Neovim Tips & Tricks

A collection of useful tips and tricks for Neovim

Sasa Markovic
smalltux@yahoo.com

November 3, 2025

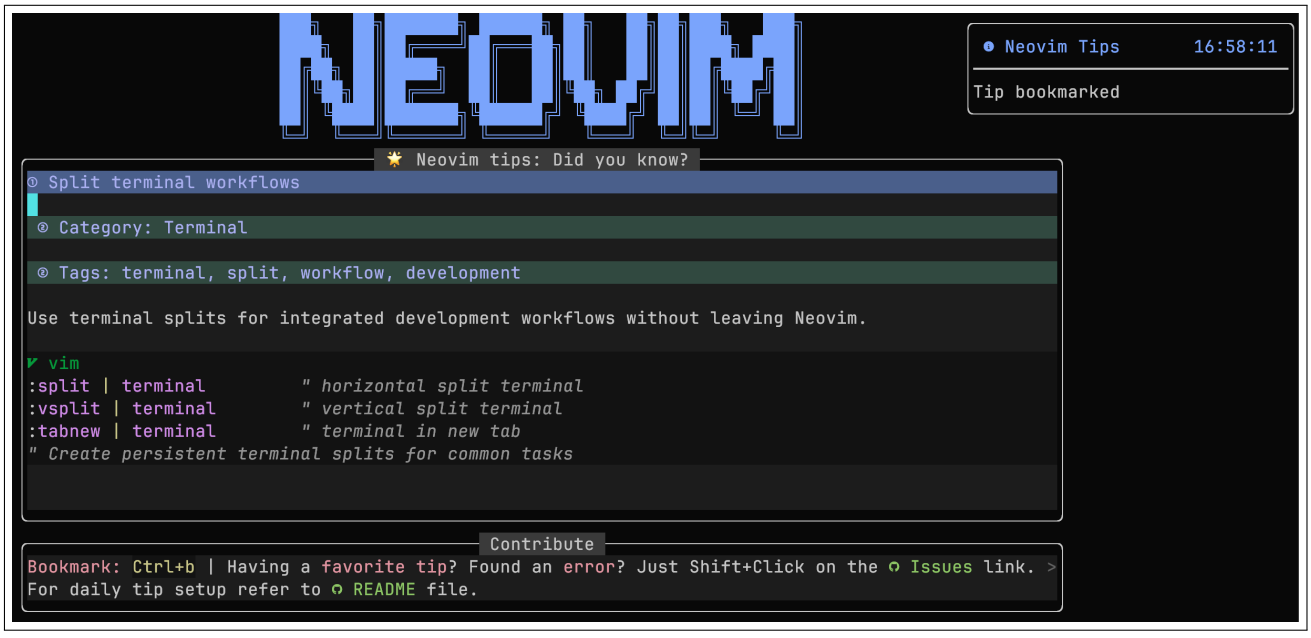
To my son Luka,
who showed me the joy of Neovim.

Table of Contents

Introduction

"I've been using Vim for about 2 years now,
mostly because I can't figure out how to exit it."
I Am Developer

This book is a printed version of my Neovim Tips plugin that can be found on Github at saxon1964/neovim-tips.



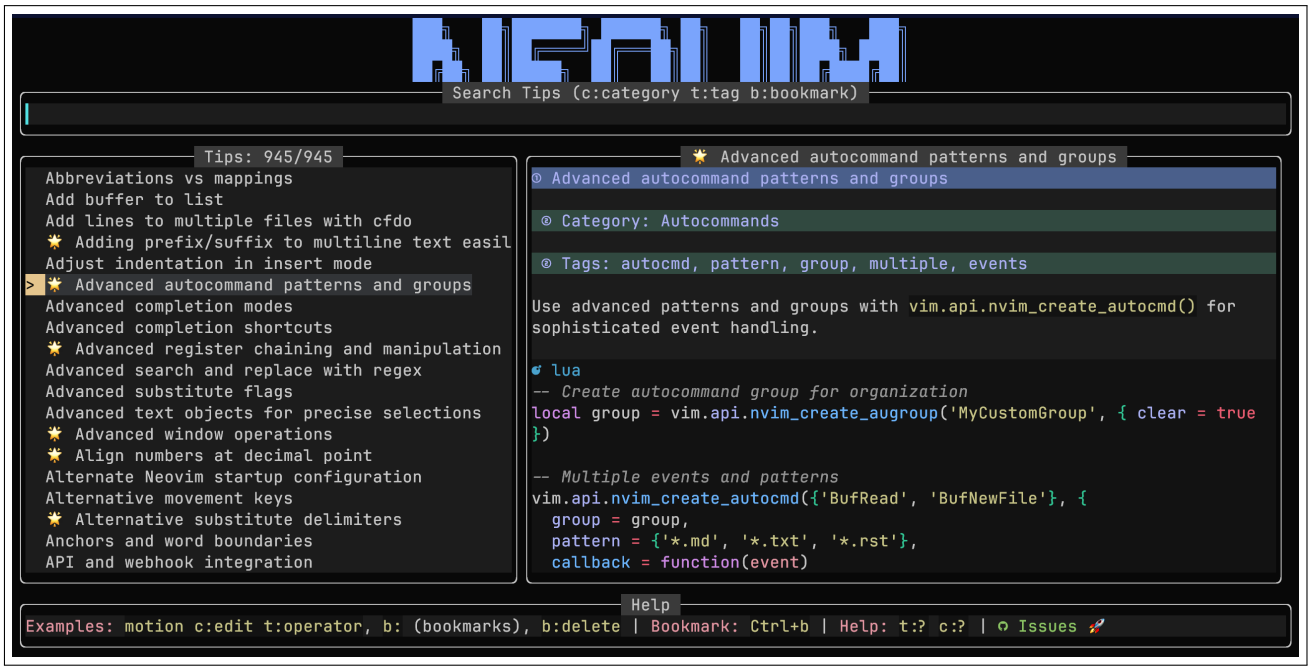
Daily tip from Neovim Tips plugin

This Lua plugin for Neovim brings together hundreds of helpful tips, tricks, and shortcuts, all available through a custom picker. It's easy to expand with your own entries, so the collection grows with you and your workflow.

I started to work on this little plugin because I love neovim and I still remember how difficult it was to learn the basic commands. This book, together with the plugin, should help you to learn some basic (:wq, write and quit) and some not so basic commands (ddp, move line down) related to Neovim.

I have provided a solid initial batch of tips and if you have your favorite one that is not listed, I will be happy to include it in the next release with proper credits. Send your commands, tips and tricks to me, create an issue or submit a pull request. Usign the plugin, you

can also add your own tips and tricks that will be stored on your local computer, you don't have to share anything with me. A few plugin screenshots can be found on the following page.



Neovim Tips plugin screenshot

CHAPTER 1

Advanced mappings

1.1 Abbreviations vs mappings

Category: Key Mappings
Tags: abbreviation, iabbrev, expand, text

Use abbreviations for text expansion that only triggers after whitespace, unlike mappings which are immediate.

Example

```
:iabbrev teh the
:iabbrev @@ your.email@domain.com
:iabbrev dts <C-r⩾strftime('%Y-%m-%d')<CR>
" Abbreviations expand after whitespace/punctuation
" Mappings activate immediately when typed
```

1.2 Auto-pair mappings

Category: Key Mappings
Tags: autopair, brackets, quotes, matching

Create smart bracket and quote auto-pairing with conditional mappings.

Example

```
:inoremap <expr> ( getline('.')[col('.')-2] =~ '\w' ? '(' : '()<Left>'
:inoremap <expr> { getline('.')[col('.')-2] =~ '\w' ? '{' : '{}<Left>'
:inoremap <expr> [ '[]<Left>'
:inoremap <expr> " '""<Left>'
" Smart auto-pairing that considers context
```

1.3 Buffer-local and mode-specific mappings

Category: Key Mappings
Tags: buffer, local, mode, specific

Use `<buffer>` for buffer-local mappings and different mode prefixes for mode-specific

key bindings.

Example

```
:nnoremap <buffer> <F5> :!python %<CR>
:vnoremap <leader>s :sort<CR>
:inoremap <C-l> <Right>
:cnoremap <C-a> <Home>
" Buffer-local mappings only affect current buffer
```

1.4 Buffer-local keymaps with Lua

Category: Key Mappings
Tags: lua, buffer, local, keymap, nvim_buf_set_keymap

Create buffer-specific keymaps in Lua that only apply to the current buffer without affecting global mappings.

Example

```
-- Set keymap for current buffer (buffer 0):
vim.keymap.set('n', '<leader>r', ':!python %<CR>', {
  buffer = 0,
  desc = 'Run current file'
})

-- In autocommand for specific filetype:
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'lua',
  callback = function(args)
    vim.keymap.set('n', '<leader>x', ':source %<CR>', {
      buffer = args.buf,
      desc = 'Source Lua file'
    })
  end,
})

-- In LspAttach autocommand:
vim.api.nvim_create_autocmd('LspAttach', {
  callback = function(args)
    local opts = { buffer = args.buf }
    vim.keymap.set('n', 'gd', vim.lsp.buf.definition, opts)
    vim.keymap.set('n', 'K', vim.lsp.buf.hover, opts)
    vim.keymap.set('n', '<leader>rn', vim.lsp.buf.rename, opts)
  end,
})
```

1.5 Command-line mappings

Category: Key Mappings
Tags: cnoremap, command, line, navigation

Use command-line mode mappings to improve command-line editing with familiar key bindings.

Example

```
:cnoremap <C-a> <Home>
:cnoremap <C-e> <End>
:cnoremap <C-b> <Left>
:cnoremap <C-f> <Right>
:cnoremap <C-d> <Delete>
" Emacs-style command line navigation
```

1.6 Conditional mappings

Category: Key Mappings
Tags: conditional, exists, hasmapto, check

Use `exists()` and `hasmapto()` to create conditional mappings that don't override existing ones.

Example

```
if !hasmapto(':make<CR>')
    nnoremap <F5> :make<CR>
endif
if exists(':Gdiff')
    nnoremap <leader>gd :Gdiff<CR>
endif
" Only create mapping if it doesn't exist or command is available
```

1.7 Context-aware mappings

Category: Key Mappings
Tags: context, aware, conditional, filetype

Create mappings that behave differently based on file type, mode, or cursor context.

