vi/Vim normal-mode reference

Commands, motions, operators, text objects and counts

Modes and command structure

vi is modal. Normal mode interprets keys as editing commands; insert mode inserts text; command-line mode accepts ex commands beginning with a colon. Press Escape to return to normal mode. In Vim, visual mode selects text for operations; classic vi implementations may differ.

The most reusable normal-mode pattern is [count] operator [count] motion, such as 3dw, c$ or y}. Operators include d (delete), c (change) and y (yank). Many motions work both alone and as operands of an operator.

```
i  a  o  O  I  A
<Esc>
:w  :q  :wq  :q!
```

Movement and navigation

h, j, k and l move left, down, up and right. w and b move by words; e moves to the end of a word. 0 goes to column zero; ^ to the first nonblank; $ to line end. gg and G navigate to the start and end of a file in Vim; G is traditional vi, while gg is not guaranteed in every classic vi.

f{char} and t{char} search forward within the line; F and T search backward. ; repeats the last character find in the same direction, while , reverses it. /pattern searches forward and ?pattern backward; n repeats a search and N reverses it.

```
w b e W B E
0 ^ $
fX tX FX TX ; ,
/pattern  ?pattern  n  N
G  42G
```

Operators and text objects

d deletes text into a register; c deletes and enters insert mode; y copies without deleting. dd, cc and yy act on complete lines. In Vim, text objects such as iw (inner word), aw (a word), i" (inside quotes) and ip (inner paragraph) let an operator act on a semantic region rather than a directional movement.

Examples: ci" changes quoted content without removing quote marks; da(deletes a parenthesised expression including delimiters; yip yanks a paragraph. Text objects are primarily Vim extensions and should not be assumed on minimal historical vi implementations.

```
dw  cw  yw
dd  cc  yy
ci"  da(  yip
3dw  2c$
```

Single-key editing and repetition

x deletes the character under the cursor; X deletes the preceding character. r{char} replaces one character without entering insert mode. R enters replace mode in Vim and many vi implementations. p and P put yanked or deleted text after/before the cursor (or below/above for linewise content). u undoes a change; Ctrl-r redoes in Vim. The dot command repeats the last change.

J joins lines, ~ toggles case for a character in traditional usage, and >> or << shift indentation in Vim. Counts can apply to many of these operations, but their exact interpretation depends on the command.

```
x  X  rX  R
p  P  u  .  J
5x  3J
```

Registers, marks and ex commands

Named registers in Vim use the double-quote prefix, such as "ayy to yank a line into register a and "ap to put it. m{letter} sets a mark; '{letter} jumps to its line and `{letter} to its exact position. :set displays or changes options, :s performs substitution and :g applies commands to matching lines.

The precise feature set differs among vi, Vim, Neovim and small embedded implementations. For portable editing, rely on the POSIX vi command subset and consult the local help for extensions.

```
ma
'a
:1,20s/foo/bar/g
:set number
:help motion.txt
```

Reference scope: commands and defaults vary by distribution and software version. Consult the locally installed manual pages for exact syntax and behaviour.