

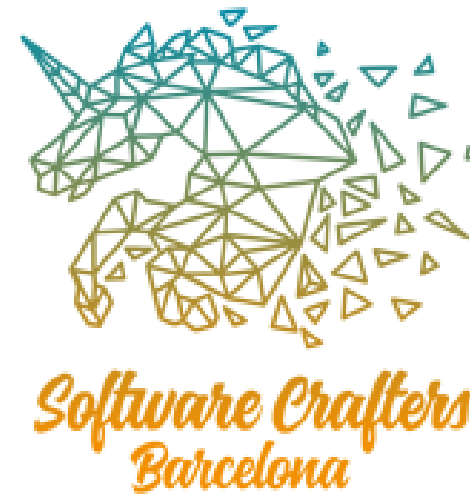
THE POWER OF VIM IN MODERN IDES

by Sergey Kudashev

ACKNOWLEDGMENTS



edpuzzle



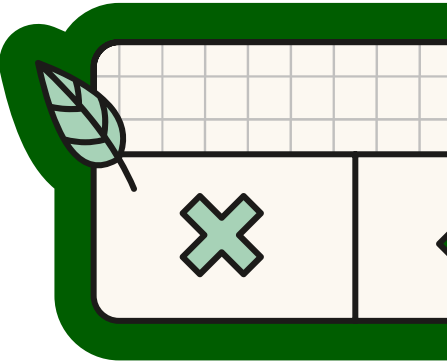


ABOUT ME

- a full-stack developer with a wide range of interests
- passionate about craft
- a constant learner
- an open-source contributor

 @kudashevs

 www.kudashevs.com





AGENDA

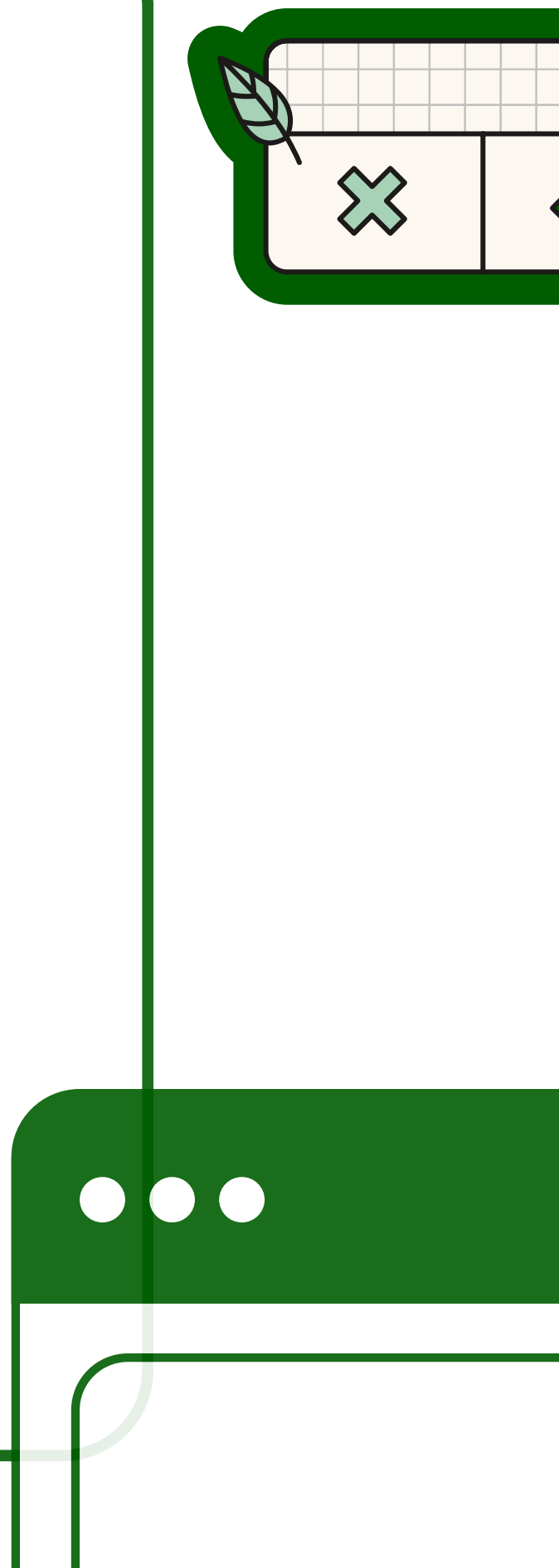
- Vim philosophy
 - Vim language
 - Commands
 - Motions
 - Objects
 - Ex-commands
 - Search
 - Marks
- 
- 
- 