Example

```
:autocmd FileType python nnoremap <buffer> <F5> :!python %<CR>
:autocmd FileType javascript nnoremap <buffer> <F5> :!node %<CR>
:autocmd FileType sh nnoremap <buffer> <F5> :!bash %<CR>
" Same key, different behavior per file type
```

1.8 Escape key alternatives

Category: Key Mappings
Tags: escape, alternative, jk, kj

Map common key combinations to escape key for faster mode switching without reaching for Esc.

Example

```
:inoremap jk <Esc>
:inoremap kj <Esc>
:inoremap jj <Esc>
:vnoremap v <Esc>
" Popular alternatives: jk, kj, jj, or double-tap current mode key
```

1.9 Expression mappings

Category: Key Mappings
Tags: expr, expression, mapping, dynamic

Use `<expr>` mappings to create dynamic key behaviors that evaluate expressions.

Example

```
:inoremap <expr> <Tab> pumvisible() ? "\<C-n>" : "\<Tab>"
:inoremap <expr> <CR> pumvisible() ? "\<C-y>" : "\<CR>"
:nnoremap <expr> n 'Nn'[v:searchforward]
" Tab for completion navigation, Enter to accept
```

1.10 Leader key mappings

Category: Key Mappings
Tags: leader, mapleader, prefix, namespace

Use `mapleader` to create a personal namespace for custom mappings, avoiding conflicts with default keys.

Example

```
:let mapleader = " "           " space as leader
:nnoremap <leader>f :find<Space>
:nnoremap <leader>b :buffer<Space>
:nnoremap <leader>w :write<CR>
" Creates ,f ,b ,w mappings (if comma is leader)
```

1.11 Mapping special characters

Category: Key Mappings
Tags: special, characters, escape, literal

Use proper escaping and notation for mapping special characters like quotes, backslashes, and pipes.

Example

```
:nnoremap <leader>" ciw"<C-r>""<Esc>
:nnoremap <leader>' ciw'<C-r>"'<Esc>
:nnoremap <leader>\ :nohlsearch<CR>
" Surround word with quotes, backslash to clear search
```

1.12 Mapping timeouts

Category: Key Mappings
Tags: timeout, ttimeout, delay, response

Use timeout settings to control how long vim waits for key sequence completion in mappings.

Example

```
:set timeoutlen=500    " wait 500ms for mapped sequence
:set ttimeoutlen=50    " wait 50ms for key code sequence
" Affects leader key combinations and escape sequences
" Lower ttimeoutlen for faster escape in terminal
```

1.13 Mapping with arguments

Category: Key Mappings
Tags: arguments, parameters, count, range

Use <count> and ranges in mappings to create flexible key bindings that accept numeric arguments.

Example

```
:nnoremap <silent> <leader>d :<C-u>call DeleteLines(v:count1)<CR>
function! DeleteLines(count)
    execute 'normal! ' . a:count . 'dd'
endfunction
" 3<leader>d deletes 3 lines
```

1.14 Multiple key mappings

Category: Key Mappings
Tags: multiple, keys, sequence, chain

Create mappings that respond to multiple key sequences or provide alternative bindings.

Example

```
:nnoremap <leader>fs :w<CR>
:nnoremap <leader>ff :find<Space>
:nnoremap <leader>fb :buffer<Space>
:nnoremap <C-s> :w<CR>
:inoremap <C-s> <Esc>:w<CR>a
" Multiple ways to save: <leader>fs and <C-s>
```

1.15 Operator-pending mappings

Category: Key Mappings
Tags: onoremap, operator, pending, motion

Use operator-pending mappings to create custom text objects and motions.

Example

```
:onoremap in( :<C-u>normal! f(vi(<CR>
:onoremap an( :<C-u>normal! f(va(<CR>
:onoremap in{ :<C-u>normal! f{vi{<CR>
" Creates 'in(' and 'an(' text objects
" Now you can use din( to delete inside next parentheses
```

1.16 Plug mappings

Category: Key Mappings
Tags: plug, scriptname, unique, naming

Use `<Plug>` prefix to create unique mapping names that users can map to their preferred keys.

Example

```
:nnoremap <Plug>MyPluginFunction :call MyFunction()<CR>
:nmap <F5> <Plug>MyPluginFunction
" Plugin provides <Plug> mapping, user maps it to preferred key
" Prevents conflicts and allows customization
```

1.17 Recursive abbreviations

Category: Key Mappings
Tags: abbreviation, recursive, noreabbrev, expand

Use `noreabbrev` to prevent recursive abbreviation expansion, similar to noremap for mappings.

Example

```
:abbreviate W w
:noreabbrev Wq wq
:abbreviate Q q
" 'W' expands to 'w', but 'Wq' won't recursively expand the 'W' part
```

1.18 Script-local mappings

Category: Key Mappings
Tags: script, local, SID, unique

Use `<SID>` (Script ID) to create mappings that call script-local functions, avoiding global namespace pollution.

Example

```
:nnoremap <silent> <F5> :call <SID>CompileAndRun()<CR>
function! s:CompileAndRun()
    " Script-local function
    execute '!gcc % -o %:r && ./%:r'
endfunction
" <SID> ensures function is only accessible from this script
```

1.19 Silent and no-remap mappings

Category: Key Mappings
Tags: noremap, silent, mapping, recursive

Use `noremap` and `<silent>` modifiers to create safe, non-recursive mappings that don't echo commands.

Example

```
:nnoremap <silent> <leader>w :w<CR>
:inoremap jk <Esc>
" noremap prevents recursive mapping, silent suppresses command echo
" Use noremap by default to avoid unexpected behavior
```

1.20 Special key notation

Category: Key Mappings
Tags: special, keys, notation, modifiers

Use special key notation like `<C-key>`, `<M-key>`, `<S-key>` for modifier combinations and special keys.

Example

```
:nnoremap <C-j> <C-w>j        " Ctrl+j to move down
:nnoremap <M-h> :tabprev<CR>  " Alt+h for previous tab
:nnoremap <S-Tab> :bprev<CR>  " Shift+Tab for previous buffer
:nnoremap <F12> :set invnumber<CR>  " F12 to toggle line numbers
```

1.21 Terminal mode mappings

Category: Key Mappings
Tags: tnoremap, terminal, mode, escape

Use terminal mode mappings to control built-in terminal behavior and key bindings.

Example

```
:tnoremap <Esc> <C-\><C-n>
:tnoremap <C-w>h <C-\><C-n><C-w>h
:tnoremap <C-w>j <C-\><C-n><C-w>j
:tnoremap <C-w>k <C-\><C-n><C-w>k
:tnoremap <C-w>l <C-\><C-n><C-w>l
" Escape to exit terminal mode, window navigation
```

1.22 Visual mode mappings

Category: Key Mappings
Tags: visual, vnoremap, selection, range

Use visual mode mappings to operate on selections with custom key combinations.

Example

```
:vnoremap <leader>s :sort<CR>
:vnoremap <leader>u :!uniq<CR>
:vnoremap * y/\V<C-r>"<CR>
:vnoremap # y?\V<C-r>"<CR>
" Sort selection, remove duplicates, search for selection
```

CHAPTER 2

Advanced neovim

2.1 Buffer-local variables with vim.b

Category: Advanced Neovim
Tags: buffer, local, variables, vim.b

Use `vim.b` to access buffer-local variables from Lua, providing cleaner syntax than traditional vim variables.

Example

```
:lua vim.b.my_setting = 'value'
:lua print(vim.b.my_setting)
:lua vim.b[0].setting = 'buffer 0 specific'
" Cleaner than :let b:my_setting = 'value'
```

2.2 Command preview and substitution

Category: Advanced Neovim
Tags: command, preview, substitution, inccommand

Use `inccommand` for live preview of Ex commands, especially substitution with real-time feedback.

Example

```
:set inccommand=split     " preview in split window
:set inccommand=nosplit   " preview inline
:%s/old/new/g            " shows live preview while typing
" Preview works with :substitute, :global, :sort, etc.
```

2.3 Custom completion sources

Category: Advanced Neovim
Tags: completion, custom, source, omnifunc

Use `vim.lsp.omnifunc` and custom completion functions to create intelligent completion sources.

Example

```
function! MyCompletion(findstart, base)
  if a:findstart
    return col('.') - 1
  else
    return ['custom1', 'custom2', 'custom3']
  endif
endfunction
:set omnifunc=MyCompletion
```

2.4 Deep inspection with vim.inspect

Category: Advanced Neovim
Tags: inspect, debug, pretty, print

Use `vim.inspect()` to pretty-print complex Lua data structures for debugging and development.

Example

```
:lua local data = {a = {b = {c = 'nested'}}, list = {1, 2, 3}}
:lua print(vim.inspect(data))
:lua print(vim.inspect(vim.bo, {depth = 1}))  " buffer options
:lua print(vim.inspect(vim.api, {depth = 1}))  " API structure
```

2.5 Event loop and scheduling

Category: Advanced Neovim
Tags: event, loop, schedule, async

Use `vim.schedule()` to defer function execution to the next event loop iteration for async operations.

Example

```
:lua vim.schedule(function()
  print('This runs in the next event loop')
  vim.cmd('echo "Deferred execution"')
end)
" Useful for async operations and avoiding blocking
```

2.6 Extmarks for persistent highlighting

Category: Advanced Neovim
Tags: extmarks, highlight, persistent, namespace

Use extmarks to create persistent, trackable highlights that survive buffer changes, unlike matchadd().

Example

```
:lua ns = vim.api.nvim_create_namespace('my_highlights')
:lua vim.api.nvim_buf_set_extmark(0, ns, 0, 0, {
  end_col=10, hl_group='Search', priority=100
})
:lua vim.api.nvim_buf_clear_namespace(0, ns, 0, -1)  " clear all
```

2.7 Filetype detection API

Category: Advanced Neovim
Tags: filetype, detection, api, lua

Use `vim.filetype.add()` to register custom filetype detection patterns and functions.

Example

```
:lua vim.filetype.add({
  extension = { log = 'log', conf = 'conf' },
  filename = { ['.eslintrc'] = 'json' },
  pattern = { ['.*%.env%..*'] = 'sh' }
})
```

2.8 Global variables with vim.g

Category: Advanced Neovim
Tags: global, variables, vim.g, configuration

Use `vim.g` to manage global variables from Lua, providing type-safe access to vim global variables.

