

# How to be a Vim Ninja

Robert Bittle

robert.bittle@dominionenterprises.com

github.com/guywithnose

December 12, 2014

Slides and demos at:

<https://github.com/guywithnose/de-devcon-vim>

<http://bit.ly/11po3hk>

# Why is Vim hard to learn?

- Vim is extremely complex
- Vim is full of features that are not immediately discoverable
- Vim is designed for experienced developers not beginners.

# Why is Vim hard to learn?

- Vim is extremely complex
- Vim is full of features that are not immediately discoverable
- Vim is designed for experienced developers not beginners.

# Why is Vim hard to learn?

- Vim is extremely complex
- Vim is full of features that are not immediately discoverable
- Vim is designed for experienced developers not beginners.

# Why is Vim worth the steep learning curve?

- Vim is designed to maximize your productivity potential.
- Your vim skills can grow slowly but consistently over time

# Why is Vim worth the steep learning curve?

- Vim is designed to maximize your productivity potential.
- Your vim skills can grow slowly but consistently over time

## Why Vim?

## The power of modes

version 1.1  
April 1st, 06

## vi / vim graphical cheat sheet

**Esc**  
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :x	% next ident	( begin sentence	) end sentence	"soft" bol down	+ next line
., goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol	- prev line	= auto-format
Q ex mode	W next WORD	E end WORD	R replace mode	T back till	Y yank line	U undo line	I insert at bol	O open above	P paste before	{ begin parag.	}	end parag.
q record macro	w next word	e end word	r replace char	t till	y yank	u undo	i insert mode	o open below	p paste after	* misc	*	misc
A append at eol	S subst line	D delete to eol	F "back" find char	G eof/ find ch	H screen top	J join lines	K help	L screen bottom	. ex cmd	! reg. spec	bol/ goto col	
a append	s subst char	d delete	f find char	g extra <sup>6</sup>	h ←	j ↓	k ↑	l →	* repeat t/T/t/f	* goto mk. bol	\ not used!	
Z quit <sup>4</sup>	X back-space	C change to eol	V visual lines	B prev word	N prev (find)	M screen mid	< un-indent	> indent	? find (rev.)			
Z extra <sup>5</sup>	X delete char	c change	v visual mode	b prev word	n next (find)	m set mark	reverse	* repeat cmd	/ find			

**motion** moves the cursor, or defines the range for an operator  
**command** direct action command, if **red**, it enters insert mode  
**operator** requires a motion afterwards, operates between cursor & destination  
**extra** special functions, requires extra input commands with a dot need a char argument afterwards  
bol = beginning of line, eol = end of line, mk = mark, yank = copy  
words: `quux(foo) bar baz`  
WORDS: `quux(foo) bar baz`

**Main command line commands ('ex'):**  
:w (save), :q (quit), :q! (quit w/o saving)  
:e f (open file f),  
:s/s/x/y/g (replace 'x' by 'y' filewide),  
:ih (help in vim), :new (new file in vim),

**Other important commands:**  
CTRL-R: redo (vim),  
CTRL-F/-B: page up/down,  
CTRL-E/-Y: scroll line up/down,  
CTRL-V: block-visual mode (vim only)

**Visual mode:**  
Move around and type operator to act on selected region (vim only)

**Notes:**  
(1) use "x before a yank/paste/del command to use that register ('clipboard') (x=a..z.) (e.g.: "ay\$ to copy rest of line to reg 'a')  
(2) type in a number before any action to repeat it that number of times (e.g.: 2p, daw, 5i, dd)  
(3) duplicate operator to act on current line (dd = delete line, >> = indent line)  
(4) ZZ to save & quit, ZQ to quit w/o saving  
(5) zt: scroll cursor to top, zb: bottom, zz: center  
(6) gg: top of file (vim only), gI: open file under cursor (vim only)

For a graphical vi/vim tutorial & more tips, go to [www.viemu.com](http://www.viemu.com) - home of ViEmu, vi/vim emulation for Microsoft Visual Studio

# vimtutor

- vimtutor is a primer for the base functionality, but it is only enough to get you started.
- The most important thing you learn in vimtutor is where to find the manual.



# vimtutor

- vimtutor is a primer for the base functionality, but it is only enough to get you started.
- The most important thing you learn in vimtutor is where to find the manual.

# Vim documentation

- Vim has very exhaustive documentation.
- It is not intended to be read completely.
- Learn to use the search functionality (:helpgrep).

# Vim documentation

- Vim has very exhaustive documentation.
- It is not intended to be read completely.
- Learn to use the search functionality (:helpgrep).

# Vim documentation

- Vim has very exhaustive documentation.
- It is not intended to be read completely.
- Learn to use the search functionality (:helpgrep).

# Maintain a cheatsheet

- Decide on a few (no more than 10) things you want to learn.
- Write them down where you can see them while you are working in vim.
- Force yourself to use them in real world scenarios.

# Maintain a cheatsheet

- Decide on a few (no more than 10) things you want to learn.
- Write them down where you can see them while you are working in vim.
- Force yourself to use them in real world scenarios.

# Maintain a cheatsheet

- Decide on a few (no more than 10) things you want to learn.
- Write them down where you can see them while you are working in vim.
- Force yourself to use them in real world scenarios.

# Recognize mechanical operations

- How often do I do this?
- How long did it take me?
- Can it be scripted?



# Basic commands

- q[0-9a-z] - start recording
- Do something that is repeatable
- q - stop recording
- @[0-9a-z]
- :help complex-repeat

# Tips

- Try to start the macro with an operation that will fail after the last iteration.
- If the macro is applied linewise to a block of consecutive lines use visual block mode.
- Try to avoid absolute motions.
- Use search motions, word motions, mark motions, etc.

# What is a vimrc file?

- Your vimrc file is where you store your custom vim configuration.
- Vim loads this information every time that it starts up.
- As you become a more mature vim user, your vimrc will grow to include all the special sauce that makes vim work the way you like it.

# Mappings

- Mappings are an easy way to bind macros to a key or combination of keys.
- Over time as you find ways to automate common operations with macros you can save them as mappings in you vimrc file.

# Motions and Actions (Vim grammar)

## Actions

- Delete - d
- Change - c
- Yank (Copy) - y

## Prepositions

- in - i
- around - a

## Motions

- Word motions – w, W, b, B, e, ge, E, gE
- Search motions – t, T, f, F, /
- Text object motions – p, s, }, ], ), >, t, ', "

# Dot Command

- Think of this like an automatic mini macro.
- Combine simple actions using the `change(c)` action.

# My Favorite Plugins

- Editing tasks
  - Surround
  - Ultisnips
- Folding (Usually language specific)
- Interface
  - fugitive
  - CtrlP

# How to install plugins

## Pathogen

- Add pathogen.vim to ~/.vim/autoload
- Add this line to your .vimrc  
    execute pathogen#infect()
- Plugins get loaded from ~/.vim/bundle



# How to install plugins

## Git

Once you have Pathogen installed

- Copy plugins to your bundle directory.
- If the plugin is in git you can check it out to a directory in bundle.
- Take advantage of git submodules.

# Directory structure

- plugin
- ftplugin
- autoload
- doc