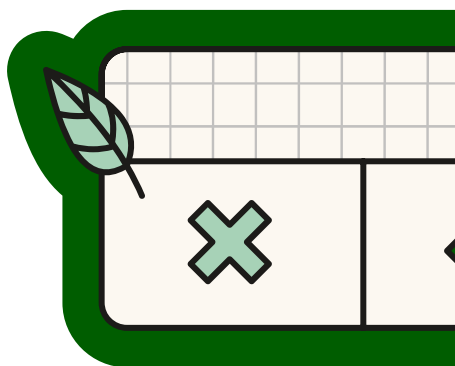


DISCLAIMER

I don't want to convince anyone that Vim is the best tool ever.

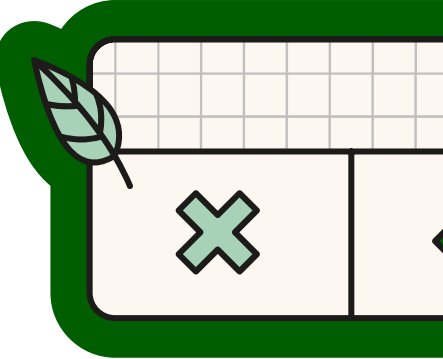

My goal is to show a way how the things can be done, to show the specific approach that Vim provides and spark your interest to the tool. If you adopt even one of the ideas that I will show next, I will be glad and delighted.







PHILOSOPHY

- treats editing text as the primary action (usually inserting is the primary action)
 - less keystrokes as possible
 - mouseless and arrowless development
 - intuitive operators and commands
 - composition of operators/motions/objects
 - always a way to repeat something
 - always a way to undo something
 - always a better/shorter way
- 
- 

