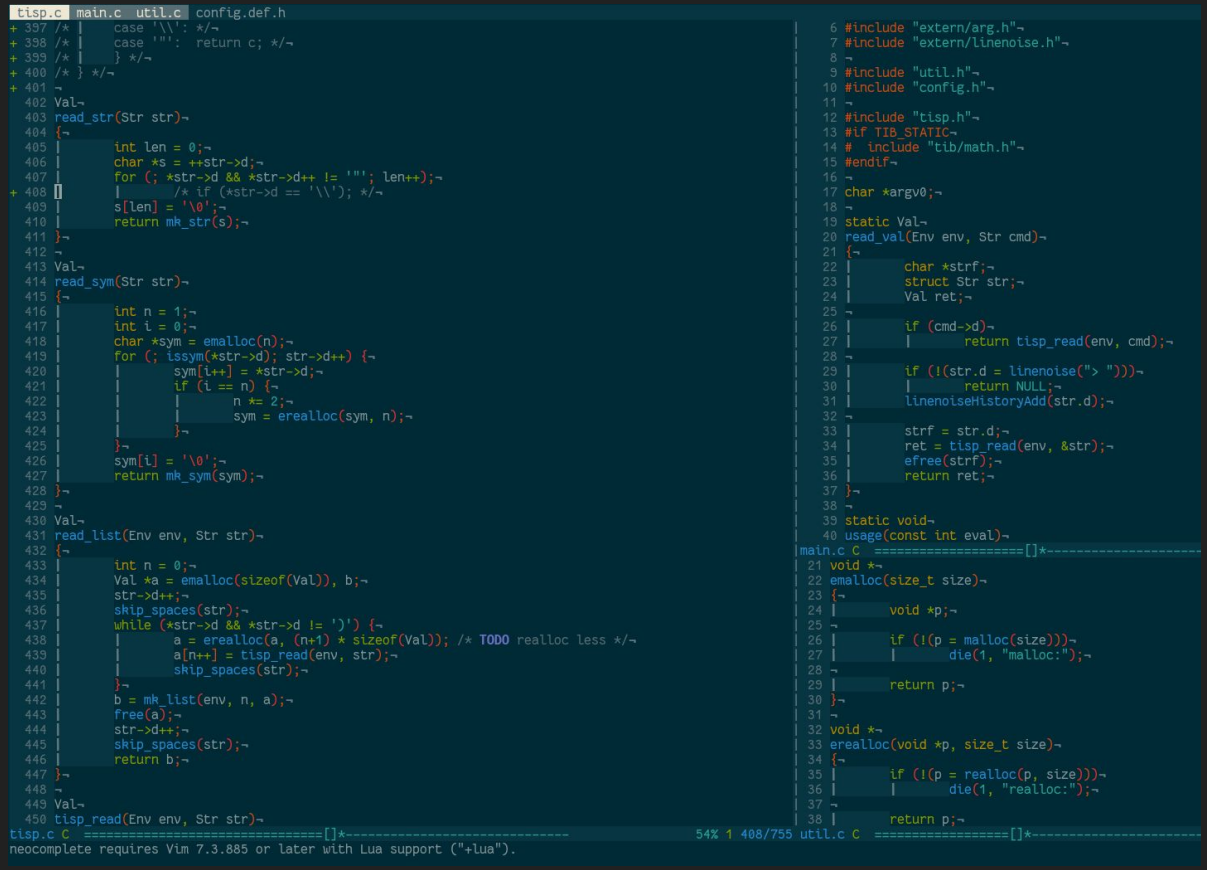
Vi IMproved

“Nobody actually creates perfect code the first time around, except me. But there's only one of me”

Linus Torvalds

What

- Universal text editor
- Mainly for source code
- Text user interface (TUI)
- Free & open source
- Modal
- Command composition
- Multiplatform
- Fast
- Small
- Powerful
- Large language support
- Customizable
- Built-in scripting-language
- Thousands of plugins
- Completely keyboard based
- Designed for touch typists



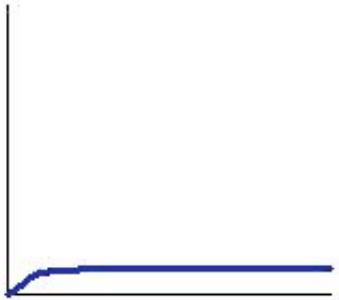

QED 1965
ed 1969
ex 1976
vi 1979
vim 1991

Why

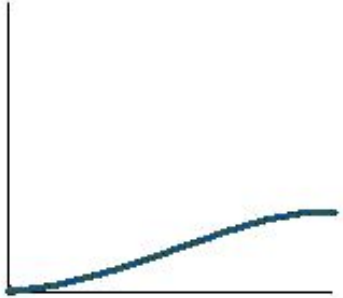
Classical learning curves for some common editors