Example

```
:lua vim.g.mapleader = ' '
:lua vim.g.loaded_netrw = 1        " disable netrw
:lua vim.g.python3_host_prog = '/usr/bin/python3'
" Equivalent to :let g:mapleader = ' '
```

2.9 Health check system

Category: Advanced Neovim
Tags: health, check, system, diagnostic

Use Neovim's health check system to create custom health checks for your configura-

tions and environments.

Example

```
:checkhealth            " run all health checks
:checkhealth vim.lsp    " check specific component
" Create ~/.config/nvim/lua/health/myconfig.lua
" with check() function for custom health checks
```

2.10 Highlight group API

Category: Advanced Neovim
Tags: highlight, api, colors, groups

Use `vim.api.nvim_set_hl()` to programmatically define and modify highlight groups from Lua.

Example

```
:lua vim.api.nvim_set_hl(0, 'MyHighlight', {
  fg = '#ff0000', bg = '#000000', bold = true
})
:lua local hl = vim.api.nvim_get_hl(0, {name = 'Comment'})
:lua print(vim.inspect(hl))
```

2.11 Keymap API with descriptions

Category: Advanced Neovim
Tags: keymap, api, description, which-key

Use `vim.keymap.set()` to create keymaps with descriptions and options, supporting which-key integration.

Example

```
:lua vim.keymap.set('n', '<leader>f', '<cmd>find<CR>', {
  desc = 'Find file', silent = true, buffer = 0
})
:lua vim.keymap.del('n', '<leader>f')  " delete keymap
```

2.12 Lua heredoc syntax

Category: Advanced Neovim
Tags: lua, heredoc, multiline, syntax

Use Lua heredoc syntax in vimscript for clean multiline Lua code blocks within vim configuration.

Example

```
lua << EOF
local function my_function()
  print("This is a multiline Lua function")
  vim.cmd('echo "Mixed Lua and Vim commands"')
end
my_function()
EOF
```

2.13 Lua require and module system

Category: Advanced Neovim
Tags: lua, require, module, package

Use Lua's require system to load and organize Neovim configuration modules with automatic caching and reloading.

Example

```
" Create ~/.config/nvim/lua/config/keymaps.lua
:lua require('config.keymaps')
:lua package.loaded['config.keymaps'] = nil  " force reload
:lua R = function(name) package.loaded[name] = nil; return require(name) end
```

2.14 Namespace management

Category: Advanced Neovim
Tags: namespace, management, api, isolation

Use namespaces to isolate highlights, extmarks, and diagnostics from different sources or plugins.

Example

```
:lua local ns1 = vim.api.nvim_create_namespace('source1')
:lua local ns2 = vim.api.nvim_create_namespace('source2')
:lua vim.api.nvim_buf_set_extmark(0, ns1, 0, 0, {hl_group = 'Search'})
:lua vim.api.nvim_buf_clear_namespace(0, ns1, 0, -1)  " clear ns1 only
```

2.15 Option management with vim.opt

Category: Advanced Neovim
Tags: options, vim.opt, configuration, lua

Use `vim.opt` for intuitive option management from Lua with proper data types and operations.

Example

```
:lua vim.opt.number = true
:lua vim.opt.tabstop = 4
:lua vim.opt.path:append('**')     " add to path
:lua vim.opt.wildignore:append('*.pyc')  " add to ignore list
```

2.16 RPC and job control (vim.system)

Category: Advanced Neovim
Tags: rpc, job, control, async

Use `vim.system()` for modern job control and `vim.rpcnotify()` for RPC communication with external processes.

Example

```
:lua local job = vim.system({'ls', '-la'}, {
  text = true,
  stdout = function(err, data) print(data) end
})
:lua job:wait()  " wait for completion
```

2.17 Ring buffer for undo history

Category: Advanced Neovim
Tags: undo, history, ring, buffer

Use Neovim's enhanced undo system with ring buffer capabilities for advanced undo tree navigation.

Example

```
:lua print(vim.fn.undotree())  " inspect undo tree
:earlier 1f  " go back 1 file write
:later 1f    " go forward 1 file write
:undolist    " show numbered undo states
```

2.18 Runtime path manipulation

Category: Advanced Neovim
Tags: runtime, path, rtp, manipulation

Use runtime path manipulation to dynamically load configurations and plugins at runtime.

Example

```
:lua vim.opt.rtp:prepend('~/my-custom-config')
:lua vim.opt.rtp:append('~/additional-plugins')
:lua for path in vim.gsplit(vim.o.rtp, ',') do print(path) end
" Runtime paths searched for configs and plugins
```

2.19 Secure mode and restrictions

Category: Advanced Neovim
Tags: secure, mode, restrictions, safety

Use secure mode and option restrictions to safely execute untrusted vim configurations and scripts.

Example

```
:set secure               " enable secure mode
:set exrc                 " allow local .vimrc files
:lua vim.o.secure = true  " Lua equivalent
" Restricts dangerous commands in local configs
```

2.20 Snippet expansion API

Category: Advanced Neovim
Tags: snippet, expansion, api, completion

Use `vim.snippet` API for snippet expansion and navigation without external snippet engines.

Example

```
:lua vim.snippet.expand('for var in iterable:\n\tpass')
:lua if vim.snippet.active() then vim.snippet.jump(1) end
" Built-in snippet support in Neovim 0.10+
```

2.21 Tab-local variables with vim.t

Category: Advanced Neovim
Tags: tab, local, variables, vim.t

Use `vim.t` to manage tab-local variables for tab-specific settings and state management.

Example

```
:lua vim.t.project_root = vim.fn.getcwd()
:lua vim.t[2].custom_title = 'Tab 2'  " specific tab
:lua print('Current tab project:', vim.t.project_root)
```

2.22 Treesitter API access

Category: Advanced Neovim
Tags: treesitter, api, ast, parsing

Use `vim.treesitter` API to query and manipulate the abstract syntax tree programmatically.

Example

```
:lua local parser = vim.treesitter.get_parser(0, 'lua')
:lua local tree = parser:parse()[1]
:lua local query = vim.treesitter.query.parse('lua', '(function_declaration)
↪  @func')
:lua for id, node in query:iter_captures(tree:root(), 0) do
↪  print(node:type()) end
```

2.23 UI events and hooks

Category: Advanced Neovim
Tags: ui, events, hooks, interface

Use UI event hooks to customize Neovim's behavior for different UI clients and frontends.

Example

```
:lua vim.api.nvim_set_option_value('guifont', 'Monospace:h12', {})
:lua if vim.g.neovide then vim.g.neovide_cursor_animation_length = 0.1 end
:lua print(vim.loop.os_uname().sysname)  " detect OS
```

2.24 User commands with Lua

Category: Advanced Neovim
Tags: user, command, lua, api

Use `vim.api.nvim_create_user_command()` to create custom commands with Lua functions and completion.

Example

```
:lua vim.api.nvim_create_user_command('Hello',
  function(opts) print('Hello ' .. opts.args) end,
  {nargs = 1, desc = 'Greet someone'}
)
:Hello World  " prints 'Hello World'
```

2.25 Virtual text annotations

Category: Advanced Neovim
Tags: virtual, text, annotations, inline

Use virtual text to display inline annotations like diagnostics, git blame, or documentation without modifying buffer content.

Example

```
:lua vim.api.nvim_buf_set_extmark(0, ns, vim.fn.line('.')-1, 0, {
  virt_text = {{'← This is a note', 'Comment'}},
  virt_text_pos = 'eol'
})
" Adds virtual text at end of current line
```

2.26 Window configuration API

Category: Advanced Neovim
Tags: window, configuration, api, layout

Use window configuration API for advanced window management and layout control.

Example

```
:lua vim.api.nvim_win_set_config(0, {
  relative = 'win', win = vim.api.nvim_get_current_win(),
  width = 50, height = 20, row = 5, col = 10
})
:lua local config = vim.api.nvim_win_get_config(0)
:lua print(vim.inspect(config))
```

2.27 Window-local variables with vim.w

Category: Advanced Neovim
Tags: window, local, variables, vim.w

Use `vim.w` to manage window-local variables from Lua for window-specific settings and state.

Example

```
:lua vim.w.quickfix_title = 'My Results'
:lua vim.w[1001].custom_setting = true  " specific window ID
:lua for winid, vars in pairs(vim.w) do print(winid, vim.inspect(vars)) end
```

CHAPTER 3
Advanced options

3.1 Automatic session restoration

Category: Configuration
Tags: sessionoptions, session, restore, automatic

Use `set sessionoptions` to control what gets saved in sessions, enabling automatic workspace restoration.

Example

```
:set sessionoptions=buffers,curdir,folds,help,tabpages,winsize,winpos
:mksession! ~/mysession.vim    " save session
:source ~/mysession.vim        " restore session
```

3.2 Automatic text wrapping

Category: Configuration
Tags: textwidth, wrap, formatoptions, auto

Use `set textwidth=80` with appropriate `formatoptions` to automatically wrap text at specified column width.

Example

```
:set textwidth=80
:set formatoptions+=t    " auto-wrap text using textwidth
:set formatoptions+=c    " auto-wrap comments
:set formatoptions+=r    " continue comments on new line
```

3.3 Backup and swap file locations

Category: Configuration
Tags: backupdir, directory, swap, backup

Use `set backupdir` and `set directory` to organize backup and swap files in dedicated directories.

Example

```
:set backupdir=~/.vim/backup//
:set directory=~/.vim/swap//
:set undodir=~/.vim/undo//
" // at end means use full path for unique filenames
```

3.4 Clipboard integration

Category: Configuration
Tags: clipboard, unnamed, system, copy

Use `set clipboard=unnamedplus` to automatically use system clipboard for yank and paste operations.

Example

```
:set clipboard=unnamedplus     " use system clipboard
:set clipboard=unnamed         " use * register (X11 primary)
:set clipboard=unnamed,unnamedplus  " use both
```

3.5 Complete options configuration

Category: Configuration
Tags: completeopt, completion, popup, menu

Use `set completeopt=menu,menuone,noselect,preview` to configure completion popup behavior and appearance.

Example

```
:set completeopt=menu,menuone,noselect,preview
" menu: show popup menu
" menuone: show menu even for single match
" noselect: don't auto-select first item
" preview: show extra info in preview window
```

3.6 Cursor line and column

Category: Configuration
Tags: cursorline, cursorcolumn, highlight, position

Use `set cursorline cursorcolumn` to highlight current cursor position with line and column indicators.