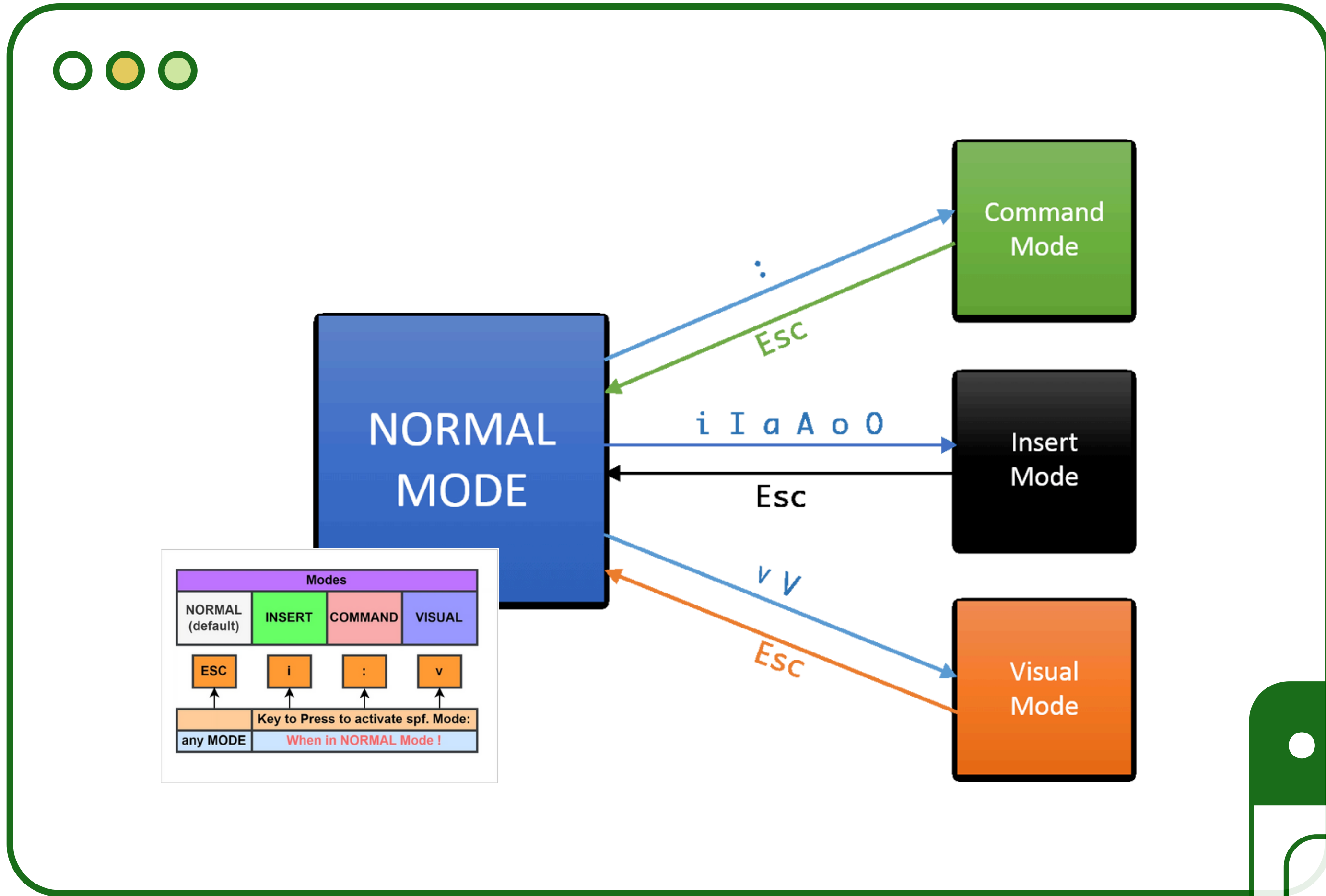


MODAL EDITING

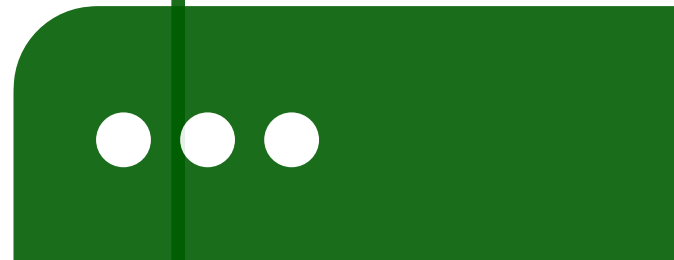
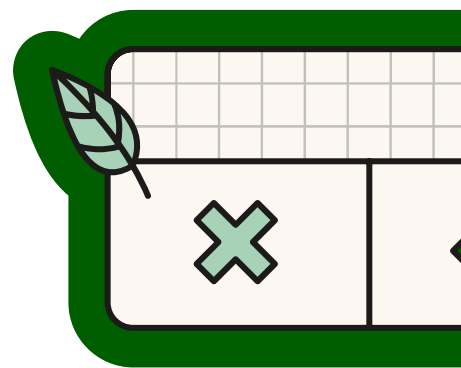
The main modes are:

- normal-mode,
- insert-mode,
- visual-mode,
- command-line-mode

* keystrokes mean different things depending on what mode you are in.



Modes			
NORMAL (default)	INSERT	COMMAND	VISUAL
ESC	i	:	v
Key to Press to activate spf. Mode:			
any MODE	When in NORMAL Mode !		