11-17-04

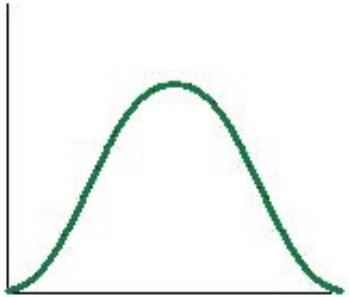
Notepad



Pico



Visual Studio

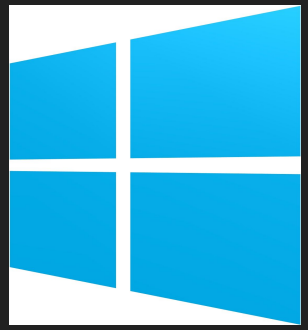


vi

emacs

Stack Overflow Developer Survey

VimScript is GitHub's 14th most popular language



~1.1 kB

Vim Cheat Sheet for Programmers

Esc Normal — Revision 2.0 Sept. 11, 2011 — Vim 7.3+ :version

http://michael.PeopleOfHonorOnly.com/vim/

HOW-TO** make Vim not suck Out of the Box:* ***:help statusline :set nocompatible ruler laststatus=2 showcmd showmode number — ***Search*** :set incsearch ignorecase smartcase hlsearch — ***Remove useless splash screen*** :set shortmess+=I

Best tips: http://vim.wikia.com/ — **Best scripts:** http://www.vim.org/scripts/index.php — :map <F9> :e $HOME/_vimrc<CR> :map <F6> :so $HOME/_vimrc<CR>

	Ctrl `	Ctrl 1	Ctrl @	Ctrl 3	Ctrl 4	Ctrl 5	Ctrl ^	Ctrl 7	Ctrl 8	Ctrl 9	Ctrl 0	Ctrl _	Ctrl =
	~ toggle case	! extern filter	@• play macro	# prev identifier	$ →|	% goto match	^ soft ←	& repeat :s	* next identifier	(begin sentence	) end sentence	_ cur line	+ ↓
	`• goto mark	1	2	3	4	5	6	7	8	9	0 hard ←	- ↑	= auto-format
Tab		14 block select	^w• window…	scroll line ↑	12 :redo	10 ctags return	scroll line ↓	half page ↑	Ctrl I	prev mark	9 ↑	Normal { ↖ paragraph	} paragraph ↘ ctags identifier
		Q ex mode	W WORD ↘	E end WORD ↘	R Replace	T• ← until char	Y copy line	U undo line	I insert ←	O open ↑	P paste ↑	[• misc.	]• misc.
		q• record macro	w word ↘	e end word ↘	r replace char	t• until char →	y copy 2	u undo	i → insert	o open ↓	p paste ↓ 6		
Caps		7 incr. #		10 half page ↓	page ↓	file/cursor info	Ctrl H	15 Ctrl J		redraw	Ctrl ;	Ctrl '	Ctrl \
		A append →	S subst line	D del →	F• ← find char	G goto eof / goto line#	H Top screen	J Join lines	K man page identifier	L Bottom screen	: cmd line	"• register 0	| ←\ goto col#
		a append ↵	s subst char	d del 2	f• find char →	g• extra 6	h ←	j ↓	k ↑	l →	; "next" f/F/t/T 16	'• goto mark ←	\•
Ctrl ^		:suspend	4,11 decr. #	Normal / Cancel	block select		page ↑	9,16 ↓	15 Ctrl M	Ctrl ,	Ctrl .	Ctrl /	
Shift ⇧		Z• quit	X ← del char	C change →|	V select lines		B ↖ WORD	N "prev" find	M Middle screen	< undent 2	> indent 2	?• find ↖	
		z• extra 3	x del char →	c change 0 2	v select chars		b ↖ word	n find "next"	m• set mark	, "prev" f/F/t/T 16	. repeat cmd	/• find ↘	

16 The search direction is relative; **next** is the initial direction, **previous** is the opposite direction. n repeat same initial direction find. N repeat opposite initial direction find. **Note:** ; , only searches cursor line, n N searches buffer.

