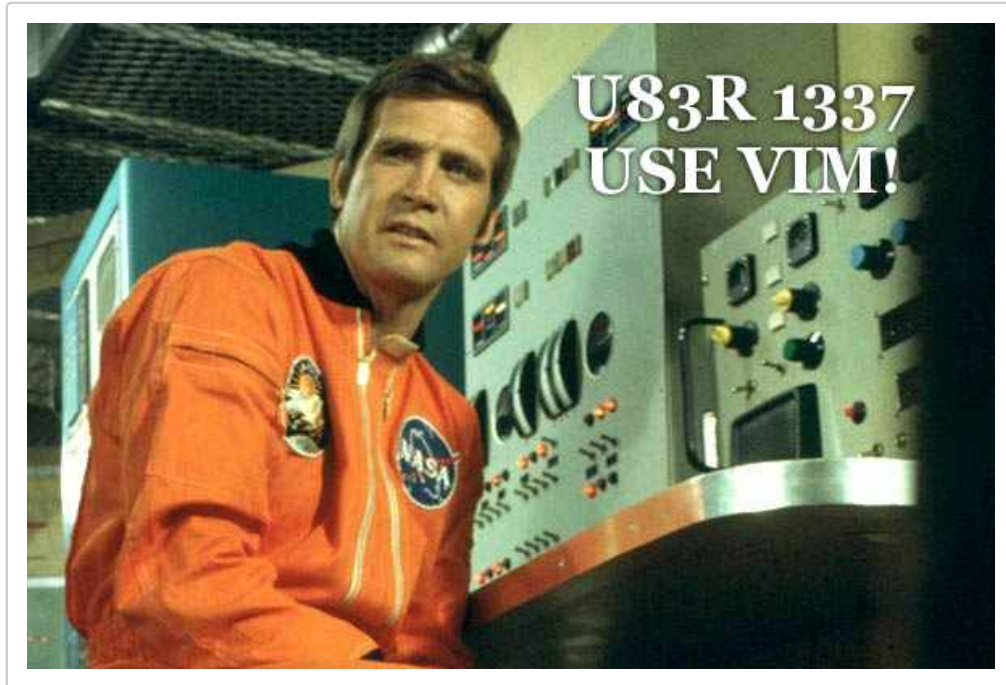


# Learn Vim Progressively



*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<sup>†</sup> the Six Billion Dollar editor*

*Better, Stronger, Faster.*

Learn *vim*<sup>†</sup> 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<sup>†</sup>
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:*

- `h j k l` (*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

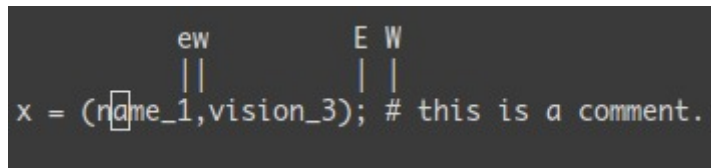
Let's look at how vim could help you to repeat yourself:



1. `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**.



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 : `Oy$` means

- `O` → 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: `O ^ $ g_ f F t T , ;`

- `O` → 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>`

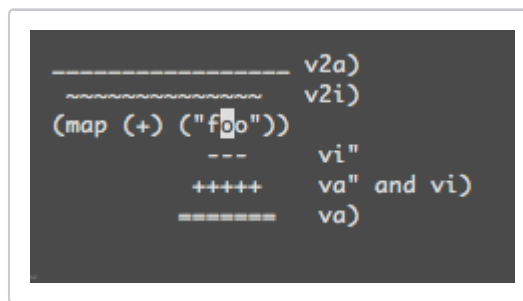
These command can only be used after an operator in visual mode. But they are very powerful. Their main pattern is:

`<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"))`.

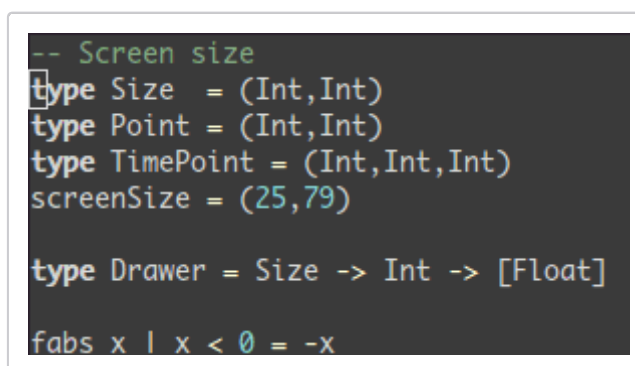
- `vi"` → will select `foo`.
- `va"` → will select `"foo"`.
- `vi)` → will select `"foo"`.
- `va)` → will select `("foo")`.
- `v2i)` → will select `map (+) ("foo")`
- `v2a)` → will select `(map (+) ("foo"))`



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

Rectangular blocks are very useful for commenting many lines of code. Typically: `O<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