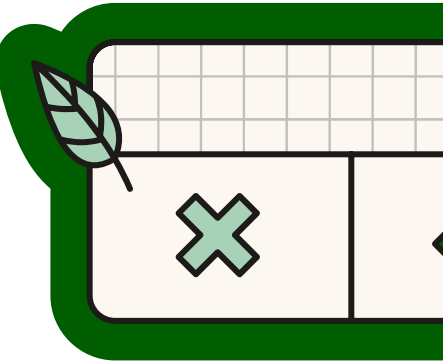


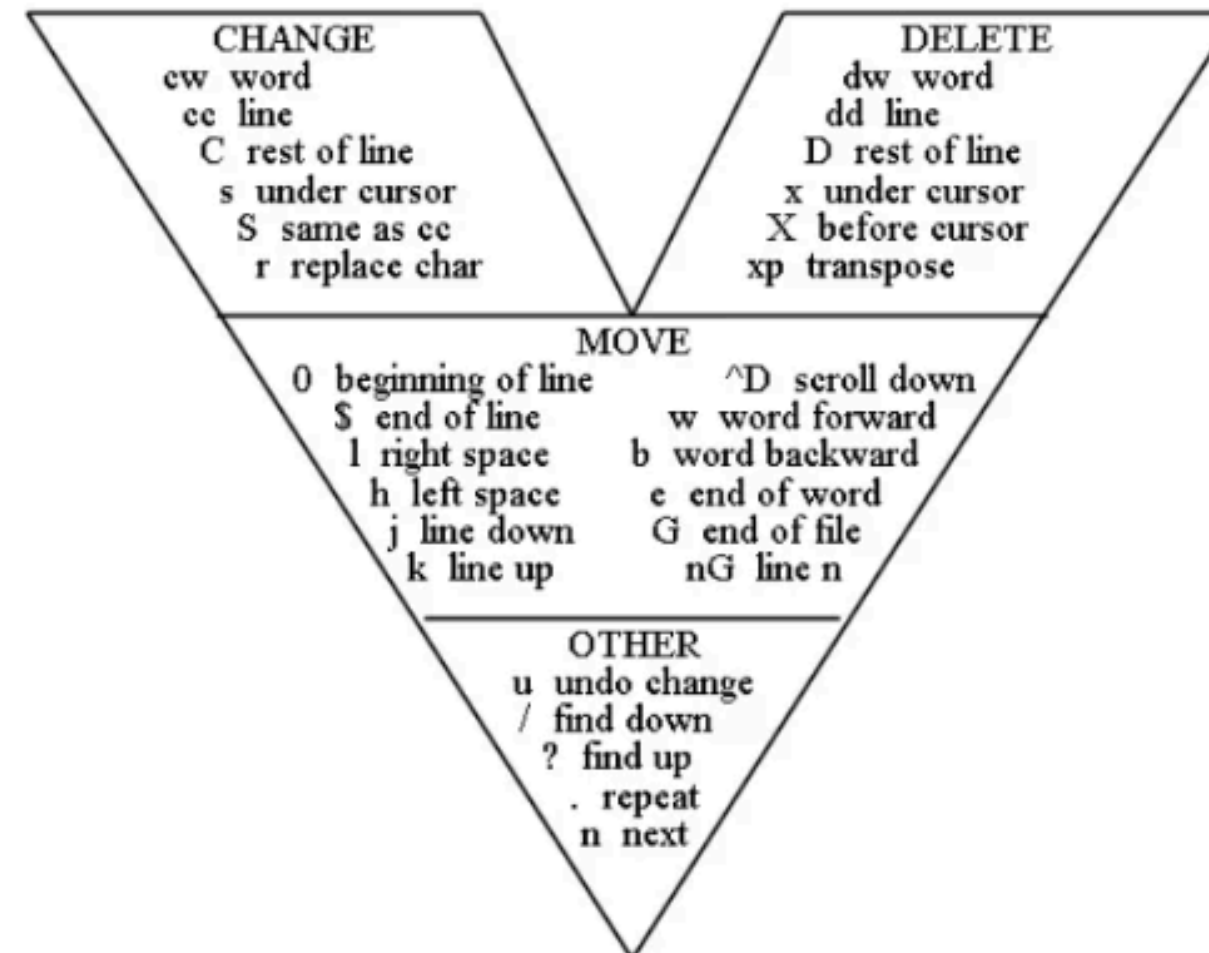
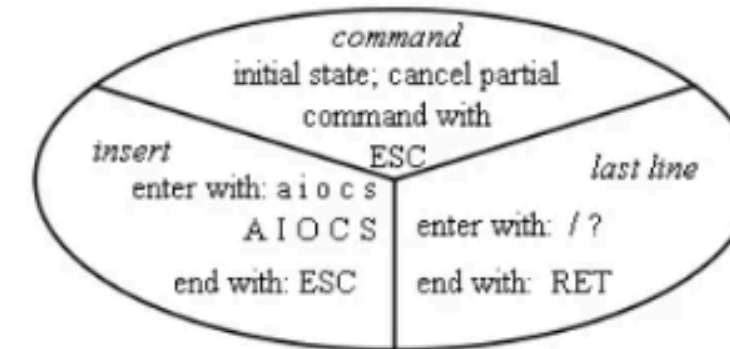