Unused & Duplicate keys

```
\ Ctrl-K Ctrl-S (free)
Ctrl-L (redraw)
13 ` near dup of '
14 Ctrl-Q = Ctrl-V
15 Ctrl-J = Ctrl-M = ^N
```

Legend:

Macro	Register name (0-9a-zA-Z) required
Op	Motion req.; act between cursor & dst
Cmd	Command
Ins	Command and enter insert mode
Move	Moves cursor or defines range for op
Find	Search (↖ = reverse, ↘ = forward)
tag	ctags / diffs / folding
Code	Code formatting, whitespace, etc.
Extra	Extended functionality; req. extra chars
•	Char arg req. g z Z ^w ' " ` …

Modes — :help modes

n	Normal	Esc ^[^c	
i	Insert	a i r s	
v	Visual	v V ^v ^q	
o	Op pending	c d y < >	
c	Command Line	: / ? !	
word	Foo (src , ß dst , ß len);		
WORD	Foo (src , ß dst , ß len);		

Note: There is no whitespace in-between 'Foo(src,' but before/after 'dst,'

Startup

vim <filename> +123	goto line 123
vim <file> … -t Foo	edit at tag 'Foo'
vim <file> … -c "/Foo"	cmd: find 'Foo' & edit
vim -g or gvim	start GUI ver.
Linux :set guifont=ProggyTinyTT\ 12	
OSX :set guifont=ProggyTiny\:h11	
gvimdiff <file1> <file2> [<file3>]	

Broken Keys Ctrl-I = Tab, Ctrl-[= ESC

Vim is *still* **unable** to map certain keys for your own use…

Caps, Ctrl-1, Ctrl-Shift-1, Ctrl-I, Ctrl-\, etc.

§
0 *See:* src/ops.c -c "/valid_yank_reg" *for "• reg. names*
6 *See:* src/normal.c -c "/nv_cmds" *for g• extra cmds*
11 *See:* src/edit.c -c "/ctrl_x_msgs" *for ^x• insert cmds*

:help cmdline

:w save	:gui	switch to GUI	
:q quit	:q!	quit w/o save	
:e <file>	edit file in new buffer		
:source %	**exec cmds in cur file**		
:exec '...'	**do cmd**		

:help movement

soft ^ ⇤	**Start of Line**	1st non-whitespace	
hard 0 ⇤	**Start of Line**	column 0	
$ →|	**End of Line**		
|	move col 0	#|	**move col #**
^b page ↑		^f page ↓	
^u ½ page ↑		^d ½ page ↓	
^e scroll line ↑		^y scroll line ↓	
1g start of file		0g end of file	
#g **goto line #**		G **end of file**	
[[**begin this func {**			
]] **begin next func {**			
:set matchpairs=(:),{:},[:],<:>,?:\:			
% **goto matching { } < > [ ]**			

:help range

:s/Foo/Bar	find Foo replace w/ Bar
:s/Foo/Bar/g	…all instances on line
:%s/Foo/Bar	**apply to whole file**
.,.+#	**cur line, cur line + # lines**
$ **last line**	'< start of select
	'> end of select

Tags

:ts	list active tags
^]	**jump to tag under cursor**
^t	**restore cursor before tag jump**
^p	**complete word**
:ta Foo	manual jump to tag 'Foo'

Diff

[c **prev diff**	:hi DiffAdd	guifg=#rrggbb	
]c **next diff**	:hi DiffChange	guibg=#rrggbb	
:diffupdate	:hi DiffText	gui=none	
resync	:hi DiffDelete		

Folding

zR	**fold remove**
zo	fold open
zc	fold close
zi	**invert all**
zr	fold reduce
zm	fold more

Changes

:changes	
g;	older change
g,	newer change

Syntax

:syntax enable	
:set filetype=	
c cpp sh make perl python	
Note: chose only ONE type!	

:help recording

q•	start recording
@•	playback
q	stop recording
@@	repeat

Convert <eol>

:set fileformat=	
unix or *dos* or *mac*	
then :w to convert	

Code = < > << >>

:set backspace=indent,eol,start	allow backspace join lines
:set shiftwidth=#	indent width for ai
:set autoindent!	toggle auto-indent
:set lisp	lisp indent mode

:set tabstop=#	set tab stop every #th col
:set expandtab!	toggle hard/soft tabs
:set listchars=…	*tab:>-,trail:-,nbsp:%,eol:$*
:set list!	toggle whitespace
:set colorcolumn=80	visible right margin indicator
noremap + :s/^\/\//<CR>	block comment

\ :map \ :Explore<CR>	manually type <,C,R,>
§0 "•	before del/copy/paste to use register
"+x	cut to system clipboard reg. '+'
"+gP	paste from system clipboard
1	Number before any action repeats it
2p	paste twice 3. repeat thrice
2	Repeat op to act on current line
yy	copy line dd del line
<<	**undent line** >> **indent line**
3 #	highlight words under cursor
4 ZZ	save & quit ZQ quit w/o save
5 zz	**center cursor line in window**
zh	**scroll left** zl **scroll right**
zt	scroll top zb scroll bottom
§6 gg	top of file
gf	**open file under cursor**
7 ^a	incr # under cursor (Dec / Hex)
^x	decr # under cursor (Dec / Hex)
8 *	start a "new" search

