



Build a Better Process

## FINANCE

Glossary

Quant 101

## TECH

Data Science

HTML

Linux

Solr

Vim

# Vim Cheat Sheet

**Feeling overwhelmed? This go-to Vim cheat sheet is trimmed down to the essential commands.**

1. **Get started** - Expect to struggle at first.
2. **A better way** - Ditch those one-page pdf cheat sheets that are too condensed and confuse beginners.
3. **Modes** - Simplify hundreds of command combinations by starting with four modes.
4. **Plateaus** - Try new commands and see if they save you time.



your answer without getting sidetracked.

by **Paul Alan Davis**, CFA, September 13, 2017

*Updated: December 10, 2018*

**Memorizing the thousands of Vi shortcuts and Vim command combinations can take years. Having a solid base of understanding will improve your odds of success.**

◀ Outline    ◀◀ Back

Next ▶▶

~/ [home](#) / [tech](#) / vim cheat sheet

---

## Vim Cheat Sheet Outline

### ***For Those Just Getting Started***

★☆☆ **Beginner**

The goal with these seven sections is to give the beginner confidence to replace existing tools with the Vim editor.

1. **Vim Modes** - Navigating modes
2. **Vim Command Structure** - Counts, Operators and Motions
3. **Vim Editing** - Undo, replace, delete and append
4. **Vim Cut, Copy and Paste** - Edit in Visual and Normal Modes
5. **Vim Search and Replace** - Edit in Normal and Command Modes
6. **Vim Command Mode** - Save, quit, write, read
7. **Vim Registers** - Reuse saved blocks of text
8. **Vim Tabs** - Manage multiple files in tabs

Accepting that Vim has over 1,000 commands, we only had space to cover about 80 essential commands, but these should provide a good base of knowledge to launch a further exploration of Vim commands.

### **Vim and Vi Version Confusion**



replacement for the Vi editor, despite being identified with the command **vi**.

If this is the case for you, your system is most-likely running a limited version of Vim, called **vim-tiny**. You can find your version using **:ve** inside Vim. If it says "Small version without GUI" then you have a limited, but small program. Many of the commands covered in this Vim cheat sheet will not work for you as vim-tiny emulates vi commands and vi shortcuts, so keep that in mind.

If that is the case for you, and you can upgrade to the full version of Vim, called **vim-runtime**, look for a tutorial on this later.

## Vim Modes

Keystrokes have alternative meanings in each mode. So a vital first step is knowing your current mode and how to navigate between them.

- **Normal Mode** - navigating and some editing
- **Insert Mode** - editing and appending text
- **Visual Mode** - selecting and moving text
- **Command Mode (Ex Mode)** - entering Vim commands

The default or home mode is **Normal Mode** and **Esc** returns you here.

### Navigating Vim Modes

Keystrokes used to navigate between Vim modes.

Current Mode	Desired Mode	Character(s)
Normal Mode	Insert Mode	<b>A, a, C, I, i, O, o, S, s</b>
Normal Mode	Command Mode (Ex Mode)	<b>:</b>

   		
Any Mode	Normal Mode	<b>Esc</b> (sometimes twice)

So if you make unintended keystrokes, return to **Normal Mode** using **Esc** . There are other ways to navigate between modes of course, and here we cover the basics for beginners.

## Vim Command Structure

### Counts, Operators and Motions

A combined command is one that can be applied multiple times, taking a variety of actions in all directions.

- **Counts** - The number of times to repeat an Operator or Motion, often abbreviated as **N**
- **Operators** - Actions or Vim commands to perform
- **Motions** - A navigational direction to move the cursor

Below is a short example of three combined commands translated into Counts, Operators and Motions.

Combined command	Count	Operator	Motion	What does it do?
5dw	5	d	w	Delete 5 words
3j	3		j	Move down 3 lines
3g~l	3	g~	l	Swap case for next 3 characters

### Counts



Motion afterwards. Use **ESC** to erase an unwanted count. Counts are optional, and without a Count you can think of the default being 1.

**0** is not a Count, but instead moves the cursor to the first column of a line.

## Operators





Of the 15 Operators in total, below are those commonly used by beginners. Operators are optional.