VIM LANGUAGE

The Vim language includes:

- operators/commands,
- motions,
- ex-commands,
- extra





INSERT a after cursor A at end of line i before cursor I at beginning of line o open line below O open line above yy yank line p put	
ex COMMANDS :se nu set numbers :se nonu no numbers :r file read in file :!cmd run command :se wm=10 wrap words	
SAVE/QUIT :w write buffer :q quit :wq write and quit :q! abandon buffer ZZ same as :wq ^Z suspend vi	



version 1.1
April 1st, 06

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	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 ¹	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-indent ³	> 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)

For a graphical vi/vim tutorial & more tips, go to www.viemu.com - home of ViEmu, vi/vim emulation for Microsoft Visual Studio



COMMANDS

u - undo
<C-r> - redo
U - restore line

Text entry commands:

a - append text following current cursor position
A - append text to the end of current line
i - insert text before the current cursor position
I - insert text at the beginning of the cursor line
o - open up a new line following the current line
O - open up a new line in front of the current line

* think of commands as verbs/operators





COMMANDS

p - paste below cursor

P - paste above cursor

c - change <something>

C - change to the end of the line

d - delete <something>

D - delete till the end of line

r - replace character

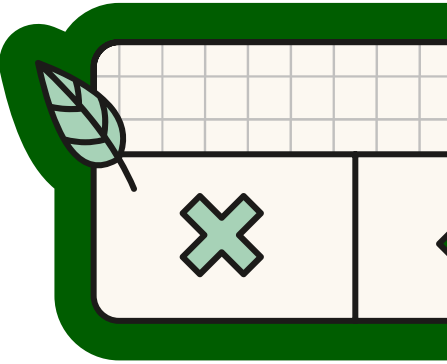
R - replace mode

s - substitute character

S - replace a whole line

x - delete character

X - delete previous character (backspace)





COMMANDS

y - yank <something>
Y - yank a whole line
> - indent right
< - indent left
= - auto-indent
v - select <something>
V - select a whole line (line mode)

J - join next line down to the end of the line



COMBINATIONS

s - is same as xi (the power of composition)

yy - copy a line

dd - delete a line

xp - swap two letters

. - repeat the last command

MOTIONS

The basic motions are:

- h - moves the cursor one character to the left
- j - moves the cursor down one line
- k - moves the cursor up one line
- l - moves the cursor one character to the right