Insert mode

9 ^p	prev auto-complete	^n	next
10 ^d	undent	^t	indent
§11 ^x*	^f filename completion		
^s	spelling	:set spell!	
^k	dictionary	]s	**next bad**
^t	thesaurus	:help spell	
12 ^r*	paste register 0-9a-zA-Z or …		
+	**clipboard** (or '*')	:help c_CTRL-R	
"	last del/copy	% filename	

:set numbers!	toggle line numbers
:set wrap!	toggle linewrap display
:set showmatch	highlite matching ()
noremap - :s/^\/\//<CR>	uncomment

:buffer #

:buffers	list files
:new	blank file/buffer
:bn	**next file**
:bp	prev file
:bd	**close file**
:bd!	force close
:set lines=#	
:set columns=#	
:winpos # #	GUI

Windows

:help windows	
^w* or :wincmd *	
w	**:switch to next**
c	**:close!**
n	:new
s	**:split** horz.
v	:vsplit vertical
o	:only maximize
=	**all same size**
h	move to win ←
j	move to win ↓
k	move to win ↑
l	move to win →
:sp [<filename>]	
edit in split window	

Cursor Bookmarks

:marks	` ' ^o
ma	mark local 'a'
'A	goto global 'A'
''	**prev location**

File / Directory

:Explore	or :e .
:set browsedir=…	
one of *buffer last*	

vim

[operator][count][motion]

:h operator — :h navigation

Operator	
d	delete/cut
y	yank/copy
c	change
gU	make uppercase
~	swap case
<	shift left
=	indent

Any motion can follow an operator. Marks and searches count as motions, too! `d/foo` will delete from the cursor to the next instance of "foo". `y3fi` will yank from the cursor to the 3rd "i" on the line after it. Counts can also come before operators: `5dd` will delete five lines.

w	word
W	WORD
s	sentence
[,]	[ ] block
(,)	( ) block
<, >	< > block
t	XML/HTML tag
{, }	{ } block
", '	quoted string

:h text-objects

(use text-objects) — starting cursor position on "t" of "text"; iw = "text", iW = "text-objects", i(= "use text-objects"

Up/down motions (:h up-down-motions)

gg	first line
^b	up 1 page
^u	up ½ page
k	up 1 line
j	down 1 line
^d	down ½ page
^f	down 1 page
G	last line

Left/right motions (:h left-right-motions)

0	beginning of line
^	first non-blank character
B	previous WORD
b	previous word
h	previous character
l	next character
e	end of word
w	beginning of next word
E	end of WORD
W	beginning of next WORD
$	end of line

Tabs

	ts	sw	sts	et
use spaces only	*n*	*n*	*n*	on
use tabs only	*n*	*n*	0	off

Set *n* to desired tab width (default 8)

tabstop	ts	Columns per tabstop
shiftwidth	sw	Columns per `<<`
softtabstop	sts	Spaces per tab
expandtab	et	`<Tab>` inserts spaces

MIXING TABS AND SPACES IS RIGHT OUT.
(that means don't do it.)

:retab	Replace all tabs with spaces according to current tabstop setting
fileformat ff	Try changing this if your line-endings are messed up
list	Display whitespace visibly according to listchars

Help (:h help)

:h *cmd*	Normal mode *cmd* help
:h i_*cmd*	Insert mode *cmd* help
:h v_*cmd*	Visual mode *cmd* help
:h c_*cmd*	Command-line editing *cmd* help
:h :*cmd*	Command-line *cmd* help
:h '*option*'	*Option* help
:helpgrep	Search through all help docs!

Tags (:h tags-and-searches)

^]	Jump to tag under cursor, including [tags] in help files
^t	Jump back up the tag-list
g^]	Jump to tag if it's the only match; else list matching tags

Keycodes (:h keycodes)

<CR>	^m	\r	Enter
<Tab>	^i	\t	Tab
<C-*n*>	^*n*		Ctrl-*n*
<M-*n*>			Alt-*n*
<Esc>	^[		Escape
<BS>	^h	\b	Backspace
<Del>			Delete

Word motions (:h word-motions)

7 words
http://www.vimcheatsheet.com
1 WORD

SEARCHING

:h pattern-searches

Prev	Next	Forward	Backward	Matches
N	n	/*foo*	?*foo*	*foo*
N	n	*	#	word under cursor
;	,	t*x*	T*x*	upto *x*
;	,	f*x*	F*x*	find *x*

Marks (:h mark-motions)

m*m*	set mark *m* (a-z) in file
m*M*	set mark *M* (A-Z) across files
'[	jump to first char of just-changed text
'*m*	jump to first char of line containing *m*
`*m*	jump to exact character of *m*
''	jump back to last jump

Pass a directory to the `:edit` command to open a directory explorer. Instructions for usage are at the top of the screen.