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:*

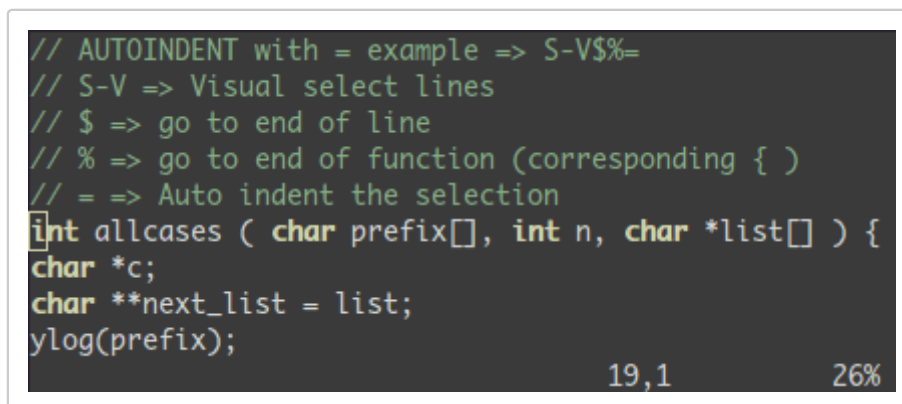
- **qaYp<C-a>q** →
  - **qa** start recording.
  - **Yp** duplicate this line.
  - **<C-a>** increment the number.
  - **q** stop recording.
- **@a** → write 2 under the 1
- **@@** → write 3 under the 2
- Now do **100@@** will create a list of increasing numbers until 103.



## 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



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.

*If you liked this article, there is a follow up: Vim as IDE<sup>†</sup>*



These social sharing links preserve your privacy

[Home](#) / [Blog](#) / [Softwares](#) / [About](#)

↑ Top ↑

## 6. Comments

132 Comments yannesposito

 Login ▾

♥ Recommend 8

 Share

Sort by Best ▾



Join the discussion...

**Doron Linder** • 4 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>

43 ^ | v • Reply • Share ›

**yogsooth** ➔ Doron Linder • 4 years ago

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

8 ^ | v • Reply • Share ›

**johan** ➔ Doron Linder • 4 years ago

Dude this is awesome. Thanks!

3 ^ | v • Reply • Share ›

**Mark** ➔ Doron Linder • 4 years ago

It's a good game!

2 ^ | v • Reply • Share ›

**Goddard** ➔ Doron Linder • a year ago

Nice game, but honestly a little pricey and an offline mode would be nice.

1 ^ | v • Reply • Share ›

**Daniel Angel Muñoz Trejo** ➔ Doron Linder • 6 months ago

Loved the game!

^ | v • Reply • Share ›

**yen3** ➔ Doron Linder • 3 years ago

It looks very interesting !

^ | v • Reply • Share ›

**Neo** ➔ Doron Linder • 3 years ago

Good Job!

^ | v • Reply • Share ›

**behemoth** ➔ Doron Linder • 3 years ago

very nice, thanks

^ | v • Reply • Share ›

**Jean-Pierre de SOZA** • 3 years 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

D -> paste back the first word

Easy :)

12 ^ | v • Reply • Share ›

**S.Nkm** ➔ Jean-Pierre de SOZA • a year ago

similarly xp swaps 2 chars.

^ | v • Reply • Share ›

**Asif Sulikeri** • 4 years ago

Learning VIM was never so wonderful...!! Thanks to you...!! R.E.S.P.E.C.T...!!

6 ^ | v • Reply • Share ›



**Harbort** • 5 years ago

A small error though: "t," will go just \*before\* &#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 ^ | v • Reply • Share ›



**yogsototh** ➔ Harbort • 5 years ago

Thanks for the comment! I fixed the error.

I&#039;ll definitely add a text\_objects subsection under in the super power section. And your example of "y2a(" is great.

^ | v • Reply • Share ›

**Alex Berez** • 3 years ago

I would highly encourage to disable arrow keys, before starting vim ;)

3 ^ | v • Reply • Share ›

**Shanthakumar** ➔ Alex Berez • 2 years ago

or do something useful with them :)

```
nnoremap <left> :vertical resize +5<cr>
```

```
nnoremap <right> :vertical resize -5<cr>
```



```
nnoremap <right> :verticalresize +5<cr>
```

```
nnoremap <up> :resize +5<cr>
```

```
nnoremap <down> :resize -5<cr>
```

3 ^ | v • Reply • Share ›

**Alex Berez** ➔ Shanthakumar • 2 years ago

I like your idea a lot! Thanks for great advice!  
(adding to my .vimrc.after)

^ | v • Reply • Share ›

**Jared** ➔ Alex Berez • 2 years ago

I like to use <left> and <right> for my buffers.

" Better use of arrows

```
nnoremap <silent> <right> :bnext<cr>
```

```
nnoremap <silent> <left> :bprev<cr>
```

^ | v • Reply • Share ›



**Jian H. L.** • 5 years ago

Good article!