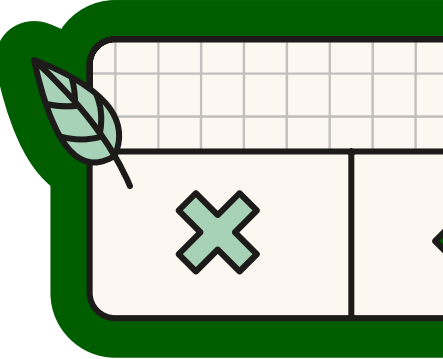
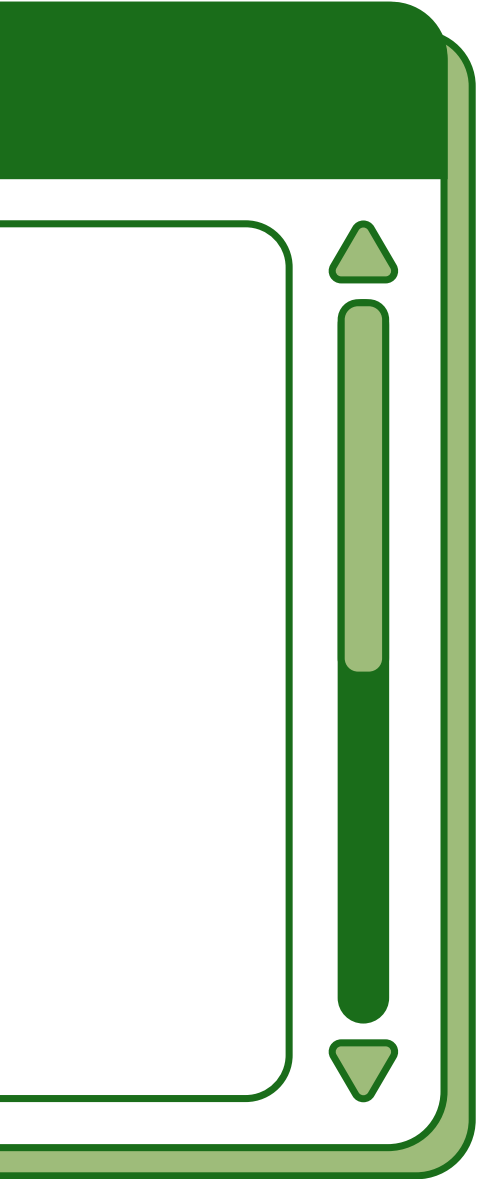
* but we don't want the basic motions only



MOTIONS (INSIDE FILE)

gg - move to the beginning of the file
G - move to the end of the file
<num>gg/G - move to a line number (ex-command also)

{ - jump to previous paragraph
} - jump to next paragraph
{[- jump to previous unmatched {
}] - jump to next unmatched }
[m - jump to previous method
]m - jump to next method
% - jump to corresponding parentheses (objects)



*car.ts

```
1 class Car
2 {
3     model: String;
4     doors: Number;
5     isElectric: Boolean;
6
7     constructor(model: String, doors: Number, isElectric: Boolean)
8     {
9         this.model = model;
10        this.doors = doors;
11        this.isElectric = isElectric;
12    }
13
14    make(): void
15    {
16        console.log(`This car is ${this.model} which has ${this.doors} doors` );
17    }
18 }
19
20 let newCar = new Car('Innova', 4, false);
21 newCar.make();
22
```

*cars.ts


```
1 class Car
2 {
3     model: String;
4     doors: Number;
5     isElectric: Boolean;
6
7     constructor(model: String, doors: Number, isElectric: Boolean)
8     {
9         this.model = model;
10        this.doors = doors;
11        this.isElectric = isElectric;
12    }
13
14    make(): void
15    {
16        console.log(`This car is ${this.model} which has ${this.doors} doors`);
17    }
18 }
19
20 let newCar = new Car('Innova', 4, false);
21 newCar.make();
22
```



MOTIONS (INSIDE LINE)

0 - start of a line
\$ - end of a line
^ - first non-blank character
g_ - last non-blank character (g is a magic key)

w - one word forward
W - one WORD forward
b - one word backward
B - one WORD backward
e - end of a current word
ge - end of a previous word (g is a magic key)

