Exit

:q[uit]	Quit Vim. This fails when changes have been made.
:q[uit]!	Quit without writing.
:cq[uit]	Quit always, without writing.
:wq	Write the current file and exit.
:wq!	Write the current file and exit always.
:wq {file}	Write to {file}. Exit if not editing the last
:wq! {file}	Write to {file} and exit always.
:[range]wq[!]	[file] Same as above, but only write the lines in [range].
ZZ	Write current file, if modified, and exit. (Command Mode)
ZQ	Quit current file and exit (same as ":q!"). (Command Mode)

Editing a File

:e[dit]	Edit the current file. This is useful to re-edit the current file, when it has been changed outside of Vim.
:e[dit]!	Edit the current file always. Discard any changes to the current buffer. This is useful if you want to start all over again.
:e[dit] {file}	Edit {file}.
:e[dit]! {file}	Edit {file} always. Discard any changes to the current buffer.
gf	Edit the file whose name is under or after the cursor. Mnemonic: "goto file". (Command Mode)

Start Insert Mode

a	Append text after the cursor [count] times.
A	Append text at the end of the line [count] times.
i	Insert text before the cursor [count] times.
I	Insert text before the first non-blank in the line [count] times.
gI	Insert text in column 1 [count] times.
o	Begin a new line below the cursor and insert text, repeat [count] times.
O	Begin a new line above the cursor and insert text, repeat [count] times.

Inserting into file

:r[ead] [name]	Insert the file [name] below the cursor.
:r[ead] !{cmd}	Execute {cmd} and insert its standard output below the cursor.

Deleting Text

<Del> or x	Delete [count] characters under and after the cursor
X	Delete [count] characters before the cursor
d{motion}	Delete text that {motion} moves over
dd	Delete [count] lines
D	Delete the characters under the cursor until the end of the line
{Visual}x or {Visual}d	Delete the highlighted text (for {Visual} see Selecting Text).
{Visual}CTRL-H or {Visual}	When in Select mode: Delete the highlighted text
{Visual}X or {Visual}D	Delete the highlighted lines
:[range]d[elete]	Delete [range] lines (default: current line)
:[range]d[elete] {count}	Delete {count} lines, starting with [range]

Changing (or Replacing) Text

r{char}	replace the character under the cursor with {char}.
R	Enter Insert mode, replacing characters rather than inserting
~	Switch case of the character under the cursor and move the cursor to the right. If a [count] is given, do that many characters.
~{motion}	switch case of {motion} text.
{Visual}~	Switch case of highlighted text

Substituting

:[range]s[ubstitute]/{pattern}/{string}/[c] [e][g][p][r][i][I] [count]	For each line in [range] replace a match of {pattern} with {string}.
:[range]s[ubstitute] [c][e][g][r][i][I] [count] :[range]&[c][e][g][r][i][I] [count]	Repeat last :substitute with same search pattern and substitute string, but without the same flags. You may add extra flags

```
The arguments that you can use for the substitute commands:
[c]  Confirm each substitution.  Vim positions the cursor on the matching
  string.  You can type:
      'y'      to substitute this match
      'n'      to skip this match
         to skip this match
      'a'      to substitute this and all remaining matches {not in Vi}
      'q'      to quit substituting {not in Vi}
      CTRL-E  to scroll the screen up {not in Vi}
      CTRL-Y  to scroll the screen down {not in Vi}.
[e]     When the search pattern fails, do not issue an error message and, in
  particular, continue in maps as if no error occurred.
[g]  Replace all occurrences in the line.  Without this argument,
  replacement occurs only for the first occurrence in each line.
[i]  Ignore case for the pattern.
[I]  Don't ignore case for the pattern.
[p]  Print the line containing the last substitute.
```