Normal mode

p	paste after cursor	P	paste before cursor	^[	return to Normal mode
u	undo	^r	redo	.	repeat
gf	find file under cursor in path and jump to it	dd	delete current line	yy	yank current line
x	delete character after cursor	%	jump to matching paren	r	replace char under cursor
*n*G	jump to line *n*	^o	jump back	^i	jump forward
zz	center screen on cursor	zt	align top of screen with cursor	zb	align bottom of screen with cursor
==	auto-indent current line	<<	shift current line left by shiftwidth	>>	shift current line right by shiftwidth

Using `^[` to return to Normal mode lets you keep your fingers on the home row. It's even easier if you map Caps Lock to Control!

ENTERING INSERT MODE

I	beginning of line	i	before cursor	a	after cursor	A	end of line		
O	previous line	o	next line	s	substitute character	S	substitute line	C	line from cursor

COOL INSERT MODE STUFF

:h insert.txt

^w	delete word before cursor	^u	delete line before cursor
^r*r*	insert the contents of register *r*	^r=	use the expression register (try `^r=5+10`)
^t	increase line indent by shiftwidth	^d	decrease line indent by shiftwidth
^x^l	line completion	^n	find next completion suggestion according to complete

ENTERING VISUAL (SELECT) MODE

v	The most basic type. Use Visual mode to select characters within a line.
V	Useful for moving chunks of a program around the file. Use Visual Line mode to select one or more lines.
^v	Great for working with tables made of text, or anything that happens to be conveniently aligned. Visual Block mode can be used to select boxes across lines.

o	switch cursor to start/end	:h v_o
gv	re-select previous area	:h gv
I	prepend to each Visual block line	:h v_b_I
'<	jump to start of prior area	:h '<

COMMAND-LINE MODE ONLY

:h cmdline.txt

^f	edit using Normal mode	cmdwin
^r^w	insert word under cursor	cmdline-editing
^d	completion suggestions	cmdline-completion

Put `cnoremap %% <C-R>=expand('%:h').'/'<CR>` in your `.vimrc` so you can type `%%` in Command-line mode to refer to the directory of the current file, regardless of `pwd`.

Supply `%` as a range to the `:substitute` command to run it on every line in the file.
`:%s/Scribbl/Design/` — "Scribbled" -> "Designed"

Specify the "`g`" flag to apply the substitution to *every* match on a line.
`:s/[dla]//g` — "badly" -> "by" (:h s_flags, :h /[])

Vim supports many regular expression features.
`:s/..k/ax/` — "Mook" -> "Max" (:h usr_27, :h /.)

Use `\_.` instead of `.` if you want to search across multiple lines.
`:%s/heat\_.*Bungle/anto/` — "Cheatsheet\nBungler" -> "Cantor" (:h /_.)

Special escapes can be used to change the case of substitutions.
`:s_\(f..\)_\U\1\E_` — "foobar" -> "FOObar" (:h sub-replace-special)

Use `:global` to perform a command on matching lines.
`:g/foobar/delete` — Delete all lines containing "foobar"

If your pattern contains slashes, just use a different character as your delimiter.
`:s_Data/Lore_Brent Spiner_`

Use `\=` to evaluate expressions with replacement groups.
`:s_\d_\=submatch(0) + 1_g` — "10 25" -> "21 36" (:h sub-replace-\=)

Files

ZZ	Write current file, if modified, and quit
ZQ	Quit without checking for changes (like `:q!`)
:write	Write current file
:wq	Write current file and quit

Use `:scriptnames` to list all files sourced during initialization.

:syntax	Enable and configure syntax highlighting. Use `:sy sync fromstart` to redraw broken highlights
:make	Run a compiler and enter quickfix mode (:h quickfix)
:!	Execute external shell command
!	Filter motion with shell command

Use `:earlier` and `:later` to quickly jump backward and forward in a file's history.

:read	Read external program output into current file

Options (:h options)

:set *opt*?	View current value of *opt*
:set no*opt*	Turn off flag *opt*
:set *opt*	Turn on flag *opt*
:set *opt*=*val*	Overwrite value of *opt*
:set *opt*+=*val*	Append to value of *opt*
:echo &*opt*	Access *opt* as a variable

hidden	hid	Lets you switch buffers without saving
laststatus	ls	Show status line never (0), always (2) or with 2+ windows (1)
hlsearch	hls	Highlight search matches. Also see 'highlight'
number	nu	Show line numbers
showcmd	sc	Show commands as you type them
ruler	ru	Show line and column number of the cursor
backspace	bs	Set to '2' to make backspace work like sane editors
wrap		Control line wrapping
background	bg	Set to 'dark' if you have a dark color scheme

Use `a` instead of `i` when beginning text-object motions to include delimiters or surrounding whitespace. For example, `di(` will change "(foo)" into "()", but `da(` will delete the parentheses as well.