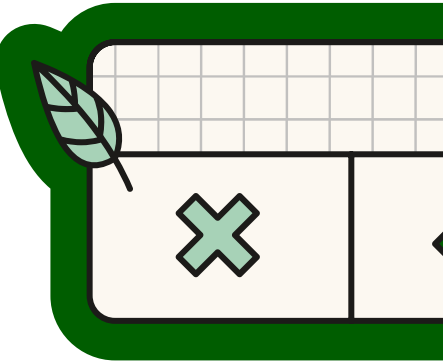




MOTIONS (INSIDE LINE)

f - go forward to <something>
F - go backward to <something>
t - go forward till <something>
T - go backward till <something>

; - repeat last f/t motion
, - repeat in a reverse order





OBJECTS

(or), [or], { or }, < or > - for matching pairs

", ', ` - for matching pairs of quotes

p - paragraph

s - sentence

b - block

t - tag

i - inner object

a - outer object

* but we don't want basics only





* not the magic, but the power of composition



COMPOSITION

`[count][operator][text object/motion]`

3dw - delete 3 words `[count][operator][motion]`

cit - change inside a tag `[operator][object]`

va{V - select a block `[operator][object][operator]`

d2at - remove two tags `[operator][count][object]`





EX-COMMANDS

`:` for entering the world of ex-commands

`:h <smth>` - to get help

`:q` - exit a document

`:q!` - exit without saving



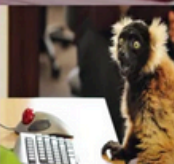


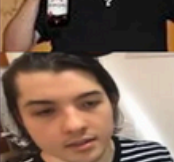

`:w` - write a document

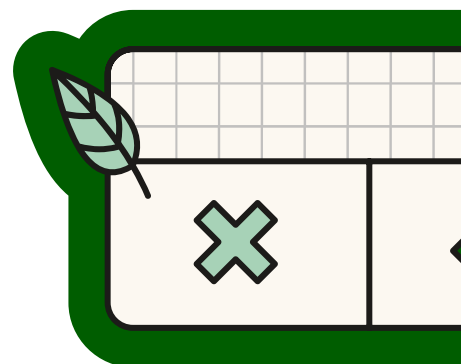
`:wq` - write and exit

`:h(elp) set`





hjkl	:q!	i			
p	esc	yy		w	b
v	:wq	u	C-R		
10j	ZZ	ZQ	nnoremap		daw
C-U	:split	C-D	dip		
Plugins plugins plugins	Leader key	"+	autocomplete		
C-I	C-O	block-editing			
:tabnew	q	C-X		C-A	
C-]	.? (..+?)\\1+	/v			
m	:normal				
buffers buffers buffers	:set undofile	:set lazyredraw			
wildmenu	Quickfix list				
Custom text-objects	Vim on the phone				
Maybe I should use Emacs??					


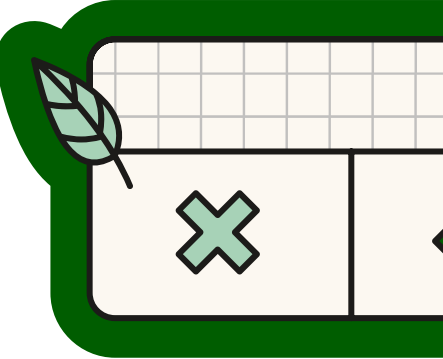




SEARCH

`/<smth>` - search for something forward
`?<smth>` - search for something backward
`n/N` - repeat last search `<times>`

`*` - search forward for the word under the cursor
`#` - search backward for the word under the cursor





