Movement

h j k l	Left, down, up, right
^	Go to the beginning of the line
$	Go to the end of the line
gg	Go to the first line
G	Go to the last line
:n	Go to line n

Replacing Text

r	Replace character
cw	Change word
cc	Change line
c{motion}	Change from cursor to {motion}

Copy/Paste

y{motion}	Yank {motion}
yy	Yank line
p	Paste after cursor
P	Paste before cursor

Searching

/{pattern}	Forward search for {pattern}
?{pattern}	Reverse search for {pattern}
n	Repeat the last search
N	Repeat the last search in the opposite direction

Repeating Commands

{num}{command}	Repeat command {num} times
.	Repeat previous change

Inserting Text

i	Insert at cursor
I	Insert at the beginning of the line
a	Append after cursor
A	Append at the end of the line
o	Open a new line below the current line
O	Open a new line above the current line

Deleting Text

x	Delete character
dd	Delete line
dw	Delete word
d{motion}	Delete {motion}

Undo/Redo

u	Undo
Ctrl-r	Redo

Save and Quit

:w	Write (save)
:wq	Write and quit
:q	Quit
:q !	Force quit, don't save changes
:wq!	Force write and quit

Help

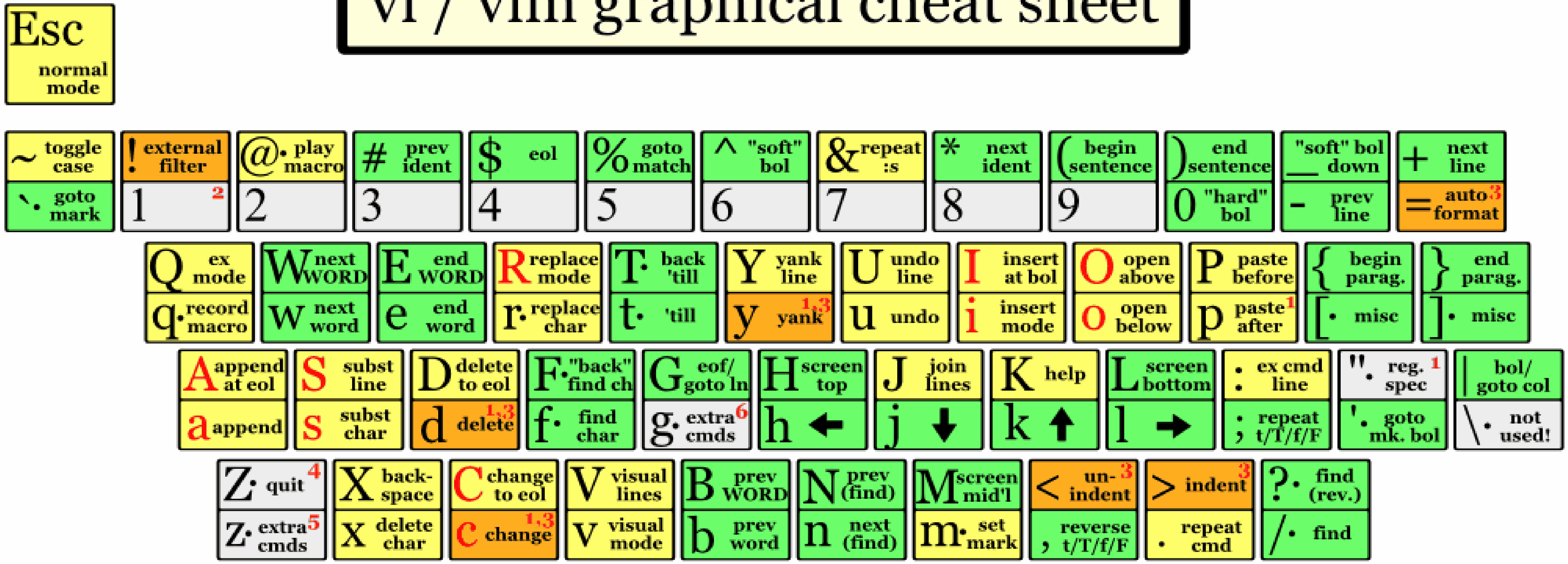
:help [topic/command]	Get help on topic or command.
vimtutor	Tutorial

Find and Replace

:s/{old}/{new}/{options}	Substitute {new} for {old} on the current line
:%s/{old}/{new}/{options}	Substitute {new} for {old} in the entire document
	The g option substitutes all occurrences on a line, otherwise just the first occurrence is changed per line.

http://www.LinuxTrainingAcademy.com

vi / vim graphical cheat sheet



🟩 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
q·	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/x/y/g (replace 'x' by 'y' filewide),
:h (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, d2w, 5i, d4j)

(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), gf: open file under cursor (vim only)

:h operator   :h navigation

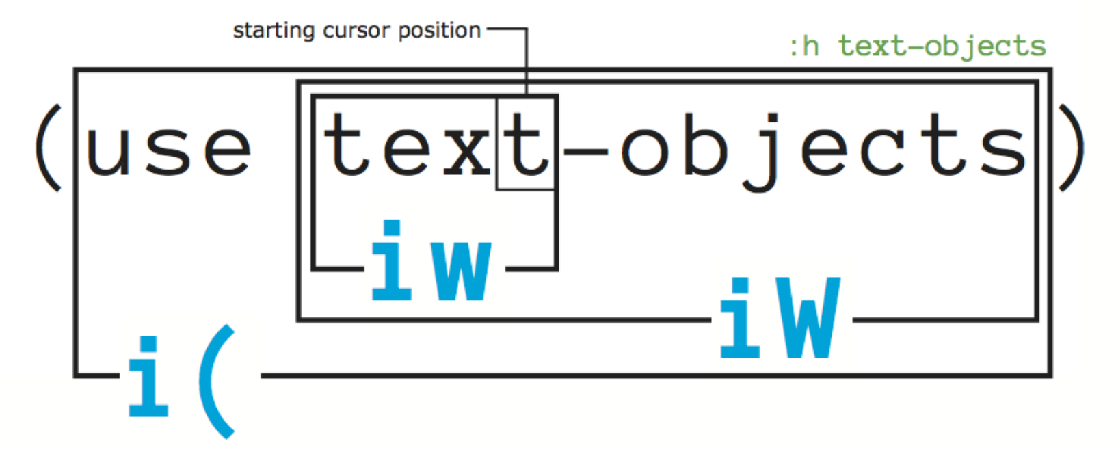
[operator][count][motion]

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



: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

	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 *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!

: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

:h left-right-motions

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

: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

:h pattern-searches

SEARCHING

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

: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.

p	paste after cursor	P	paste before cursor	^[	return to Normal mode
u	undo	^r	redo	.	repeat
gf (file-searching)	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-motions)	jump back	^i (jump-motions)	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!

: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

ENTERING INSERT MODE

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

:h insert.txt

COOL INSERT MODE STUFF

^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

: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

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

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.

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

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

:h cmdline.txt

COMMAND-LINE MODE ONLY

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

: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

vim one-liner used to sort the list of names by length:

```
:exe 'g/^/let @x = len(getline(".")) | normal "xPa ' | sort n | :g//normal dw
```