Copying and Moving Text

"{a-zA-Z0-9.%#:-" }	Use register {a-zA-Z0-9.%#:-"} for next delete, yank or put (use uppercase character to append with delete and yank) ({.%#:} only work with put).
:reg[isters]	Display the contents of all numbered and named registers.
:reg[isters] {arg}	Display the contents of the numbered and named registers that are mentioned in {arg}.
:di[splay] [arg]	Same as :registers.
["x]y{motion}	Yank {motion} text [into register x].
["x]yy	Yank [count] lines [into register x]
["x]Y	yank [count] lines [into register x] (synonym for yy).
{Visual}["x]y	Yank the highlighted text [into register x] (for {Visual} see Selecting Text).
{Visual}["x]Y	Yank the highlighted lines [into register x]
:[range]y[ank] [x]	Yank [range] lines [into register x].
:[range]y[ank] [x] {count}	Yank {count} lines, starting with last line number in [range] (default: current line), [into register x].
["x]p	Put the text [from register x] after the cursor [count] times.
["x]P	Put the text [from register x] before the cursor [count] times.
["x]gp	Just like "p", but leave the cursor just after the new text.
["x]gP	Just like "P", but leave the cursor just after the new text.
:[line]pu[t] [x]	Put the text [from register x] after [line] (default current line).
:[line]pu[t]! [x]	Put the text [from register x] before [line] (default current line).

Undo/Redo/Repeat

u	Undo [count] changes.
:u[ndo]	Undo one change.
CTRL-R	Redo [count] changes which were undone.
:red[o]	Redo one change which was undone.
U	Undo all latest changes on one line. {Vi: while not moved off of it}
.	Repeat last change, with count replaced with [count].

Moving Around

```
Basic motion commands:

          k
        h    l
          j
```

h or	[count] characters to the left (exclusive).
l or	[count] characters to the right (exclusive).
k or CTRL-P	[count] lines upward
j or CTRL-J or CTRL-N	[count] lines downward (linewise).
0	To the first character of the line (exclusive).
<Home>	To the first character of the line (exclusive).
^	To the first non-blank character of the line
$ or <End>	To the end of the line and [count - 1] lines downward
g0 or g<Home>	When lines wrap ('wrap on): To the first character of the screen line (exclusive). Differs from "0" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the leftmost character of the current line that is on the screen. Differs from "0" when the first character of the line is not on the screen.
g^	When lines wrap ('wrap' on): To the first non-blank character of the screen line (exclusive). Differs from "^" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the leftmost non-blank character of the current line that is on the screen. Differs from "^" when the first non-blank character of the line is not on the screen.

g$ or g<End&gr;	When lines wrap ('wrap' on): To the last character of the screen line and [count - 1] screen lines downward (inclusive). Differs from "$" when a line is wider than the screen. When lines don't wrap ('wrap' off): To the rightmost character of the current line that is visible on the screen. Differs from "$" when the last character of the line is not on the screen or when a count is used.
f{char}	To [count]'th occurrence of {char} to the right. The cursor is placed on {char} (inclusive).
F{char}	To the [count]'th occurrence of {char} to the left. The cursor is placed on {char} (inclusive).
t{char}	Till before [count]'th occurrence of {char} to the right. The cursor is placed on the character left of {char} (inclusive).
T{char}	Till after [count]'th occurrence of {char} to the left. The cursor is placed on the character right of {char} (inclusive).
;	Repeat latest f, t, F or T [count] times.
,	Repeat latest f, t, F or T in opposite direction [count] times.
- <minus>	[count] lines upward, on the first non-blank character (linewise).
+ or CTRL-M or <CR>	[count] lines downward, on the first non-blank character (linewise).
_ <underscore>	[count] - 1 lines downward, on the first non-blank character (linewise).
<C-End> or G	Goto line [count], default last line, on the first non-blank character.
<C-Home> or gg	Goto line [count], default first line, on the first non-blank character.
<S-Right> or w	[count] words forward
<C-Right> or W	[count] WORDS forward
e	Forward to the end of word [count]
E	Forward to the end of WORD [count]
<S-Left> or b	[count] words backward
<C-Left> or B	[count] WORDS backward
ge	Backward to the end of word [count]
gE	Backward to the end of WORD [count]

These commands move over words or WORDS.

A word consists of a sequence of letters, digits and underscores, or a sequence of other non-blank characters, separated with white space (spaces, tabs, A WORD consists of a sequence of non-blank characters, separated with white space. An empty line is also considered to be a word and a WORD.