Character	Operation
<b>c</b>	Change
<b>d</b>	Delete
<b>g~</b>	Swap case
<b>gu</b>	Make lowercase
<b>gU</b>	Make uppercase
<b>y</b>	Yank (copy)
<b>&lt;</b>	Shift left
<b>&gt;</b>	Shift right

## Motions

Vim Motions can optionally be preceded by a Count and/or an Operator. Vim will move or take action from the cursor a specified number of times and remain in **Normal Mode**. Also note, there are many ways (over 90) to initiate a Motion but to save time and to avoid feeling overwhelmed, start with those in the first column.

Character	Synonym(s)	Motion	Units
<b>h</b>	Backspace, Ctrl-h	Left	Characters





   			
j	Enter, Ctrl-j, Ctrl-m, Ctrl-n	Down	Lines
k	Ctrl-p	Up	Lines
b		Backward	Words
w		Forward	Words
{		Backward	Paragraphs
}		Forward	Paragraphs
Ctrl-d		Down	1/2 Screens
Ctrl-u		Up	1/2 Screens
Ctrl-b		Backward	Screens
Ctrl-f		Forward	Screens

The **Ctrl-s** keystroke is used for Linux Terminal control and may lock your Vim console. If this happens, hit **Ctrl-q** to regain control.

### Other Motions

The following are important and commonly-used Motions.

Character	Prefix with Count?	Motion
0 (zero)	No	Move cursor to beginning of current line
\$	Yes	Move cursor to end of current line
^	No	Move cursor to first non-blank character of current line
g_	Yes	Move cursor to last non-blank character of current line

			
<b>G</b>	No	Go to end of file	
<b>H</b>	No	Go to top of screen	
<b>L</b>	No	Go to bottom of screen	
<b>M</b>	No	Go to middle of screen	
<b>zz</b>	No	Center the screen around the cursor	

A Count plus the vertical bar or pipe will go to that column of the current line. So **80|** will go to column 80.

## Vim Editing

### Two Modes for Editing: Normal Mode and Insert Mode

Here we cover editing files from two modes. Light edits are performed while remaining in the default **Normal Mode**, while others such as appending text take a Motion followed by a change to **Insert Mode**.

#### Normal Mode Vim Editing

Character	Prefix with Count?	Meaning
<b>u</b>	Yes	Undo the last edit
<b>Crtl-r</b>	Yes	Redo last undo changes
<b>r</b>	Yes	Replace one or several characters
<b>x</b>	Yes	Delete character under cursor and forward
<b>.</b>	Yes	Repeat the last change at the cursor

The **r** command is recommended for beginners over **R** which kicks you into a whole different mode called **Replace Mode**. It is often easier to use **Insert**



## Insert Mode Vim Editing

Character	Prefix with Count?	Meaning
<b>a</b>	Yes	Append text after cursor position
<b>A</b>	Yes	Append text at end of line
<b>C</b>	Yes	Cut text from cursor to end of line
<b>I</b>	Yes	Insert text at the first non-blank character of the line
<b>i</b>	Yes	Insert text before the cursor
<b>O</b>	Yes	Open line(s) above cursor
<b>o</b>	Yes	Open line(s) below cursor
<b>S</b>	Yes	Swap or delete line(s) for newly created line
<b>s</b>	Yes	Swap or delete character(s) for newly type characters

## Vim Cut, Copy and Paste





### Two Modes: Visual Mode and Normal Mode

Here are two ways to select blocks of text to accomplish standard cut, copy and paste operations. First, and easiest for those coming from other visual editors, is in **Visual Mode**. Second is remaining in **Normal Mode**.

#### Visual Mode

The following keystrokes perform the most basic copy and paste, switching from **Normal Mode** to **Visual Mode**.



   		
v		Switch to Visual Mode for character selection
y		Yank (copy) text selected with Motions
p		Put (paste) selected text block after the cursor
x	d	Delete a block of selected text
Ctrl-v, (select rows), Shift+i, (add text), Esc		This enters Visual Block mode and is helpful for adding text or comment characters in front of multiple lines.

Once you enter **Visual Mode** text is highlighted as you select blocks of text with standard Motions. After you are happy with the block, type your cut or copy command from the table above. You will then automatically return to **Normal Mode** to paste the text.

Many find **Visual Mode** easier to grasp at first but advancing to **Normal Mode** for cut, copy and paste operations can save keystrokes.

