

vi / vim graphical cheat sheet

Esc
normal mode

~ toggle case	! external filter	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	& repeat :s	* 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 ³ 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 ^{1,3}	u undo	i insert mode	o open below	p paste after ¹	[misc]	misc
A append at eol	S subst line	D delete to eol	F "back" find ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	.	ex cmd line	! reg. ¹ spec	bol/ goto col
a append	s subst char	d delete ^{1,3}	f find char	g extra ⁶ cmds	h ←	j ↓	k ↑	l →	.	repeat t/T/f/F	' goto mk. bol	\ not used!
Z quit ⁴	X back-space	C change to eol	V visual lines	B prev WORD	N prev (find)	M screen mid'l	< un- ³ indent	> indent ³	?	find (rev.)		
Z extra ⁵ cmds	X delete char	c change ^{1,3}	V visual mode	b prev word	n next (find)	m set mark	.	reverse t/T/f/F	.	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
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)

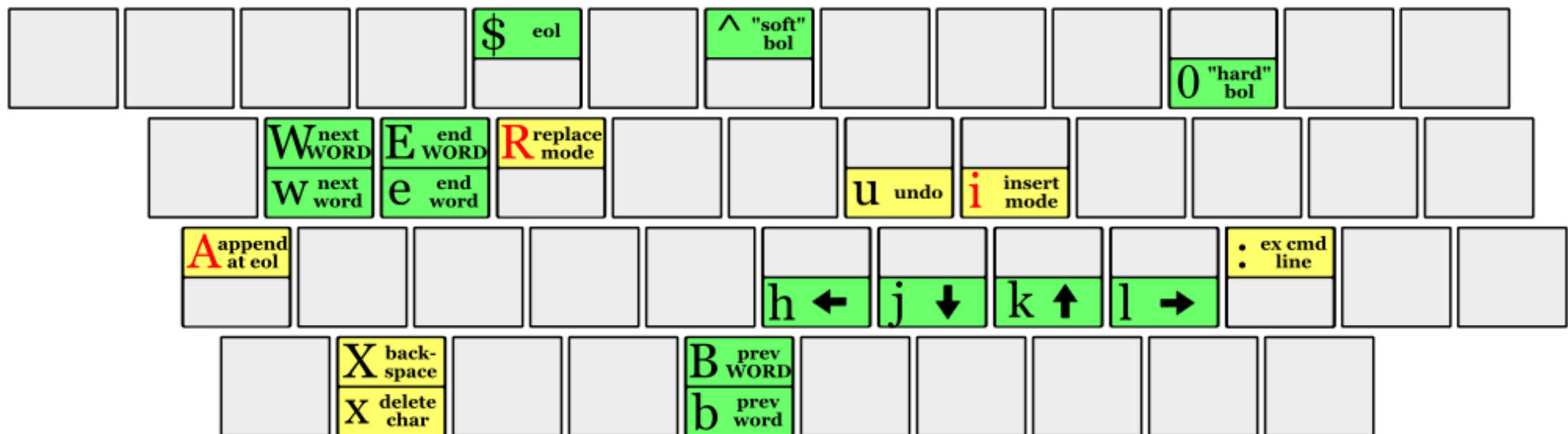
Notes:

- (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)

vi/vim lesson 1 - basic editing

motion moves the cursor, or defines the range for an operator
command direct action command, if **red**, it enters insert mode

Esc
normal mode



Basics:

h j k l are vi/vim cursor keys – use them as they are much closer than regular cursor keys!

Use **i** to enter insert mode, cursor turns from a block into a vertical line, and you can type in text. Use **Esc** to return to normal mode.

Use **x** to delete the current character, or **X** to delete the one to the left

Use **A** to go insert text at the end of the line (wherever you are in the line!)

(Note: insert mode is actually very similar to a regular editor, you can use cursor/navigation keys, backspace, delete...)

Extras:

u to undo the last action – traditional vi has a single level, while vim supports unlimited undo (CTRL - **R** to redo)

0 jumps directly to the beginning of the line, **\$** to the end, and **^** to the first non-blank

Use **w b e** to move along 'words'. A 'word' is a sequence of all alphanumeric or punctuation signs: `quux(foo, bar, baz);`

Use **W B E** to move along WORDs. A 'WORD' is a sequence of any non-blank characters: `quux(foo, bar, baz);`

Use **R** to enter insert mode with an overstrike cursor, which types over existing characters.