(	[count] sentences backward
)	[count] sentences forward
{	[count] paragraphs backward
}	[count] paragraphs forward
]]	[count] sections forward or to the next '{' in the first column. When used after an operator, then the '}' in the first column.
][	[count] sections forward or to the next '}' in the first column
[[	[count] sections backward or to the previous '{' in the first column
[]	[count] sections backward or to the previous '}' in the first column

Marks

m{a-zA-Z}	Set mark {a-zA-Z} at cursor position (does not move the cursor, this is not a motion command).
m' or m`	Set the previous context mark. This can be jumped to with the "''" or "``" command (does not move the cursor, this is not a motion command).
:[range]ma[rk]{a-zA-Z}	Set mark {a-zA-Z} at last line number in [range], column 0. Default is cursor line.
:[range]k{a-zA-Z}	Same as :mark, but the space before the mark name can be omitted.
'{a-z}	To the first non-blank character on the line with mark {a-z} (linewise).
'{A-Z0-9}	To the first non-blank character on the line with mark {A-Z0-9} in the correct file
`{a-z}	To the mark {a-z}
`{A-Z0-9}	To the mark {A-Z0-9} in the correct file
:marks	List all the current marks (not a motion command).
:marks {arg}	List the marks that are mentioned in {arg} (not a motion command). For example:

Searching

/{pattern}[/]	Search forward for the [count]'th occurrence of {pattern}
/{pattern}/{offset}	Search forward for the [count]'th occurrence of {pattern} and go {offset} lines up or down.
/<CR>	Search forward for the [count]'th latest used pattern
//{offset}<CR>	Search forward for the [count]'th latest used pattern with new. If {offset} is empty no offset is used.
?{pattern}[?]<CR>	Search backward for the [count]'th previous occurrence of {pattern}
?{pattern}?{offset}<CR>	Search backward for the [count]'th previous occurrence of {pattern} and go {offset} lines up or down
?<CR>	Search backward for the [count]'th latest used pattern
??{offset}<CR>	Search backward for the [count]'th latest used pattern with new {offset}. If {offset} is empty no offset is used.
n	Repeat the latest "/" or "?" [count] times.
N	Repeat the latest "/" or "?" [count] times in opposite direction.

Selecting Text (Visual Mode)

To select text, enter visual mode with one of the commands below, and use motion commands to highlight the text you are interested in. Then, use some command on the text.

```
The operators that can be used are:
  ~  switch case
  d  delete
  c  change
  y  yank
  >  shift right
  <  shift left
  !  filter through external command
  =  filter through 'equalprg' option command
  gq  format lines to 'textwidth' length
```

v	start Visual mode per character.
V	start Visual mode linewise.
<Esc>	exit Visual mode without making any changes

How to Suspend

CTRL-Z	Suspend Vim, like ":stop". Works in Normal and in Visual mode. In Insert and Command-line mode, the CTRL-Z is inserted as a normal character.
:sus[pend][!] or :st[op][!]	Suspend Vim. If the '!' is not given and 'autowrite' is set, every buffer with changes and a file name is written out. If the '!' is given or 'autowrite' is not set, changed buffers are not written, don't forget to bring Vim back to the foreground later!