Example

```
:set cursorline       " highlight current line
:set cursorcolumn     " highlight current column
:set cursorline!      " toggle cursorline
```

3.7 Diff options configuration

Category: Configuration
Tags: diffopt, diff, comparison, algorithm

Use `set diffopt` to configure diff behavior, including algorithm choice and display options for better file comparison.

Example

```
:set diffopt=internal,filler,closeoff,hiddenoff,algorithm:patience
" internal: use internal diff engine
" filler: show filler lines
" algorithm:patience: use patience diff algorithm
```

3.8 Fold column display

Category: Configuration
Tags: foldcolumn, fold, display, gutter

Use `set foldcolumn=4` to display fold indicators in a dedicated column, making fold structure visible.

Example

```
:set foldcolumn=4     " show fold column with width 4
:set foldcolumn=0     " hide fold column
" Shows +/- indicators for folded code blocks
```

3.9 Incremental command preview

Category: Configuration
Tags: inccommand, preview, substitute, live

Use `set inccommand=split` to preview substitute commands in real-time with a split window showing changes.

Example

```
:set inccommand=split
" Now :%s/old/new/g shows live preview in split
:set inccommand=nosplit  " preview inline without split
```

3.10 Line break at word boundaries

Category: Configuration
Tags: linebreak, breakat, word, wrap

Use `set linebreak` with `set breakat` to wrap long lines at word boundaries rather than character boundaries.

Example

```
:set linebreak
:set breakat=\ \t!@*-+;:,./?   " break at these characters
:set showbreak=>>\              " show symbol at wrapped lines
```

3.11 Mouse support in terminal

Category: Configuration
Tags: mouse, terminal, scroll, select

Use `set mouse=a` to enable full mouse support in terminal Neovim for scrolling, selecting, and window operations.

Example

```
:set mouse=a          " enable mouse in all modes
:set mouse=n          " only in normal mode
:set mouse=           " disable mouse completely
```

3.12 Persistent undo across sessions

Category: Configuration
Tags: undofile, persistent, undo, history

Use `set undofile` to maintain undo history across vim sessions. Set `undodir` to control where undo files are stored.

Example

```
:set undofile
:set undodir=~/.vim/undodir
```

```
" Undo history persists even after closing files
```

3.13 Scroll context lines

Category: Configuration
Tags: scrolloff, sidescrolloff, context, buffer

Use `set scrolloff=8 sidescrolloff=8` to maintain context lines around cursor when scrolling vertically and horizontally.

Example

```
:set scrolloff=8        " keep 8 lines above/below cursor
:set sidescrolloff=8    " keep 8 columns left/right of cursor
:set scrolloff=999      " keep cursor centered (max context)
```

3.14 Search highlighting timeout

Category: Configuration
Tags: hlsearch, timeout, highlight, search

Use `set hlsearch` with timeouts to automatically clear search highlighting after inactivity.

Example

```
:set hlsearch
" Add to vimrc to clear highlighting after 5 seconds:
:autocmd CursorHold * set nohlsearch
:autocmd CmdlineEnter /,\? set hlsearch
```

3.15 Show invisible characters

Category: Configuration
Tags: listchars, invisible, whitespace, tabs

Use `set list listchars=tab:>\ ,eol:$,trail:.,space:.` to visualize invisible characters like tabs, spaces, and line endings.

Example

```
:set list
:set listchars=tab:>\ ,eol:$,trail:.,space:.
" Shows tabs as >, line endings as $, trailing spaces as .
```

3.16 Show line numbers relatively

Category: Configuration
Tags: relativenumber, number, navigation, jumping

Use `set relativenumber` with `set number` to show both absolute and relative line numbers for easier navigation.

Example

```
:set number relativenumber
" Shows current line number and relative distances
" Useful for commands like 5j, 3k
```

3.17 Smart case searching

Category: Configuration
Tags: ignorecase, smartcase, search, intelligent

Use `set ignorecase smartcase` for intelligent case handling - ignore case unless uppercase letters are typed.

Example

```
:set ignorecase smartcase
" /hello matches Hello, HELLO, hello
" /Hello only matches Hello, HELLO
```

3.18 Spell checking configuration

Category: Configuration
Tags: spell, spellfile, spelllang, dictionary

Use `set spell spelllang=en_us` to enable spell checking and configure custom word lists with `spellfile`.

Example

```
:set spell spelllang=en_us
:set spellfile=~/.config/nvim/spell/en.utf-8.add
" zg adds word under cursor to personal dictionary
" z= shows spelling suggestions
```

3.19 Virtual editing mode

Category: Configuration
Tags: virtualedit, cursor, beyond, eol

Use `set virtualedit=all` to allow cursor movement beyond end of lines, useful for block editing and column alignment.

Example

```
:set virtualedit=all    " cursor can go anywhere
:set virtualedit=block  " only in visual block mode
:set virtualedit=insert " only in insert mode
```

3.20 Wildmenu enhanced completion

Category: Configuration
Tags: wildmenu, completion, cmdline, enhanced

Use `set wildmenu` with `set wildmode=longest:full,full` for enhanced command-line completion with visual menu.

Example

```
:set wildmenu
:set wildmode=longest:full,full
" Now tab completion shows visual menu with options
```

CHAPTER 4

Advanced search patterns

4.1 Anchors and word boundaries

Category: Advanced Search
Tags: regex, anchor, boundary, word, line

Use ^ for line start, $ for line end, \< and \> for word boundaries.

Example

```
/^hello         " 'hello' at beginning of line
/hello$         " 'hello' at end of line
/\<word\>       " exact word 'word' with boundaries
/\<\u\w*\>      " word starting with uppercase letter
```

4.2 Atom and group matching

Category: Advanced Search
Tags: atom, group, capture, match

Use \(and \) for grouping and capturing, \1 to \9 for backreferences.

Example

```
/\(\w\+\)\s\+\1         " word repeated with whitespace
/\(.*\)\n\1             " duplicate lines
/\(<\w\+>\).\{-}\1      " XML/HTML tag pairs
:%s/\(\w\+\) \(\w\+\)/\2, \1/g  " swap first and last name
```

4.3 Branch and alternation

Category: Advanced Search
Tags: branch, alternation, or, choice

Use \| for alternation (OR), \%(... \) for grouping without capturing.

Example

```
/hello\|world       " match 'hello' OR 'world'
/\(foo\|bar\)baz    " match 'foobaz' or 'barbaz'
/\%(red\|blue\)     " non-capturing group for 'red' or 'blue'
```

4.4 Case sensitivity control

Category: Advanced Search
Tags: case, sensitive, insensitive, ignore, match

Use `\c` for case insensitive, `\C` for case sensitive, `\%#=1` for old regex engine.

Example

```
/hello\c        " case insensitive search
/Hello\C        " case sensitive search
/\c\<WORD\>     " case insensitive word boundary
/\%#=1pattern   " use old regex engine (sometimes faster)
```

4.5 Change ”Last, First” to ”First Last”

Category: Advanced Search
Tags: search, replace, regex

You have a list of names in this form:

Example

```
Doe, John
Smith, Alan
```

You want to change it to:

Example

```
John Doe
Alan Smith
```

This can be done with just one command:

Example

```
:%s/\([^,]*\), \(.*\)/\2 \1/
```

4.6 Character classes in search

Category: Advanced Search
Tags: regex, character, class, range, search

Use `[abc]` to match any of a, b, or c. Use `[a-z]` for ranges, `[^abc]` for negation.

Example

```
/[aeiou]        " match any vowel
/[0-9]          " match any digit
/[a-zA-Z]       " match any letter
/[^0-9]         " match any non-digit
/[[:alpha:]]    " match alphabetic characters
/[[:digit:]]    " match digits
```

4.7 Column and line position matching

Category: Advanced Search
Tags: position, column, line, range, specific

Use `\%23l` for line 23, `\%23c` for column 23, `\%>23l` for after line 23.

Example

```
/\%23lpattern   " pattern only on line 23
/\%>10l\%<20l   " pattern between lines 10 and 20
/\%5cword       " 'word' starting at column 5
/\%>50ctext     " 'text' after column 50
```

4.8 Composing complex patterns

Category: Advanced Search
Tags: complex, combine, pattern, advanced

Combine multiple regex features for sophisticated pattern matching.

Example

```
/\v^(\s*)(class|function)\s+\w+\s*\(
" Very magic pattern matching:
" - Line start with optional whitespace
" - 'class' or 'function' keyword
" - Whitespace and word (name)
" - Opening parenthesis

/\v<(https?|ftp)://[^\s]+>
" URL matching pattern
```

4.9 Lookahead and lookbehind patterns

Category: Advanced Search
Tags: regex, lookahead, lookbehind, assertion

Use `\@=` for positive lookahead, `\@!` for negative lookahead, `\@<=` for positive lookbehind, `\@<!` for negative lookbehind.

Example

```
/hello\@=world      " 'hello' followed by 'world'
/hello\@!           " 'hello' NOT followed by anything
/\@<=good morning   " 'morning' preceded by 'good'
/\@<!bad morning    " 'morning' NOT preceded by 'bad'
```

4.10 Mark position matching

Category: Advanced Search
Tags: mark, position, range, between

Use `\%'m` to match at mark m, `\%>'a` for after mark a, `\%<'b` for before mark b.

Example

```
/\%'apattern    " pattern at mark 'a' position
/\%>'a\%<'b     " between marks 'a' and 'b'
/\%>'<\%<'>     " within last visual selection
```

4.11 Multiline pattern matching

Category: Advanced Search
Tags: multiline, pattern, across, lines

Use `\_` prefix for character classes that include newlines.

Example

```
/function\_.\{-}end     " match function to end across lines
/\_^pattern             " pattern at start of any line
/\_$                    " end of any line
/\_s\+                  " one or more whitespace including newlines
/\_[a-z]                " any lowercase letter or newline
```

4.12 Non-greedy matching

Category: Advanced Search
Tags: regex, non-greedy, lazy, minimal

Use `{-}` for non-greedy version of `*`, `{-n,m}` for non-greedy quantified matching.