The **Visual Block Mode** example above can be a timesaver if for example you wanted to comment out 20 lines of code in Python with the hash # character. Start at the first line where you want to enter a comment and hit Ctrl+v, then navigate down the 19 lines, hit Shift+i and input the text you would like, here a # and space, then Esc.

### Normal Mode

To perform cut, copy and paste operations from within **Normal Mode** requires a good understanding of Motions.

Character	Prefix with Count?	Meaning
y\$	Yes	Yank (copy) to end of line



yw	Yes	Yank (copy) current word
Y	Yes	Yank (copy) the whole current line
P	Yes	Put (paste) before the cursor
p	Yes	Put (paste) after the cursor

## Vim Search and Replace

### Search Lines, Files and Replace Text

Vim offers many commands to search for and replace text at the line-level and file-level.

We will stick with the basics here, but at a more advanced level, with regular expressions, the Vim editor offers the ability initiate any search imaginable, even across whole filesystems.

#### Search Lines

Searching for characters within lines can be helpful for jumping to specific coding symbols and letters.

Character	Prefix with Count?	Meaning
f{character}	Yes	Search current line right for N occurrence of {character}
F{character}	Yes	Search current line left for N occurrence of {character}

#### Search Files



most users memorize one of the two pattern recognition commands because when combined with the **n** and **N** symbols noted above, they both can direct Vim to search up and down a file just as easily, so using **/** or **?** is your preference.

Character	Prefix with Count?	Meaning
<b>/</b> {pattern} <b>Return</b>	Yes	Search for next N occurrence of {pattern}
<b>?</b> {pattern} <b>Return</b>	Yes	Search for previous N occurrence of {pattern}
<b>n</b>	Yes	Search for next N occurrence of an already initiated search in the same direction
<b>N</b>	Yes	Search for next N occurrence of an already initiated search in the opposite direction
<b>:noh</b> <Enter>	No	To turn off highlighting of searched-for text.

## Search and Replace

Here we start to see entries in **Command Mode** using the **:** character. The term **Command Mode**, often confuses beginners because you can enter Vim commands from any mode. Official Vim documentation refers to commands entered with **:** as **Ex Commands**, but most people use the term **Command Mode**.

Character(s)	Meaning
<b>:%s</b> {search}/{replace} <b>/gc</b>	Search over the range (%) for all occurrences of



`:10,20s/{search}/{replace}/gc` Search over the range (lines 10 to 20) for all occurrences of {search} and replace with {replace}.

The only spaces allowed in the string above sit between the `/` characters, and the `{` and `}` characters are not typed. The `g` means globally over the whole file and `c` instructs Vim to provide a confirmation before each replacement is made.

A confirmation dialog will prompt you for one of seven choices.

1. `y` - substitute at current stop
2. `n` - skip this substitution
3. `a` - accept all future substitutions
4. `q` - quit the search and replace
5. `l` - substitute at current stop then quit
6. `^E` (Ctrl-e) - scroll up to previous stop
7. `^Y` (Ctrl-y) - scroll down to next stop

If you leave the `c` off the end of the command, Vim will perform all replacements in the whole file almost instantly, so it is best for beginners to start with confirming changes.

### A Search and Replace Example

The following command uses Regular Expressions to remove the last one or several blank spaces at the end of each line in a file `:%s/\s\+$//`.

## Vim Command Mode

### Save, Quit, Write and Read

Many essential commands sit in **Command Mode**, sometimes called **Ex Command Mode**. All of the following commands are input from **Normal**



Normal Mode. When entering Command Mode, once the colon character `:` is pressed, your additional keystrokes appear at the bottom left of the screen.

There are as many as 515 commands in **Command Mode** and many can be abbreviated. Also, unlike with other modes, commands here, because they are open-ended, require **Enter** to complete.