Buffers (:h buffers)

:ls	List all open files
:b *path*	Jump to unique file matching *path*. Use `<Tab>` to scroll through available completions!
:b*n*	Jump to file *n*, number from first column of :ls
:bnext	Jump to next file
:bprev	Jump to previous file
:bdelete	Remove file from the buffer list
:edit	Open a file for editing
:enew	Open a blank new file for editing

Windows (:h windows)

:split	Split current window horizontally
:vsplit	Split current window vertically
^w hjkl	Move cursor to window left, below, above or to the right of the current window
^w HJKL	Move current window to left, bottom, top, or right of screen
^w r	Rotate windows clockwise
^w +-<>	Increase/decrease current window height/width
^w T	Move current window to a new tab
:only	Close all windows except current window
:bufdo	Execute a command in each open file

REGISTERS are CLIPBOARDS

All commands that delete, copy, or paste text use registers. To change which register is used by a command, type the register before the command. The default register is called "the unnamed register", and it is invoked with a pair of double-quotes (`""`). Typing `dd` or `yy` is the same as typing `""dd` or `""yy`. Think of the first `"` as a short way of saying "register", so `""` is pronounced "register `"`", and `"a`, "register `a`".

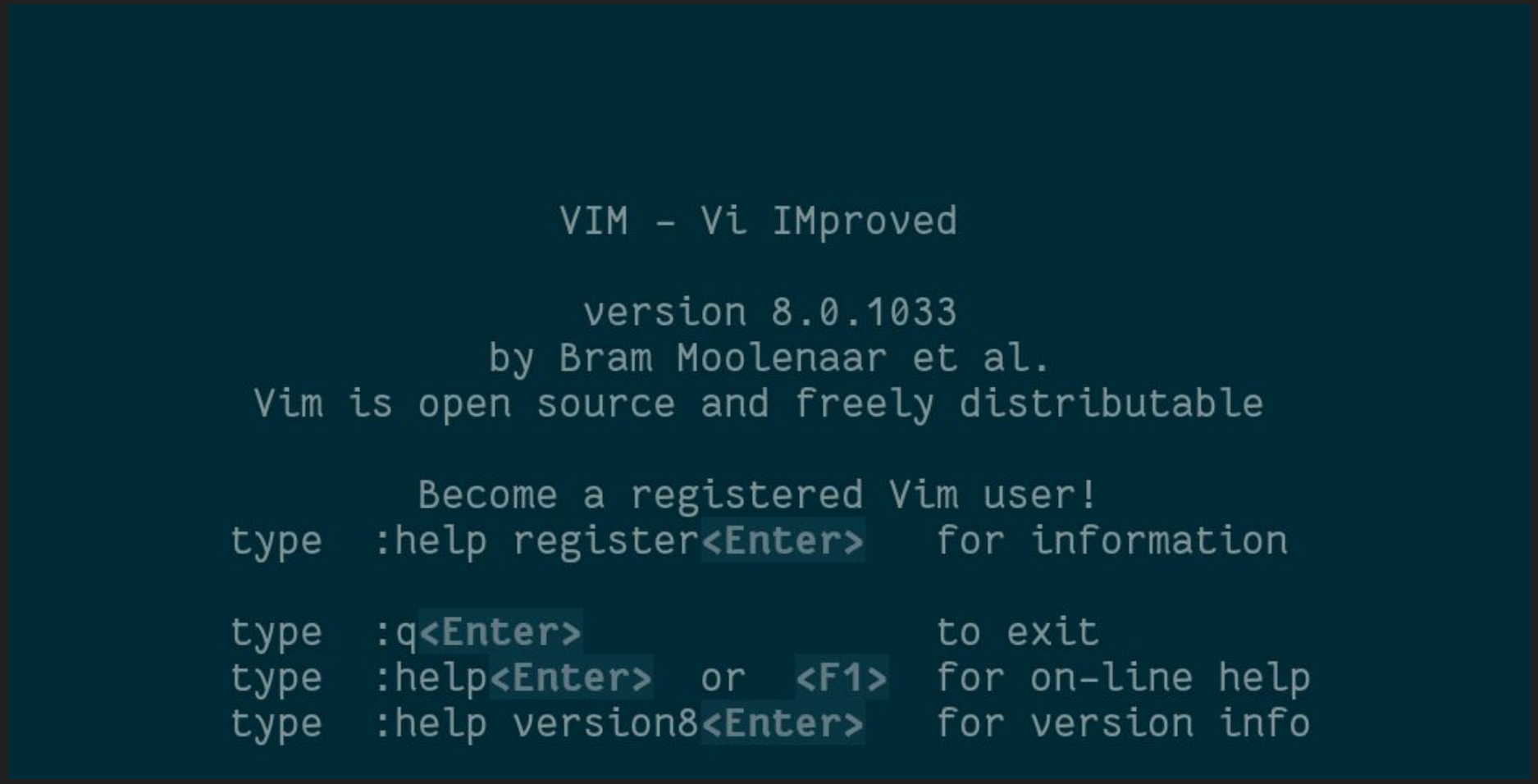
:h registers

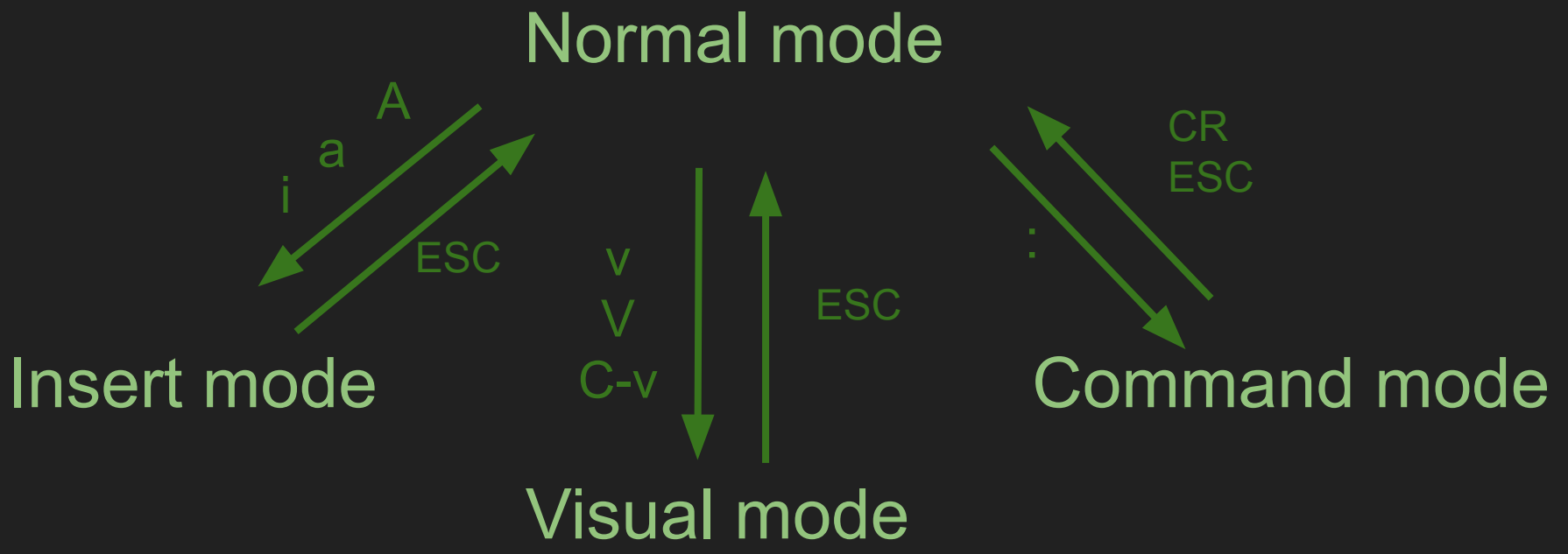
:registers		View all current registers
:echo @*r*		Access register *r* as a variable
"/	Last search pattern register	Contains the last pattern you searched for
"_	The black hole register	Use this to delete without clobbering any register (`"_dd`)
"0	Last yank register	Contains the last text you yanked
"1	Last big delete register	Contains the last line(s) you deleted
"2-"9	Big delete register stack	Every time `"1` is written to, its content is pushed to `"2`, then `"2` to `"3`, and so on
"-	Small delete register	Contains the last text you deleted within a single line
"+	System clipboard	If the OS integration gods smile upon you, this register reads and writes to your system clipboard.
"a-"z	Named registers	26 registers for you to play with
"A-"Z	Append registers	Using upper-case to refer to a register will append to it rather than overwrite it
q*r*	Record	Record into register *r*. Stop recording by hitting `q` again
@*r*	Playback	Execute the contents of register *r*
@@	Repeat last playback	Repeat the last @*r*, this is particularly useful with a count