Example

```
/".*"           " greedy: matches entire "hello" "world"
/".{-}"         " non-greedy: matches "hello" and "world" separately
/a.\{-}b        " non-greedy: shortest match from 'a' to 'b'
```

4.13 Pattern modifiers and flags

Category: Advanced Search
Tags: modifier, flag, option, behavior

Use various flags to modify search behavior and pattern interpretation.

Example

```
/pattern/e      " position cursor at end of match
/pattern/s      " set search pattern but don't jump
/pattern/b      " search backward
/pattern/+2     " position cursor 2 lines after match
/pattern;/next  " search for pattern, then search for 'next'
```

4.14 Quantifiers in search patterns

Category: Advanced Search
Tags: regex, quantifier, repeat, match

Use `*` for zero or more, `+` for one or more, `?` for zero or one, `{n}` for exactly n.

Example

```
/ab*            " a followed by zero or more b's
/ab+            " a followed by one or more b's (very magic: /\vab+)
/ab?            " a followed by zero or one b (very magic: /\vab?)
/ab{3}          " a followed by exactly 3 b's (very magic: /\vab{3})
/ab{2,5}        " a followed by 2 to 5 b's (very magic: /\vab{2,5})
```

4.15 Recursive patterns

Category: Advanced Search
Tags: recursive, pattern, nested, structure

Use `\%(\)` and backreferences for matching nested structures.

Example

```
/([^()]*\([^()]*\)[^()]*)   " match balanced parentheses (simple)
/\v"([^"\\]|\\.)*"          " match quoted strings with escapes
```

4.16 Search and replace using custom lua functions

Category: Advanced Search
Tags: group, capture, search, replace, lua, function

You can define your own function directly in Neovim. It won't survive Neovim restart but such functions can be useful anyway. For example, the following function capitalizes the first letter in the text:

Example

```
:lua function capitalize(text) return text:sub(1,1):upper() .. text:sub(2)
↪  end
```

You can now use your own custom function to capitalize every word in the document using `%s/<pattern>/\=v:lua.your_fun(submatch(0))/`.

Example

```
:%s/\w\+/\=v:lua.capitalize(submatch(0))/gc
```

You can also apply your custom function to every line that is matching the given pattern using `g/<pattern>/s//\=v:lua.your_fun(submatch(0))`

Example

```
:g/\w\+/s//\=v:lua.capitalize(submatch(0))
```

Credits: Different-Ad-8707@Reddit

4.17 Search and replace with expressions

Category: Advanced Search
Tags: expression, function, dynamic, replace

Use `\=` in replacement to evaluate expressions dynamically.

Example

```
:%s/\d\+/\=submatch(0)*2/g      " double all numbers
:%s/$/\=' - line '.line('.')/   " add line numbers at end
:%s/\w\+/\=len(submatch(0))/g   " replace words with their length
```

4.18 Search context and ranges

Category: Advanced Search
Tags: context, range, scope, limit

Use ranges and context to limit search scope effectively.

Example

```
:+5,+10s/old/new/g      " replace from 5 to 10 lines below cursor
:.,/pattern/s/a/b/g     " replace from cursor to first pattern match
:/start/,/end/s/x/y/g   " replace between start and end patterns
```

4.19 Search history and repetition

Category: Advanced Search
Tags: history, repeat, search, previous

Use / then arrow keys to navigate search history, /<Up> to recall previous searches.

Example

```
/<Up>           " previous search in history
/<Down>         " next search in history
/<C-p>          " previous search (alternative)
/<C-n>          " next search (alternative)
//              " repeat last search
```

4.20 Search in specific file types

Category: Advanced Search
Tags: filetype, specific, extension, file

Combine search with file patterns for targeted searching.

Example

```
:vimgrep /pattern/ **/*.py      " search in Python files only
:grep -r "pattern" --include="*.js" .   " external grep in JS files
:lvimgrep /function/ *.lua      " local search in Lua files
```

4.21 Search with confirmation

Category: Advanced Search
Tags: confirm, interactive, replace, substitute

Use the `c` flag in substitute commands for interactive confirmation.

Example

```
:%s/old/new/gc  " global replace with confirmation
:g/pattern/s/old/new/c  " replace on matching lines with confirmation
```

Prompts: `yes`, `no`, `all`, `quit`, `last`, `^E` scroll down, `^Y` scroll up.

4.22 Special characters and escaping

Category: Advanced Search
Tags: regex, escape, special, character, literal

Use `\` to escape special characters. Common escapes: `\.` for literal dot, `\\` for backslash, `\*` for asterisk.

Example

```
/file\.txt      " literal dot in 'file.txt'
/C:\\path       " literal backslashes in path
/\$price        " literal dollar sign
/\[bracket\]    " literal square brackets
```

4.23 Very magic mode shortcuts

Category: Advanced Search
Tags: very-magic, shortcut, intuitive, regex

Use `\v` to make regex more like standard regex engines.

Example

```
/\v(word1|word2)+           " one or more of word1 or word2
/\v<\w+>@<email\.com        " word before email.com
/\v^\s*#\s*include          " C include statements
/\v(function|class)\s+\w+   " function or class definitions
```

4.24 Virtual column matching

Category: Advanced Search
Tags: virtual, column, tab, display, width

Use `\%23v` for virtual column 23 (accounts for tab display width).

Example

```
/\%8vpattern    " pattern at virtual column 8
/\%>20vtext     " text after virtual column 20
/\%<10v\S       " non-whitespace before virtual column 10
```

4.25 Zero-width assertions

Category: Advanced Search
Tags: zero-width, assertion, position, match

Use zero-width patterns to match positions without consuming characters.

Example

```
/\zs\w\+\ze@        " match word before @, highlight only word
/.*\zs\w\+$         " match last word on line
/^\zs\s\+           " match leading whitespace (for highlighting)
```

CHAPTER 5

Advanced text manipulation

5.1 Advanced register chaining and manipulation

Category: Registers
Tags: register, chain, manipulation, sequence, advanced

Chain register operations and use registers creatively for complex text manipulation workflows.

Example

```
" Chain multiple register operations
"ayiw"byiw"cp       " yank word to 'a', yank to 'b', paste 'c'
"Ayiw               " append to register 'a' (uppercase)

" Register arithmetic
:let @a = @a + 1    " increment number in register 'a'
:let @b = @a . @b   " concatenate registers

" Swap register contents
:let tmp = @a | let @a = @b | let @b = tmp

" Use registers in substitution
:%s/old/@a/g        " replace 'old' with register 'a' content
:%s/\(\w\+\)/\=@a/g " replace each word with register 'a'

" Complex register macros
qa                  " start recording macro 'a'
I"<Esc>A"<Esc>j     " wrap line in quotes, go to next
q                   " stop recording
@a                  " execute macro
@@                  " repeat last macro
5@a                 " execute macro 5 times
```

5.2 Advanced text objects for precise selections

Category: Text Objects
Tags: textobject, selection, precise, custom, advanced

Use advanced text object variations for more precise text selection and manipulation.

Example

```
" Next/Last variations
in(     " inside next (
il(     " inside last (
an)     " around next )
al)     " around last )

" Multi-line text objects
ap      " around paragraph
ip      " inside paragraph
as      " around sentence
is      " inside sentence

" Advanced combinations
va"i'   " select around " then inside '
ci"<Esc>va'   " change inside " then select around '

" Custom text object for function calls
vif     " inside function (with treesitter)
vaf     " around function (with treesitter)
vic     " inside class (with treesitter)
vac     " around class (with treesitter)
```

5.3 Expression register for calculations

Category: Registers
Tags: register, expression, calculation, math, formula

Use the expression register "= to perform calculations and dynamic text insertion.

Example

```
" In insert mode:
<C-r>=42*7<CR>          " inserts 294
<C-r>=strftime('%Y-%m-%d')<CR>  " inserts current date
<C-r>=line('.')<CR>     " inserts current line number
<C-r>=expand('%:t')<CR> " inserts filename

" In command mode:
:echo @=                " show expression register content
:let @= = '2+2'         " set expression register
<C-r>=2+2<CR>          " calculate and insert result

" Complex expressions:
<C-r>=printf('Line %d: %s', line('.'), getline('.'))<CR>
<C-r>=system('date +%s')<CR>  " unix timestamp
<C-r>=repeat('-', 50)<CR>     " insert 50 dashes
```

5.4 Zero-width assertions in search patterns

Category: Advanced Search
Tags: regex, assertion, lookahead, lookbehind, pattern

Use zero-width assertions (`\@=`, `\@!`, `\@≤`, `\@<!`) for complex search patterns that match without consuming characters.

Example

```
" Positive lookahead (\@=)
/foo\@=bar          " match 'foo' only if followed by 'bar'
/\w\+\@=ing         " match word ending with 'ing'

" Negative lookahead (\@!)
/foo\@!bar          " match 'foo' only if NOT followed by 'bar'
/^\w\+\@!\d         " match line starting with non-word then digit

" Positive lookbehind (\@≤)
/\@≤foo             " match 'foo' only if preceded by pattern
/\d\@≤px            " match 'px' only after digits

" Negative lookbehind (\@<!)
/\@<!foo            " match 'foo' only if NOT preceded by pattern
/\w\@<!-            " match '-' not preceded by word character

" Complex combinations
/\@≤\d\+\.\@=       " match digits between word and dot
/\(function\)\@≤\w\+\@=(   " function names
```

CHAPTER 6

Autocommands

6.1 Auto-backup important files

Category: Autocommands
Tags: autocmd, BufWritePre, backup, copy

Use `BufWritePre` to create timestamped backups of important configuration files before saving.

Example

```
:autocmd BufWritePre .vimrc,init.lua,init.vim
    \ execute 'write! ' . expand('%') . '.backup.' .
    ↪  strftime('%Y%m%d_%H%M%S')
" Creates timestamped backups of config files
```

6.2 Auto-chmod executable scripts

Category: Autocommands
Tags: autocmd, BufWritePost, chmod, executable

Use `BufWritePost` to automatically make shell scripts executable after saving them.

Example

```
:autocmd BufWritePost *.sh,*.py,*.pl,*.rb silent !chmod +x %
:autocmd BufWritePost *
    \ if getline(1) =~ "^#!" |
    \   silent !chmod +x % |
    \ endif
" Make files with shebang executable
```

6.3 Auto-close quickfix window

Category: Autocommands
Tags: autocmd, QuickFixCmdPost, quickfix, close

Use `QuickFixCmdPost` to automatically close quickfix window when it's empty or open it

when populated.