Character(s)	Synonym(s)	Meaning
<code>:w</code>	<code>:write</code>	Write (save) current and previously-named file.
<code>:w {file}</code>	<code>:write {file}</code>	Write (save) current file and name it {file}.
<code>:wq</code>	<code>:x</code> , <code>ZZ</code>	Write (save) current and previously-named file and quit Vim.
<code>:q</code>	<code>:quit</code>	Quit Vim from an unedited file.
<code>:q!</code>	<code>ZQ</code>	Quit Vim from an edited or unedited file without saving changes.
<code>:e {file}</code>	<code>:edit {file}</code>	Edit a file named {file}.
<code>:r {file}</code>	<code>:read {file}</code>	Read an external file named {file} at the current cursor position.
<code>:sh</code>	<code>:shell</code>	To leave Vim temporarily and go to the shell. Type <code>exit</code> to return.
<code>:{cmd}</code>		To run a shell command {cmd}. Hit <code>q</code> to return to Vim.
<code>:h</code>	<code>:help</code>	Open help in a new window. Use <code>:q</code> to quit help.



Vim Registers allow you to save blocks of code to paste anywhere you like. Functionality is much like cut, copy and paste operations elsewhere but Registers allow you to save many blocks, using letters a-z and A-Z, so you have 52 slots. The last 10 yanks and deletes are automatically saved under the 0-9 slots in the Register. These can be viewed with `:reg`.

### Creating, pasting and deleting Registers

Registers work logically with actions in **Normal Mode** and **Visual Mode**. The `"` symbol starts the Register functionality. Examples below use the empty Register name `c`.

Character(s)	Synonym(s)	Meaning
<code>"cyy</code>		To save a Register from Normal Mode under the name <code>c</code> of a whole line.
<code>"cy</code>		To save a Register from Visual Mode under the name <code>c</code> of a selected section.
<code>"cp</code>		To put (paste) text from the saved Register named <code>c</code> while in Normal Mode.
<code>:call setreg('c', [])</code>		To delete an item from the Register replace it with empty text.
<code>:reg</code>	<code>:registers</code>	View the currently saved Registers. Type <code>q</code> to quit.
<code>:h reg</code>	<code>:help registers</code>	Find help on Registers. Type <code>:q</code> to quit.

Besides assigning Registers to letters and numbers, an additional 8 types exist for more advanced uses. These are described in help.



You can manage multiple text files in tabs within one Vim window. To navigate between tabs click tab titles with the mouse or use the keystrokes described below. The "X" at the top right when clicked will close that tab.

Character(s)	Synonym(s)	Meaning
<code>:tabe</code>	<code>:tabedit</code> <code>:tabnew</code>	To open a new blank tab.
<code>:tabe {file}</code>	<code>:tabedit {file}</code> <code>:tabnew {file}</code>	To open a new tab to edit {file}.
<code>:tabc</code>	<code>:tabclose</code> <code>:tabclose!</code>	To close the current tab.
<code>{count}</code> <code>gt</code>		To specify which tab with {count} or go to the next tab. This wraps around forward.
<code>{count}</code> <code>gT</code>		To specify which tab with {count} or go to the previous tab. This wraps around backward.
<code>:tabs</code>		List the current Vim tabs.

Additional Vim tab functionality includes reordering tabs, looping over tabs and closing all tabs. See `:help tabs` for more.

## Related Content

- A similarly organized [Linux Cheat Sheet](#)
- Practice hjkl navigation with the [Vimazing Race Maze Game](#)
- Common Vim commands with videos in the [Vim Reference](#)



Subscribe to our growing YouTube Channel, a companion to this free online educational website.

- For links to all Vim Reference material, click Outline.
- To review other Technology content, click Back.
- To see our Vim resources, click Next.

◀ Outline    ◀ Back

Next ▶▶

~/ [home](#) / [tech](#) / vim cheat sheet

---

Keywords: [vim cheat sheet](#) [learn vim](#) [vim keyboard shortcuts](#) [vim commands](#) [vi cheat sheet](#)  
[vim shortcuts](#) [vi commands](#) [navigating vim modes](#) [vim modes](#) [vim counts](#) [vim operators](#)  
[vim motions](#) [free cheat sheet](#) [vim keymap](#) [vi shortcuts](#) [vim quick reference](#) [vim editing](#)  
[vim search and replace](#) [vim save](#) [vim registers](#)

Get social!



Finance

Glossary of Terms  
Quant 101

Career Talk





Data Science

HTML

Linux

Solr

Vim

## Reminders



FactorPad is an independent California-based firm

[Terms](#) | [About](#)

Copyright © 2014-2019 FactorPad LLC