: **w** and press enter to save, **:** **q** and enter to quit.

vi/vim lesson 2 - operators & repetition

learned	learned in previous lessons
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

Esc
normal mode

				\$ eol		^ "soft" bol							
	1	2	3	4	5	6	7	8	9	0 "hard" bol			
		W next word	E end word	R replace mode	T back 'till								
		w next word	e end word		t 'till			u undo	i insert mode				
	A append at eol				F "back" find ch					.	ex cmd line		
				d delete	f find char			h ←	j ↓	k ↑	l →		
		X back-space			V visual lines	B prev word							
		X delete char		C change	V visual mode	b prev word					.	repeat cmd	

Basics:

f, followed by another key, moves the cursor to the next instance of that character on the current line, **F** does the same backwards.

t and **T** do the same, but they stop right before the character.

d (delete), followed, by any motion deletes the text between the cursor and that motion's destination **d w**, **d f** - ...).

c (change) does the same, but leaves you in insert mode.

Some motions, such as **j** and **k**, are linewise – deletion includes the full start/end lines.

. repeats the last editing action: text input, delete or change, etc... motion is recalculated at the new place.

Extras:

Prepend a count to any command/motion to repeat it that number of times:

d 2 w to delete up to the second word.

d 2 t , to delete up to but not including the second comma.

2 i repeats the text after you press (Esc) to finish the input session.

Repeat operator (**c c** or **d d**) to operate on the current line.

Only in vim, **v** enters visual mode. Move around with motions, the text will be highlighted. Press an operator to operate on that selection.

V enters visual-lines mode – like **v**, but selecting whole lines.

CTRL - **v** selects rectangular blocks.

vi/vim lesson 3 - yank & paste

Esc
normal mode

learned	learned in previous lessons
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

				\$ col		^ "soft" bol							
	1	2	3	4	5	6	7	8	9	0 "hard" bol			
		W next word	E end word	R replace mode	T back 'till				O open above	P paste before			
		w next word	e end word		t 'till	y yank	u undo	i insert mode	O open below	p paste after			
A append at eol				F "back" find ch					.	ex cmd line	" reg. spec		
			d delete	f find char		h ←	j ↓	k ↑	l →				
		X back-space		V visual lines	B prev word								
		X delete char	c change	V visual mode	b prev word					.	repeat cmd		

Basics

Use **y** followed by any motion to 'yank' (copy).

Use **p** to paste after (if charwise, to the right, if linewise, below).

Use **P** to paste before.

y y copies the current line.

y also works in visual mode.

Text deleted with **d**, **c**, **x** ... is also copied!

Extras

" and an **a** - **z** character before any yank/delete/paste command chooses a register.

An **A** - **Z** register before yank/delete means "append-copy".

"* or **"+** select the system clipboard.

o enters insert mode in a new empty line below the current one.

O does the same above the current line.

vi/vim lesson 4 - searching

Esc
normal mode

learned	learned in previous lessons
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

	1	2	# prev ident	\$ eol		^ "soft" bol		* next ident				
										0 "hard" bol		
		W next word	E end word	R replace mode	T back 'till				O open above	P paste before		
		w next word	e end word		t 'till	y yank	u undo	i insert mode	o open below	p paste after		
A append at eol					F "back" find ch				.	ex cmd line	!! reg. 1 spec	
			d delete 1-3	f find char		h ←	j ↓	k ↑	l →			
	X back-space			V visual lines	B prev word	N prev (find)					?. find (rev.)	
	X delete char	c change 1-3		V visual mode	b prev word	n next (find)			.	repeat cmd	/ find	

Basics:

/ is the basic search motion – type the text you are searching for after the slash, and then press return. Being a motion, you can use this after an operator, or in visual mode.

? does the same, backwards.

n repeats the last search in the same direction, **N** repeats it in the reverse direction

Be careful, because the search target is interpreted as a regular expression: **a*b** means zero or more 'a's followed by a 'b', **^abc** means 'abc' at the beginning of a line, **[0-9]** looks for the next digit, etc...