Example

```
:autocmd QuickFixCmdPost [^l]* copen
:autocmd QuickFixCmdPost l* lopen
" Auto-open quickfix/location list after commands
" Close if empty: :autocmd QuickFixCmdPost * if len(getqflist()) == 0 |
↪  cclose | endif
```

6.4 Auto-compile on save

Category: Autocommands
Tags: autocmd, BufWritePost, compile, build

Use `BufWritePost` to automatically compile or build files after saving them.

Example

```
:autocmd BufWritePost *.c,*.cpp !gcc % -o %:r
:autocmd BufWritePost *.tex !pdflatex %
:autocmd BufWritePost init.lua source %
" Compile C files, build LaTeX, reload Lua config
```

6.5 Auto-format code on save

Category: Autocommands
Tags: autocmd, BufWritePre, format, lsp

Use `BufWritePre` with LSP or external formatters to automatically format code before saving.

Example

```
:autocmd BufWritePre *.js,*.ts,*.jsx,*.tsx lua vim.lsp.buf.format()
:autocmd BufWritePre *.py !black %
:autocmd BufWritePre *.go !gofmt -w %
" Format different file types with appropriate tools
```

6.6 Auto-reload changed files

Category: Autocommands
Tags: autocmd, checktime, FileChangedShellPost, reload

Use `FileChangedShellPost` and `checktime` to automatically reload files changed by external programs.

Example

```
:set autoread
:autocmd FocusGained,BufEnter,CursorHold,CursorHoldI * checktime
:autocmd FileChangedShellPost * echohl WarningMsg | echo "File changed on
↪  disk. Buffer reloaded." | echohl None
```

6.7 Auto-resize windows on terminal resize

Category: Autocommands
Tags: autocmd, VimResized, windows, resize

Use `VimResized` autocommand to automatically redistribute window sizes when terminal is resized.

Example

```
:autocmd VimResized * wincmd =
" Equalizes window sizes when vim is resized
" Useful when terminal window size changes
```

6.8 Auto-save on focus lost

Category: Autocommands
Tags: autocmd, FocusLost, auto-save, backup

Use `FocusLost` autocommand to automatically save all buffers when vim loses focus.

Example

```
:autocmd FocusLost * :wa
" Auto-save all buffers when switching away from vim
```

6.9 Auto-toggle relative numbers

Category: Autocommands
Tags: autocmd, InsertEnter, InsertLeave, relativenumber

Use insert mode events to toggle relative line numbers, showing absolute numbers in insert mode.

Example

```
:autocmd InsertEnter * set norelativenumber
:autocmd InsertLeave * set relativenumber
" Absolute numbers in insert mode, relative in normal mode
```

6.10 Change directory to current file with autocommand

Category: Autocommands
Tags: autocmd, BufEnter, cd, directory

Use `BufEnter` to automatically change working directory to the current file's directory.

Example

```
:autocmd BufEnter * cd %:p:h
" Always work in current file's directory
" Alternative: use 'autochdir' option
:set autochdir  " same effect as above
```

6.11 Create directory on save

Category: Autocommands
Tags: autocmd, BufWritePre, mkdir, directory

Use `BufWritePre` to automatically create parent directories when saving files to new paths.

Example

```
:autocmd BufWritePre * call mkdir(expand('<afile>:p:h'), 'p')
" Creates parent directories if they don't exist
" 'p' creates intermediate directories like mkdir -p
```

6.12 Highlight long lines

Category: Autocommands
Tags: autocmd, ColorColumn, textwidth, highlight

Use autocommands to dynamically highlight long lines or set color column based on file type.

Example

```
:autocmd FileType python setlocal colorcolumn=88
:autocmd FileType javascript,typescript setlocal colorcolumn=100
:autocmd FileType gitcommit setlocal colorcolumn=72
" Set different line length limits per file type
```

6.13 Highlight yanked text

Category: Autocommands
Tags: autocmd, TextYankPost, highlight, yank

Use `TextYankPost` to briefly highlight yanked text, making copy operations more visible.

Example

```
:autocmd TextYankPost * silent! lua vim.highlight.on_yank()
" In vimscript:
:autocmd TextYankPost * silent! call matchadd('Search', @", 86400)
:autocmd TextYankPost * silent! call timer_start(150, {-> clearmatches()})
```

6.14 Jump to last cursor position

Category: Autocommands
Tags: autocmd, BufReadPost, cursor, position

Use `BufReadPost` to automatically jump to the last known cursor position when reopening files.

Example

```
:autocmd BufReadPost *
    \ if line("'\"") > 0 && line("'\"") ≤ line("$") |
    \   exe "normal! g`\"" |
    \ endif
" Jumps to last position if it exists and is valid
```

6.15 Lua autocommands with pattern matching

Category: Autocommands
Tags: lua, autocmd, pattern, nvim_create_autocmd, event

Create autocommands in Lua using `vim.api.nvim_create_autocmd` with powerful pattern matching and callback functions.

Example

```
-- Basic autocommand with single event:
vim.api.nvim_create_autocmd("BufWritePre", {
  pattern = "*.lua",
  callback = function()
    vim.lsp.buf.format()
  end,
  desc = "Format Lua files before saving"
})
```

```
-- Multiple events and patterns:
vim.api.nvim_create_autocmd({"BufNewFile", "BufRead"}, {
  pattern = {"*.md", "*.markdown"},
  callback = function()
    vim.opt_local.spell = true
    vim.opt_local.conceallevel = 2
  end,
})

-- Using augroup for organization:
local group = vim.api.nvim_create_augroup("MyAutocommands", { clear = true
↪  })
vim.api.nvim_create_autocmd("FileType", {
  group = group,
  pattern = "python",
  callback = function()
    vim.opt_local.tabstop = 4
    vim.opt_local.shiftwidth = 4
  end,
})

-- With buffer-specific autocommand:
vim.api.nvim_create_autocmd("LspAttach", {
  callback = function(args)
    vim.keymap.set('n', 'gd', vim.lsp.buf.definition, { buffer = args.buf })
  end,
})
```

6.16 Remove trailing whitespace on save

Category: Autocommands
Tags: autocmd, BufWritePre, whitespace, cleanup

Use `BufWritePre` autocommand to automatically remove trailing whitespace before saving files.

Example

```
:autocmd BufWritePre * :%s/\s\+$//e
" Remove trailing whitespace on all file saves
" 'e' flag prevents error if no matches found
```

6.17 Set file type based on content

Category: Autocommands
Tags: autocmd, BufRead, filetype, detection

Use `BufRead` autocommands to set file types based on file content or patterns not caught

by default detection.

Example

```
:autocmd BufRead,BufNewFile *.conf set filetype=conf
:autocmd BufRead,BufNewFile Jenkinsfile set filetype=groovy
:autocmd BufRead * if getline(1) =~ '^#!/usr/bin/env python' | set ft=python
↪  | endif
```

6.18 Set indent based on file type

Category: Autocommands
Tags: autocmd, FileType, indent, tabstop

Use `FileType` autocommands to set language-specific indentation and tab settings.

Example

```
:autocmd FileType python setlocal tabstop=4 shiftwidth=4 expandtab
:autocmd FileType javascript,json setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType go setlocal tabstop=4 shiftwidth=4 noexpandtab
```

6.19 Show cursor line only in active window

Category: Autocommands
Tags: autocmd, WinEnter, WinLeave, cursorline

Use `WinEnter` and `WinLeave` to show cursor line highlighting only in the active window.

Example

```
:autocmd WinEnter * set cursorline
:autocmd WinLeave * set nocursorline
" Cursor line only visible in focused window
```

6.20 Smart auto-save with update command

Category: Autocommands
Tags: autocmd, auto-save, update, silent, efficient

Use `silent! update` for efficient auto-save that only writes when buffer is modified and file has changed.

Example

```
vim.api.nvim_create_autocmd({ "BufLeave", "FocusLost" }, {
  pattern = "*",
  command = "silent! update",
  desc = "Auto-save on leave/lost focus",
})
```

6.21 Spell check for specific file types

Category: Autocommands
Tags: autocmd, FileType, spell, markdown

Use `FileType` autocommands to enable spell checking for text-based file types automatically.

Example

```
:autocmd FileType markdown,text,gitcommit set spell spelllang=en_us
:autocmd FileType help set nospell
" Enable spell check for text files, disable for help
```

6.22 Template insertion for new files

Category: Autocommands
Tags: autocmd, BufNewFile, template, skeleton

Use `BufNewFile` to automatically insert templates or skeleton code for new files.

Example

```
:autocmd BufNewFile *.html 0r ~/.vim/templates/html_template.html
:autocmd BufNewFile *.py 0r ~/.vim/templates/python_template.py
:autocmd BufNewFile *.sh 0put ='#!/bin/bash' | $put ='' | 1
```

CHAPTER 7

Buffer management

7.1 Buffer namespaces for isolated state

Category: Buffers
Tags: buffers, namespaces, isolation, extmarks

Use namespaces to manage buffer decorations and state in isolation.