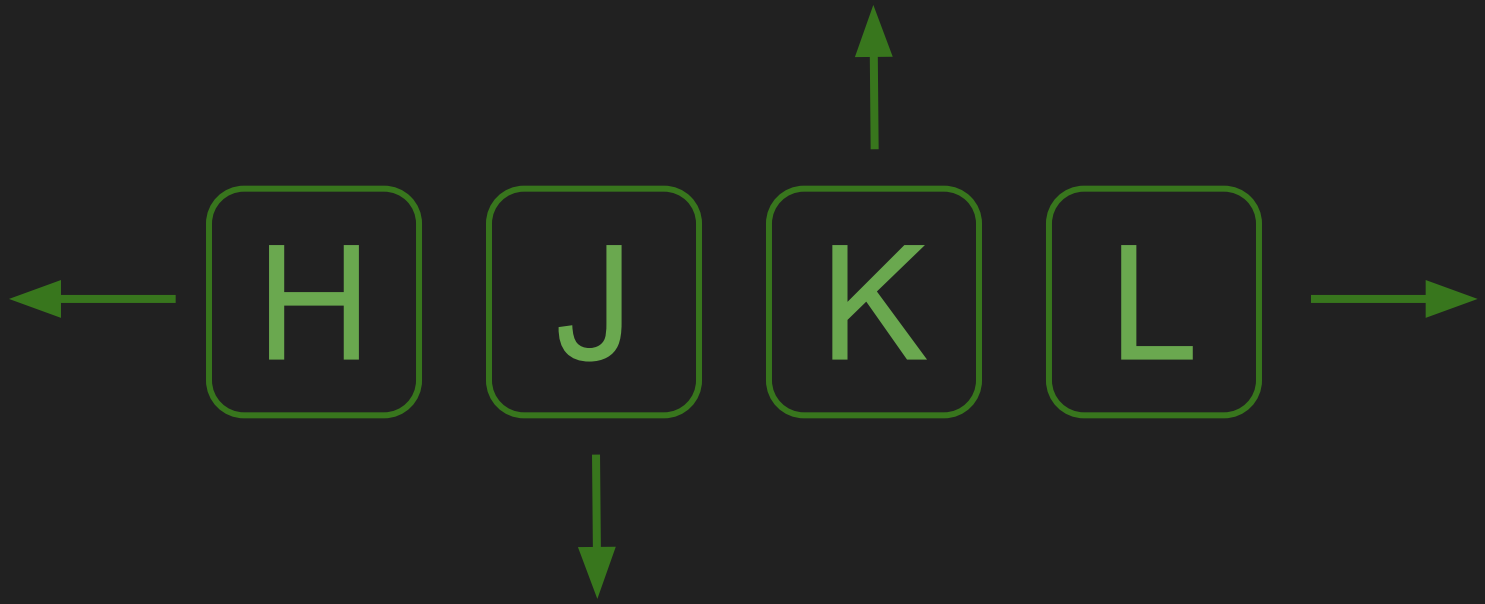
Use `:map` to view all current custom key mappings. Read :h map-which-keys for a guide on which keys are best for your own custom mappings. Get used to Vim's help system - it's a fantastic resource!

1st Edition funded by...

How


VIM - Vi IMproved
version 8.0.1033
by Bram Moolenaar et al.
Vim is open source and freely distributable
Become a registered Vim user!
type :help register<Enter> for information
type :q<Enter> to exit
type :help<Enter> or <F1> for on-line help
type :help version8<Enter> for version info


Normal mode
A
a
i
ESC
v
V
C-v
ESC
CR
ESC
:
Insert mode
Visual mode
Command mode


H
J
K
L


ESC
CTRL
SHIFT
LINE FEED
RETURN
HOME
RUB
BREAK
SHIFT
REPEAT
CLEAR

Enter insert mode at cursor

Enter insert mode after cursor

Enter insert mode at end of line

Enter insert mode at start of line

Enter normal mode

Delete character under cursor

Paste before cursor

Paste after cursor

[operator][count][motion]

"Yank" (copy)

Delete

Change

word

back

end

WORD

BACK

END

Start of line

End of line

:w save :q quit

:q! force quit ZQ

:wq save and quit ZZ

:edit

:tabedit

:make

:!*cmd*

:set *option*

:split

:vsplit

~/.vimrc

~/.vim/

$ vimtutor

:help *cmd*

dmenu.vim

neocomplete.vim

vim-lion

vim-gitgutter

vim-eunuch

vim-buftabline

vim-commentary

vim-fugitive

vim-repeat

vim-surround

vim-unimpaired

vim-rsi

:inoremap jj <Esc>

:set number

:set clipboard=unnamedplus

:set spell spelllang=en_us

inoremap (()<Left>