I learnt <C v> and = from your article.

I tried <v> in the <C v> example, seems they're not the same:)

2 ^ | v • Reply • Share ›



**@killa\_\_bee** • 5 years ago

Here's an important tip for learning vim (that duplicates most of your admittedly nice efforts above): type "vimtutor" at the terminal.

2 ^ | v • Reply • Share ›



**yogsototh** ➔ @killa\_\_bee • 5 years ago

Thanks for the reply, but I mention vimtutor in the conclusion as a way to train you (not exactly learn vim).

1 ^ | v • Reply • Share ›



**ciferkey** • 5 years ago

Thank you very much! This is the exact kind of guide I was looking for!

2 ^ | v • Reply • Share ›



**yogsototh** ➔ ciferkey • 5 years ago

You're welcome!

^ | v • Reply • Share ›



**johannes** • 2 years ago

This is just great. Thank you.

1 ^ | v • Reply • Share ›



**johannes** → johannes • 2 years ago

I'm coming back here several times a week. which is why I noticed that the layout has been broken from the middle of chapter 3 to the end - everything is horicontally centered. Just FYI.

1 ^ | v • Reply • Share ›

**yogsototh** Mod → johannes • 2 years ago

Thanks! I fixed it now.

^ | v • Reply • Share ›



**johannes** → yogsototh • 2 years ago

thanks you!

1 ^ | v • Reply • Share ›



**Mark** • 4 years ago

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



1 ^ | v • Reply • Share ›

**Alex Berez** → Mark • 2 years ago

'ti' will make the trick ;)

^ | v • Reply • Share ›

**yogsototh** Mod → Mark • 4 years ago

Thanks I fixed it.

^ | v • Reply • Share ›



**someone** • 5 years ago

thank you!

1 ^ | v • Reply • Share ›



**Krishs** • 5 years ago

Thanks a lot man... this is what i looking for.... also looking forward for the kind of articles. great job!!!

1 ^ | v • Reply • Share ›



**@wsaryoo** • 5 years ago

thanks for VIM quick learning !! dividing the high learning curve to a few steps :)

1 ^ | v • Reply • Share ›

**Pramode** • 5 years ago

Great article! Thanks for writing it!

1 ^ | v • Reply • Share ›

**claytron** • 5 years ago

I will be using this to convert new vim users, excellent way of easing in to things!

1 ^ | v • Reply • Share ›

**yogsototh** ➔ claytron • 5 years ago

Thanks for your kind comment! I greatly appreciate it.

1 ^ | v • Reply • Share ›

**Dennis** • 5 years ago

Great intro!

Here's 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 ^ | v • Reply • Share ›

**yogsototh** ➔ Dennis • 5 years ago

Thanks, I didn't knew that one. I'll try to remember it.

^ | v • Reply • Share ›

**Vlad** ➔ yogsototh • a year ago

Thank you both so much! So,

`"35,45norm I--"` comments lines from 35 to 45 and`"35,45norm ^2x"` uncomments them!

So easy! Thank you!!!!

^ | v • Reply • Share ›

**Kwpolska** • 5 years ago

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

1 ^ | v • Reply • Share ›

**yogsototh** ➔ Kwpolska • 5 years ago

Thanks another time, I fixed that too.

^ | v • Reply • Share ›

**Seren** • 5 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 ^ | v • Reply • Share ›



**yogsototh** ➔ Seren • 5 years ago

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

1 ^ | v • Reply • Share ›

**Suresh** • 5 months ago

Nice composition, thank you

^ | v • Reply • Share ›

**Saitama** • 7 months ago

Just want to say, "I like your blog.";

^ | v • Reply • Share ›

**Ketan Shukla** • 10 months ago

Thank you so much for this article. Very easy to learn when it's step by step. Article + all the comments below are a wealth of resources! Thanks you again!

^ | v • Reply • Share ›

**Arron** • a year ago

great tutorial for vim beginners!

^ | v • Reply • Share ›

**YINGQUN KUANG** • a year ago

thanks for ur share, i wanna learn vim!!!

^ | v • Reply • Share ›



**Fdo** • 2 years ago

Nice job! I use almost the same commands you described (ctrl+a was new to me).

The only editing superpower that's easy to control so it deserves mentioning but you didnt mention was: using sed to search and replace, after a (visual) selection just by :<,>s/old/new/gc

Sure sed and regex deserve dedicated post by themselves, but "s/old/new/" is basic but powerful enough.

(I'm MS in OR engineering and vim was the most useful tool I learnt in college that wasn't part of any course, just a clever suggestion by a teacher.)

^ | v • Reply • Share ›

**Shine Wang** • 2 years ago

good article ! thx..

^ | v • Reply • Share ›

↑ Menu ↑

---

Published on 2011-08-25

Follow @yogsototh

Yann Esposito©

Done with Vim & ~~nanoc~~ Hakyll