Example

```
-- Create namespace for your plugin/feature
local ns_id = vim.api.nvim_create_namespace("my_plugin")

-- Namespace provides isolation for:
-- - Extmarks
-- - Virtual text
-- - Highlights
-- - Signs

-- Add decorations in namespace
local function decorate_buffer(bufnr)
  bufnr = bufnr or 0

  -- Clear previous decorations
  vim.api.nvim_buf_clear_namespace(bufnr, ns_id, 0, -1)

  -- Add new decorations
  vim.api.nvim_buf_set_extmark(bufnr, ns_id, 0, 0, {
    virt_text = {{"Plugin Loaded", "Comment"}},
    virt_text_pos = "eol",
  })
end

-- Multiple namespaces don't conflict
local ns_errors = vim.api.nvim_create_namespace("errors")
local ns_hints = vim.api.nvim_create_namespace("hints")

-- Clear specific namespace
vim.api.nvim_buf_clear_namespace(0, ns_errors, 0, -1)

-- List all extmarks in namespace
local marks = vim.api.nvim_buf_get_extmarks(
  0, ns_id, 0, -1, {details = true}
)
```

7.2 Buffer-local autocommands

Category: Buffers
Tags: buffers, autocommands, buffer-local, events

Create autocommands that only trigger for specific buffers.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Buffer-local autocommand
vim.api.nvim_create_autocmd("BufWritePre", {
  buffer = bufnr,
  callback = function()
    print("This buffer is about to be written")
    -- Custom pre-save logic
  end,
})

-- Multiple events for one buffer
vim.api.nvim_create_autocmd({"TextChanged", "TextChangedI"}, {
  buffer = bufnr,
  callback = function()
    -- Auto-save or validation logic
    print("Buffer content changed")
  end,
})

-- Cleanup on buffer delete
vim.api.nvim_create_autocmd("BufDelete", {
  buffer = bufnr,
  callback = function()
    print("Buffer deleted, cleaning up...")
    -- Cleanup resources
  end,
})
```

7.3 Buffer-local keymaps

Category: Buffers
Tags: buffers, keymaps, buffer-local, mappings

Create keymaps that only work in specific buffers.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Buffer-local keymap (modern API)
vim.keymap.set("n", "q", function()
  vim.api.nvim_buf_delete(bufnr, {force = true})
```

```
end, {
  buffer = bufnr,
  desc = "Close this buffer"
})

-- Multiple buffer-local keymaps
local keymaps = {
  {"n", "r", ":lua RefreshBuffer()<CR>", "Refresh"},
  {"n", "s", ":lua SaveBuffer()<CR>", "Save"},
  {"n", "<Esc>", ":close<CR>", "Close"},
}

for _, map in ipairs(keymaps) do
  vim.keymap.set(map[1], map[2], map[3], {
    buffer = bufnr,
    desc = map[4],
    silent = true,
    noremap = true,
  })
end
```

7.4 Buffer-local variables

Category: Buffers
Tags: buffers, variables, buffer-local, metadata

Store buffer-specific data using buffer-local variables.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Set buffer-local variables
vim.b[bufnr].custom_type = "tool_output"
vim.b[bufnr].created_at = os.time()
vim.b[bufnr].metadata = {
  owner = "plugin_name",
  version = "1.0",
}

-- Read buffer-local variables
local function get_buffer_info(buf)
  buf = buf or 0
  return {
    type = vim.b[buf].custom_type,
    created = vim.b[buf].created_at,
    metadata = vim.b[buf].metadata,
  }
end

-- Check if buffer has specific variable
if vim.b[bufnr].custom_type then
```

```
  print("This is a special buffer:", vim.b[bufnr].custom_type)
end
```

7.5 Build a simple notes buffer system

Category: Buffers
Tags: buffers, notes, scratch, practical

Create a complete notes system using scratch buffers.

Example

```
local M = {}
local notes_buffers = {}

function M.create_note(title)
  title = title or "Note"

  -- Create scratch buffer
  local bufnr = vim.api.nvim_create_buf(true, false)

  -- Set buffer name and options
  local filename = title:gsub("%s+", "_") .. ".md"
  vim.api.nvim_buf_set_name(bufnr, filename)
  vim.bo[bufnr].filetype = "markdown"
  vim.bo[bufnr].buftype = ""

  -- Add header
  vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {
    "# " .. title,
    "",
    "Created: " .. os.date("%Y-%m-%d %H:%M"),
    "",
    "---",
    "",
  })

  -- Track note
  notes_buffers[bufnr] = {
    title = title,
    created = os.time(),
  }

  -- Switch to note
  vim.api.nvim_set_current_buf(bufnr)

  -- Jump to end
  vim.cmd("normal! G")

  return bufnr
end
```

```
function M.list_notes()
  local notes = {}
  for bufnr, info in pairs(notes_buffers) do
    if vim.api.nvim_buf_is_valid(bufnr) then
      table.insert(notes, {
        bufnr = bufnr,
        title = info.title,
        lines = vim.api.nvim_buf_line_count(bufnr),
      })
    end
  end
  return notes
end

function M.delete_note(bufnr)
  bufnr = bufnr or vim.api.nvim_get_current_buf()

  if notes_buffers[bufnr] then
    vim.api.nvim_buf_delete(bufnr, {force = true})
    notes_buffers[bufnr] = nil
    print("Note deleted")
  end
end

-- Commands
vim.api.nvim_create_user_command("NoteNew", function(opts)
  M.create_note(opts.args)
end, {nargs = "?"})

vim.api.nvim_create_user_command("NoteList", function()
  local notes = M.list_notes()
  for _, note in ipairs(notes) do
    print(string.format("%d: %s (%d lines)",
      note.bufnr, note.title, note.lines))
  end
end, {})

return M
```

7.6 Control buffer hiding behavior

Category: Buffers
Tags: buffers, bufhidden, hide, delete

Use `bufhidden` to control what happens when a buffer is hidden from view.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Options for bufhidden:
vim.bo[bufnr].bufhidden = ""        -- Keep (default)
```

```
vim.bo[bufnr].bufhidden = "hide"    -- Hide but keep in memory
vim.bo[bufnr].bufhidden = "unload"  -- Unload from memory
vim.bo[bufnr].bufhidden = "delete"  -- Delete buffer
vim.bo[bufnr].bufhidden = "wipe"    -- Wipe buffer completely

-- Example: Auto-cleanup buffer
vim.bo[bufnr].bufhidden = "wipe"
vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {"Temporary content"})
```

7.7 Create custom buffer picker

Category: Buffers
Tags: buffers, picker, selection, ui

Build a simple buffer picker using scratch buffer and floating window.

Example

```
local function buffer_picker()
  -- Get all buffers
  local buffers = {}
  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
    if vim.api.nvim_buf_is_loaded(buf) and
       vim.bo[buf].buflisted then
      local name = vim.api.nvim_buf_get_name(buf)
      name = name ~= "" and vim.fn.fnamemodify(name, ":~:.") or "[No Name]"
      table.insert(buffers, {
        bufnr = buf,
        display = string.format("%d: %s", buf, name)
      })
    end
  end

  -- Create picker buffer
  local picker_buf = vim.api.nvim_create_buf(false, true)
  local lines = vim.tbl_map(function(b) return b.display end, buffers)
  vim.api.nvim_buf_set_lines(picker_buf, 0, -1, false, lines)

  -- Open in floating window
  local width = 60
  local height = math.min(#buffers + 2, 20)
  local win = vim.api.nvim_open_win(picker_buf, true, {
    relative = "editor",
    width = width,
    height = height,
    row = 5,
    col = 10,
    style = "minimal",
    border = "rounded",
    title = " Buffers ",
    title_pos = "center",
  })
```

```
  -- Select buffer on Enter
  vim.keymap.set("n", "<CR>", function()
    local line = vim.fn.line(".")
    local selected = buffers[line]
    if selected then
      vim.api.nvim_win_close(win, true)
      vim.api.nvim_set_current_buf(selected.bufnr)
    end
  end, {buffer = picker_buf})

  -- Close on Esc or q
  for _, key in ipairs({"<Esc>", "q"}) do
    vim.keymap.set("n", key, function()
      vim.api.nvim_win_close(win, true)
    end, {buffer = picker_buf})
  end

  vim.bo[picker_buf].bufhidden = "wipe"
end

vim.api.nvim_create_user_command("Buffers", buffer_picker, {})
```

7.8 Create floating window with scratch buffer

Category: Buffers
Tags: buffers, floating, window, scratch

Combine scratch buffers with floating windows for temporary UI elements.

Example

```
-- Create scratch buffer
local bufnr = vim.api.nvim_create_buf(false, true)

-- Add content
vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {
  "╔══════════════════════════╗",
  "║    Floating Scratch      ║",
  "║    Press 'q' to close    ║",
  "╚══════════════════════════╝",
})

-- Calculate centered position
local width = 30
local height = 4
local win_width = vim.api.nvim_get_option_value("columns", {})
local win_height = vim.api.nvim_get_option_value("lines", {})

local row = math.floor((win_height - height) / 2)
local col = math.floor((win_width - width) / 2)
```

```
-- Open floating window
local win_id = vim.api.nvim_open_win(bufnr, true, {
  relative = "editor",
  width = width,
  height = height,
  row = row,
  col = col,
  style = "minimal",
  border = "rounded",
})

-- Set buffer-local keymap to close
vim.api.nvim_buf_set_keymap(bufnr, "n", "q",
  ":close<CR>",
  {nowait = true, noremap = true, silent = true})

-- Auto-wipe buffer when window closes
vim.bo[bufnr].bufhidden = "wipe"
```

7.9 Create listed buffer

Category: Buffers
Tags: buffers, listed, create

Create a listed buffer that appears in the buffer list and behaves like a regular file buffer.

Example

```
-- Create listed, non-scratch buffer
local bufnr = vim.api.nvim_create_buf(true, false)

-- true = listed in buffer list
-- false = not a scratch buffer (will have swapfile, can be saved)

-- Set buffer name
vim.api.nvim_buf_set_name(bufnr, "MyBuffer")

-- Set content
vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {
  "This is a regular buffer",
  "You can save it with :w"
})

-- Switch to buffer
vim.api.nvim_set_current_buf(bufnr)
```

7.10 Create read-only buffer

Category: Buffers
Tags: buffers, readonly, modifiable, locked

Create read-only buffers for displaying information that shouldn't be edited.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Set buffer options for read-only
vim.bo[bufnr].modifiable = false
vim.bo[bufnr].readonly = true
vim.bo[bufnr].buftype = "nofile"

-- Add content (need to make modifiable temporarily)
vim.bo[bufnr].modifiable = true
vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {
  "≡≡≡ Read-Only Content ≡≡≡",
  "This buffer cannot be edited",
})
vim.bo[bufnr].modifiable = false

-- Open in window
vim.api.nvim_win_set_buf(0, bufnr)
```

7.11 Create scratch buffer

Category: Buffers
Tags: buffers, scratch, temporary, unlisted

Use `vim.api.nvim_create_buf()` to create scratch buffers for temporary content that won't be saved.

Example

```
-- Create scratch buffer (not listed, not a file)
local bufnr = vim.api.nvim_create_buf(false, true)

-- false = not listed in buffer list
-- true = scratch buffer (no swapfile, not saved)

-- Add content to scratch buffer
vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {
  "This is a scratch buffer",
  "It won't be saved",
  "Perfect for temporary notes or output"
})

-- Open in current window
vim.api.nvim_win_set_buf(0, bufnr)
```

7.12 Create terminal buffer programmatically

Category: Buffers
Tags: buffers, terminal, shell, interactive

Create and manage terminal buffers using the API.

Example