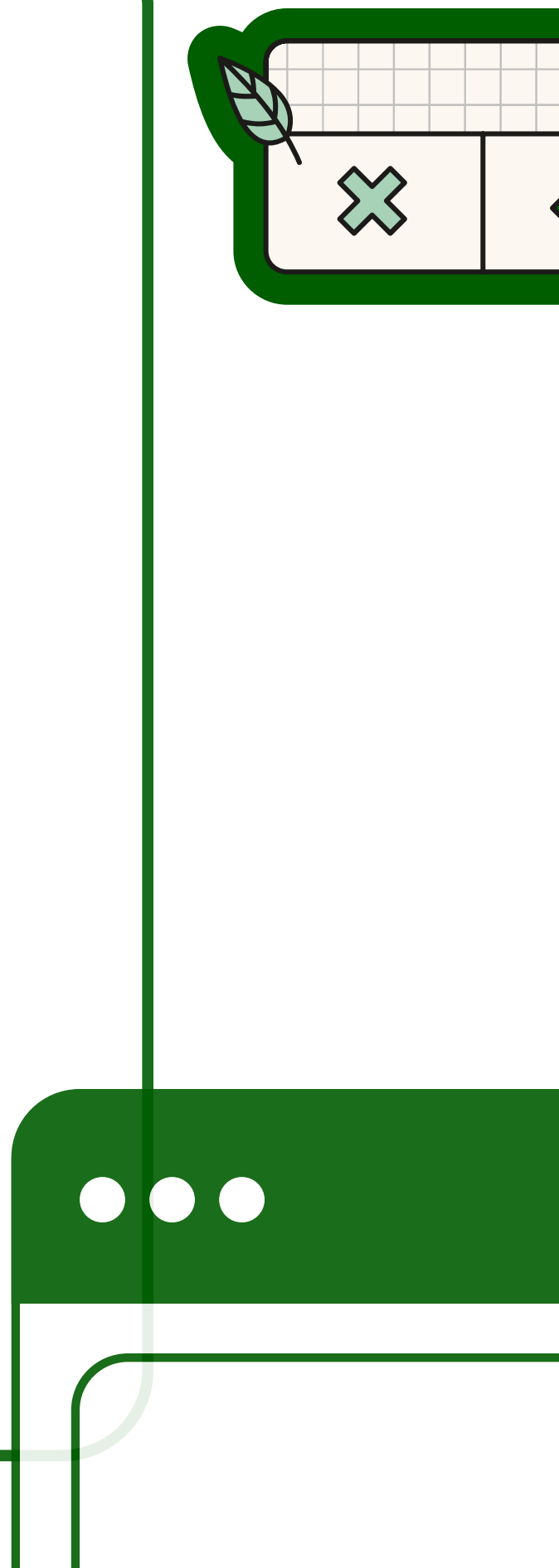
MARKS

:marks

m<char> - set mark
'<char> - go to the mark (line)
`<char> - go to the exact position
`.` - last change occurred in the current buffer

:jumplist

`" - last exited the current buffer
`0 - last file edited (previous edit point)
' - jump back to the line
`` - jump back to the position





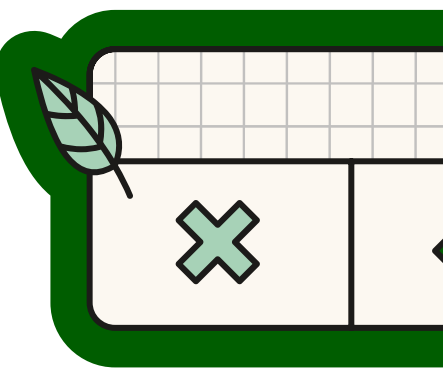
IN IDEAS



IdeaVim for IntelliJ products.



vscodevim for VSCode.





PLUGINS

`argtextobject` adds an argument text-object

`indent-object` adds an indent text-object

`surround` adds the possibility to wrap objects

`easymotion` simplifies some motions



NEXT STEPS

vimtutor - a program that teaches you the basics

Training sites:

<https://vim-adventures.com/>

<https://openvim.com/>

Documentation:

<https://vimhelp.org/>

<https://ideavim.sourceforge.net/vim/index.html>



CONCLUSION

Start with the vimtutor. Then, try to use Vim for some simple tasks. When you're comfortable with the presented ideas and the tool, try to integrate it into your workflow.