Extras:

The following very useful motions work only in vim:

***** searches forward for the next instance of the identifier under the cursor.

does the same backwards.

vi/vim lesson 5 - marks & macros

Esc
normal mode

learned	learned in previous lessons
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

		@ play macro	# prev ident	\$ eol		^ "soft" bol		* next ident				
goto mark	1	2	3	4	5	6	7	8	9	0 "hard" bol		
	q record macro	W next word	E end word	R replace mode	T back 'till				O open above	P paste before		
		w next word	e end word		t 'till	y yank	u undo	i insert mode	o open below	p paste after		
	A append at eol				F "back" find ch				.	ex cmd line	! reg. spec	
				d delete	f find char		h ←	j ↓	k ↑	l →	! goto mk. bol	
		X back-space			V visual lines	B prev word	N prev (find)					
		X delete char		c change	v visual mode	b prev word	n next (find)	m set mark		.	repeat cmd	? find (rev.)
											/ find	

Marks:

- Use **m** followed by an **a** - **z** character to set a mark.
- Use **`** followed by a character to go to that mark.
- Use **^** and a character to go to the first non-blank in that line.
- A** - **Z** marks are global, **a** - **z** per-buffer.
- `** **.** refers to the position of the last modification.

Macros:

- Use **q** followed by an **a** - **z** character to start recording.
- Use **q** afterwards to stop recording.
- @** followed by a character replays that macro.
- @ @** to repeat the last macro played.

vi/vim lesson 6 – various motions

Esc
normal mode

learned	learned in previous lessons
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

goto mark	1	2	@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol	* next ident	(begin sentence) end sentence	- prev line	+ next line
q record macro	W next word	E end word	R replace mode	T back 'till	t 'till	y yank	u undo	i insert mode	O open above	P paste before	{ begin parag.	} end parag.
A append at eol	d delete	f find char	F "back" find ch	G eof/ goto ln	H screen top	K help	L screen bottom	ex cmd line	reg. spec	goto mk. bol		
X back-space	X delete char	c change	V visual lines	B prev word	N prev (find)	M screen mid'l	m set mark	repeat cmd	find (rev.)			

- %** jumps between matching pairs of '(', ')', '[', ']', etc...
- H M L** jump directly to the top/middle/bottom of the screen.
- G** jumps to the end of the file, or to the line # typed before it.
- / +** jump to the previous/next line.
- K**, not technically a motion, jumps to the help for the word under the cursor: vim help, man page under unix, etc...

- (** and **)** jump to the beginning/end of the current sentence.
- {** and **}** jump to the previous/next empty line.
- [** **]** jumps to the previous ']' in column 0.
- [** **]** jumps to the next '[' in column 0.

vi/vim lesson 7 - various commands

Esc
normal mode

learned	learned in previous lessons
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

~ toggle case		@ play macro	# prev ident	\$ eol	% goto match	^ "soft" bol		* next ident	(begin sentence) end sentence		+ next line
\ goto mark	1	2	3	4	5	6	7	8	9		- prev line	= auto format
q record macro	W next word	E end word	R replace mode	T back 'till	Y yank line			O open above	P paste before	{ begin parag.	}	end parag.
	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 ch	G eof/ goto ln	H screen top	J join lines	K help	L screen bottom	.	ex cmd line	! reg. spec	
	S subst char	d delete	f find char		h ←	j ↓	k ↑	l →			! goto mk. bol	
	X back-space	C change to eol	V visual lines	B prev word	N prev (find)	M screen mid'l	< un-indent	> indent	?	find (rev.)		
	X delete char	c change	v visual mode	b prev word	n next (find)	m set mark		.	repeat cmd	/ find		

Basics:

- J** joins the current line with the next one, or all the lines in the current visual selection.
- r** followed by any character replaces the current character with that one.
- C** is shorthand for **c** **\$**, changes to end of line.
- D** is shorthand for **d** **\$**, deletes to end of line.
- Y** is shorthand for **y** **\$**, yanks the whole line.
- s** deletes the character under the cursor and enters insert mode.
- S** clears the current line and enters insert mode.

Extras:

- >** and a motion to indent one or more lines.
- <** and a motion to unindent.
- =** and a motion to reformat a range of text.

All of them work in visual mode, or can be repeated (**>>**, etc...) to operate on the current line.

- ~** toggles the case of the character under the cursor.

Now go grab the full cheat sheet and learn the rest.
Start with **I** **a** **,** and **;**. Piece of cake!