Esc Normal

Revision 2.0
Sept. 11, 2011

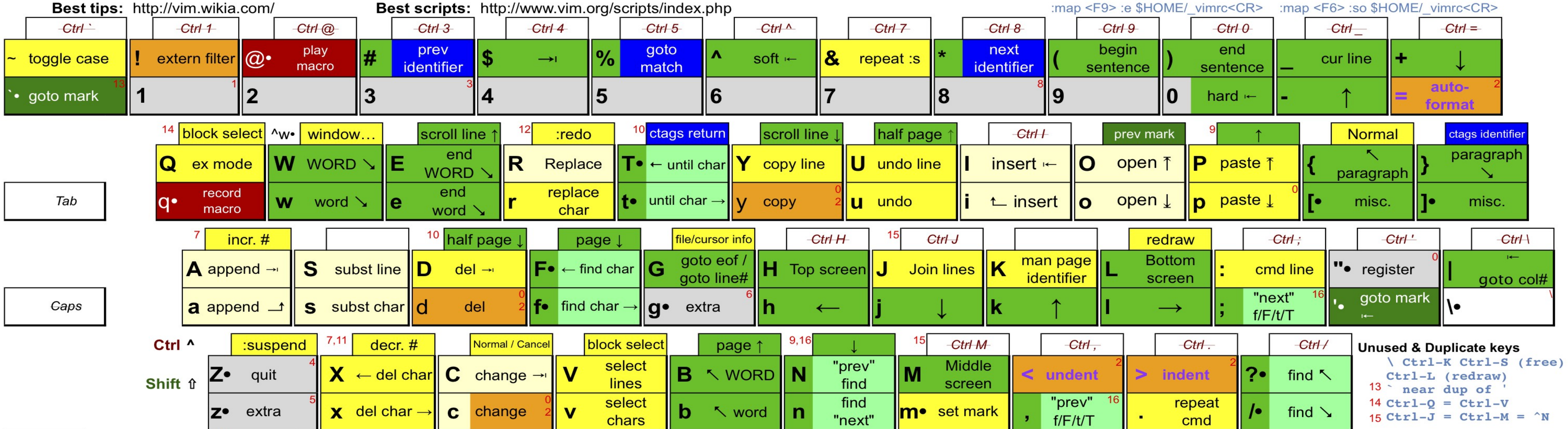
Vim 7.3+
`:version`

Vim Cheat Sheet for Programmers

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 `~~	~ toggle case	`• goto mark 13
~~Ctrl 1~~	! extern filter	1 [1]
~~Ctrl @~~	@• play macro	2
~~Ctrl 3~~	# prev identifier	3 [3]
~~Ctrl 4~~	$ →|	4
~~Ctrl 5~~	% goto match	5
~~Ctrl ^~~	^ soft |←	6
~~Ctrl 7~~	& repeat :s	7
~~Ctrl 8~~	* next identifier	8 [8]
~~Ctrl 9~~	(begin sentence	9
~~Ctrl 0~~	) end sentence	0 hard |←
~~Ctrl _~~	_ cur line	- ↑
~~Ctrl =~~	+ ↓	= auto-format [2]

Tab

14 block select	Q ex mode	q• record macro
^w• window…	W WORD ↘	w word ↘
scroll line ↑	E end WORD ↘	e end word ↘
12 :redo	R Replace	r replace char
10 ctags return	T• ← until char	t• until char →
scroll line ↓	Y copy line	y copy [0 2]
half page ↑	U undo line	u undo
~~Ctrl I~~	I insert |←	i ↰ insert
prev mark	O open ⤒	o open ⤓
9 ↑	P paste ⤒	p paste ⤓ [0]
Normal	{ ↖ paragraph	[• misc.
ctags identifier	} paragraph ↘	]• misc.

Caps

7 incr. #	A append →|	a append ⤴
	S subst line	s subst char
10 half page ↓	D del →|	d del [0 2]
page ↓	F• ← find char	f• find char →
file/cursor info	G goto eof / goto line#	g• extra [6]
~~Ctrl H~~	H Top screen	h ←
15 ~~Ctrl J~~	J Join lines	j ↓
	K man page identifier	k ↑
redraw	L Bottom screen	l →
~~Ctrl ;~~	: cmd line	; "next" f/F/t/T [16]
~~Ctrl '~~	"• register [0]	'• goto mark |←
~~Ctrl \\~~	| |← goto col#	\• [\\]

Ctrl ^
Shift ⇧

:suspend	Z• quit [4]	z• extra [5]
7,11 decr. #	X ← del char	x del char →
Normal / Cancel	C change →|	c change [0 2]
block select	V select lines	v select chars
page ↑	B ↖ WORD	b ↖ word
9,16 ↓	N "prev" find	n find "next"
15 ~~Ctrl M~~	M Middle screen	m• set mark
~~Ctrl ,~~	< undent [2]	, "prev" f/F/t/T [16]
~~Ctrl .~~	> indent [2]	. repeat cmd
~~Ctrl /~~	?• find ↖	/• find ↘

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

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.

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` `,` ƀ `dst` `,` ƀ `len` `);`

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

WORD `Foo ( src ,` ƀ `dst ,` ƀ `len );`

Startup

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

bug **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	`:r file`	insert file
`:q`	quit	`:gui`	switch to GUI
`:e <file>`	edit file in new buffer	`:q!`	quit w/o save
`: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

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

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

\ `: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

: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*