

:h up-down-motions																
	ts	sw	st	et	tabstop	ts	Columns per tabstop									
use spaces only	n	n	n	on	shiftwidth	sw	Columns per <<									
use tabs only	n	n	0	off	softtabstop	sts	Spaces per tab									
Set n to desired tab width (default 8)					expandtab	et	<Tab> inserts spaces									
<b>MIXING TABS AND SPACES IS RIGHT OUT.</b> (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											

next character	l	end of word	e	beginning of next word	w	end of WORD	E	beginning of next WORD	W	end of line	\$
	<b>P</b>	paste after cursor		<b>P</b>	paste before cursor		<b>^</b> [	return to Normal mode			
	<b>u</b>	undo		<b>^r</b>	redo		<b>.</b>	repeat			
<small>ctrl+u</small>	<b>gf</b>	find file under cursor in path and jump to it		<b>dd</b>	delete current line		<b>yy</b>	yank current line			
	<b>x</b>	delete character after cursor		<b>%</b>	jump to matching paren		<b>r</b>	replace char under cursor			
<small>ctrl+n</small>	<b>ng</b>	jump to line <i>n</i>		<b>^o</b>	jump back	<small>jump forwards</small>	<b>^i</b>	jump forward			
	<b>zz</b>	center screen on cursor		<b>zt</b>	align top of screen with cursor		<b>zb</b>	align bottom of screen with cursor			
	<b>==</b>	auto-indent current line		<b>&lt;&lt;</b>	shift current line left by <i>shiftwidth</i>		<b>&gt;&gt;</b>	shift current line right by <i>shiftwidth</i>			

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 insert.txt
```

## COOL INSERT MODE STUFF

<b>^w</b>	delete word before cursor	<b>^u</b>	delete line before cursor
<b>^r</b> <b>r</b>	insert the contents of register <b>r</b>	<b>^R</b> =	use the expression register (try <b>^R=isd</b> )
<b>^t</b>	Increase line indent by shiftwidth	<b>^d</b>	decrease line indent by shiftwidth
<b>^x</b> <b>^l</b>	line completion	<b>^n</b>	find next completion suggestion according to complete

```
:h oneline.txt
```

## COMMAND-LINE MODE ONLY

edit using Normal mode <b>^f</b> cwinid	insert word under cursor <b>^R</b> <b>^w</b> cwinid-insert	completion suggestions <b>^d</b> cwinid-completion
---	--	--

Put `vimrc $X_CG-RS-extend '%<h%>/' /<CG>` in your `vimrc` so you can type `^f` 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/$(Script)/Design/           "Scribled" -> "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\Bungle" -> "Cantor" :h \_.


Special escapes can be used to change the case of substitutions.
:s:\(f\.\.\)\_W\I\E_            "foobar" -> "FOObar"     :h sub-replace-special

Use :global to perform a command on matching lines.
:g/fooban/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-v%

```



vim

vim keycodes

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

vim 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

d-motions  
et.com

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

background
bg
Set to "dark" if you have a dark color scheme

## 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 **"dd** is pronounced "register dd", and **"a**, "register a".

```
~: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**)

Use **:map** to view all current custom key mappings. Read **7 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!

"0"	Last disk 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 contents 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
qr	Record	Record into register <b>r</b> . Stop recording by hitting <b>@</b> again
@r	Playback	Execute the contents of register <b>r</b>
@@	Repeat last playback	Repeat the last <b>@r</b> , this is particularly useful with a count

vim one-liner used to sort the list of names by length:  
 cat /dev/urandom | tr -dc 'a-z' | fold -w 60 | xargs -n1 shuf -v  
 Heracitor Tiberius and others