```
-- Create terminal buffer
local function create_terminal(cmd)
  cmd = cmd or vim.o.shell

  -- Create buffer for terminal
  local bufnr = vim.api.nvim_create_buf(false, true)

  -- Open terminal in buffer
  local term_id = vim.fn.termopen(cmd, {
    on_exit = function(job_id, exit_code, event)
      print("Terminal exited with code:", exit_code)
      vim.api.nvim_buf_delete(bufnr, {force = true})
    end,
  })

  -- Set buffer options
  vim.bo[bufnr].bufhidden = "hide"

  return bufnr, term_id
end

-- Open terminal in split
local function terminal_split(cmd, vertical)
  local bufnr, term_id = create_terminal(cmd)

  if vertical then
    vim.cmd("vsplit")
  else
    vim.cmd("split")
  end

  vim.api.nvim_win_set_buf(0, bufnr)

  -- Enter insert mode
  vim.cmd("startinsert")

  return bufnr, term_id
end

-- Open terminal in floating window
local function terminal_float(cmd)
  local bufnr, term_id = create_terminal(cmd)

  local width = math.floor(vim.o.columns * 0.8)
  local height = math.floor(vim.o.lines * 0.8)
```

```
  vim.api.nvim_open_win(bufnr, true, {
    relative = "editor",
    width = width,
    height = height,
    row = math.floor((vim.o.lines - height) / 2),
    col = math.floor((vim.o.columns - width) / 2),
    style = "minimal",
    border = "rounded",
  })

  vim.cmd("startinsert")

  return bufnr, term_id
end

-- Commands
vim.api.nvim_create_user_command("TermFloat", function(opts)
  terminal_float(opts.args ~= "" and opts.args or nil)
end, {nargs = "?"})
```

7.13 Delete buffer safely

Category: Buffers
Tags: buffers, delete, close, safe

Delete buffers with proper checks and force options.

Example

```
-- Safe buffer delete
local function delete_buffer(bufnr, force)
  bufnr = bufnr or 0
  force = force or false

  -- Check if buffer exists
  if not vim.api.nvim_buf_is_valid(bufnr) then
    print("Buffer doesn't exist")
    return false
  end

  -- Check if buffer is modified (unless force)
  if not force and vim.bo[bufnr].modified then
    print("Buffer has unsaved changes. Use force=true to delete anyway")
    return false
  end

  -- Delete buffer
  vim.api.nvim_buf_delete(bufnr, {force = force})
  return true
end

-- Delete all buffers except current
```

```
local function delete_other_buffers()
  local current = vim.api.nvim_get_current_buf()

  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
    if buf ~= current and vim.api.nvim_buf_is_loaded(buf) then
      delete_buffer(buf, false)
    end
  end
end

-- Delete all hidden/unlisted buffers
local function delete_hidden_buffers()
  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
    if not vim.bo[buf].buflisted and vim.api.nvim_buf_is_loaded(buf) then
      vim.api.nvim_buf_delete(buf, {force = true})
    end
  end
end

vim.api.nvim_create_user_command("BufOnly", delete_other_buffers, {})
vim.api.nvim_create_user_command("BufClean", delete_hidden_buffers, {})
```

7.14 Get buffer by name or number

Category: Buffers
Tags: buffers, find, search, lookup

Find buffers by name or number programmatically.

Example

```
-- Get buffer by name
local function get_buffer_by_name(name)
  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
    local buf_name = vim.api.nvim_buf_get_name(buf)
    if buf_name:match(name) then
      return buf
    end
  end
  return nil
end

-- Get buffer by exact name
local bufnr = vim.fn.bufnr("myfile.txt")
if bufnr ~= -1 then
  print("Found buffer:", bufnr)
end

-- Get all loaded buffers
local function get_loaded_buffers()
  local bufs = {}
  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
```

```
    if vim.api.nvim_buf_is_loaded(buf) then
      table.insert(bufs, {
        bufnr = buf,
        name = vim.api.nvim_buf_get_name(buf),
      })
    end
  end
  return bufs
end

-- Usage
local loaded = get_loaded_buffers()
for _, buf in ipairs(loaded) do
  print(buf.bufnr, buf.name)
end
```

7.15 Set buffer type for special buffers

Category: Buffers
Tags: buffers, buftype, special, nofile

Use `buftype` option to create special buffer types like help, quickfix, or custom tool buffers.

Example

```
local bufnr = vim.api.nvim_create_buf(false, true)

-- Different buffer types:
vim.bo[bufnr].buftype = "nofile"    -- Not associated with a file
vim.bo[bufnr].buftype = "nowrite"   -- Cannot be written
vim.bo[bufnr].buftype = "acwrite"   -- Use autocommand for writing
vim.bo[bufnr].buftype = "quickfix"  -- Quickfix buffer
vim.bo[bufnr].buftype = "help"      -- Help buffer
vim.bo[bufnr].buftype = "terminal"  -- Terminal buffer
vim.bo[bufnr].buftype = "prompt"    -- Prompt buffer (like cmdline)

-- Example: Create a read-only tool buffer
vim.bo[bufnr].buftype = "nofile"
vim.bo[bufnr].bufhidden = "wipe"    -- Delete when hidden
vim.bo[bufnr].swapfile = false
vim.bo[bufnr].modifiable = false    -- Read-only
```

7.16 Watch buffer for changes

Category: Buffers
Tags: buffers, watch, events, attach

Use `nvim_buf_attach()` to monitor buffer changes in real-time.

Example

```
-- Attach to buffer and watch changes
local function watch_buffer(bufnr)
  bufnr = bufnr or 0

  vim.api.nvim_buf_attach(bufnr, false, {
    on_lines = function(_, buf, _, first_line, last_line_old, last_line_new)
      print(string.format(
        "Lines changed in buffer %d: %d-%d (was %d lines, now %d)",
        buf, first_line, last_line_new,
        last_line_old - first_line,
        last_line_new - first_line
      ))

      -- React to changes
      -- e.g., auto-format, validate, update UI

      return false  -- don't detach
    end,

    on_changedtick = function(_, buf, tick)
      -- Called on every change (more frequent)
      print("Buffer changed, tick:", tick)
      return false
    end,

    on_detach = function(_, buf)
      print("Detached from buffer", buf)
    end,

    on_reload = function(_, buf)
      print("Buffer reloaded", buf)
      return false
    end,
  })
end

-- Example: Auto-save scratch buffer
local function create_auto_save_buffer()
  local bufnr = vim.api.nvim_create_buf(false, true)

  vim.api.nvim_buf_attach(bufnr, false, {
    on_lines = function()
      -- Save to temporary file
      local lines = vim.api.nvim_buf_get_lines(bufnr, 0, -1, false)
      local file = "/tmp/neovim_scratch.txt"
      vim.fn.writefile(lines, file)
      return false
    end,
  })

  return bufnr
end
```

CHAPTER 8
Builtin functions

8.1 Buffer and window information

Category: Functions
Tags: bufnr, winnr, tabpagenr, info

Use `bufnr()`, `winnr()`, `tabpagenr()` to get current buffer, window, and tab numbers for scripting.

Example

```
:echo bufnr('%')        " current buffer number
:echo winnr()           " current window number
:echo tabpagenr()       " current tab number
:echo winnr('$')        " total number of windows
```

8.2 Buffer content functions

Category: Functions
Tags: getbufline, setbufline, append, delete

Use `getbufline()` and `setbufline()` to read and modify buffer content without switching to the buffer.

Example

```
:echo getbufline(1, 1, 10)              " get lines 1-10 from buffer 1
:call setbufline(2, 1, 'new first line') " set line 1 in buffer 2
:call append(line('.'), 'new line')      " append after current line
:call delete(line('.'))                  " delete current line
```

8.3 Cursor and mark functions

Category: Functions
Tags: cursor, getpos, setpos, marks

Use `cursor()`, `getpos()`, `setpos()` for precise cursor and mark manipulation.

Example

```
:call cursor(10, 5)              " move cursor to line 10, column 5
:let pos = getpos('.')           " get current cursor position
:call setpos('.', pos)           " restore cursor position
:echo getpos("'a")               " get position of mark 'a'
```

8.4 Date and time functions

Category: Functions
Tags: strftime, localtime, getftime, date

Use `strftime()` and `localtime()` for date/time manipulation, and `getftime()` for file timestamps.

Example

```
:echo strftime('%Y-%m-%d %H:%M:%S')       " current date/time
:echo strftime('%Y-%m-%d', localtime())   " current date
:echo getftime(expand('%'))               " file modification time
:put =strftime('%Y-%m-%d')                " insert current date
```

8.5 File and directory functions

Category: Functions
Tags: glob, globpath, isdirectory, readable

Use `glob()`, `globpath()`, `isdirectory()` for file system operations and path expansion.

Example

```
:echo glob('*.txt')              " find all .txt files
:echo globpath(&rtp, 'plugin/*.vim')  " find plugins in runtimepath
:echo isdirectory(expand('%:h')) " check if directory exists
:echo readable(expand('%'))      " check if file is readable
```

8.6 Fold information functions

Category: Functions
Tags: foldclosed, foldtext, foldlevel, folding

Use folding functions to query and manipulate code folds programmatically.

Example

```
:echo foldclosed(line('.'))      " check if current line is folded
:echo foldlevel(line('.'))       " fold level of current line
:echo foldtext()                 " default fold text
:set foldtext=MyCustomFoldText() " custom fold text function
```

8.7 Get file type and encoding

Category: Functions
Tags: getftype, getfperm, file, info

Use `getftype()` to determine file type and `getfperm()` to get file permissions for the current or specified file.

Example

```
:echo getftype(expand('%'))     " file type (file, dir, link, etc.)
:echo getfperm(expand('%'))     " file permissions (rwxrwxrwx)
:echo getfsize(expand('%'))     " file size in bytes
```

8.8 Highlighting and syntax functions

Category: Functions
Tags: synID, synIDattr, hlID, syntax

Use syntax highlighting functions to query and manipulate syntax highlighting programmatically.

Example

```
:echo synID(line('.'), col('.'), 1)      " syntax ID under cursor
:echo synIDattr(synID(line('.'), col('.'), 1), 'name')  " syntax name
:echo hlID('Comment')                    " highlight group ID
:echo synIDattr(hlID('Comment'), 'fg')   " foreground color
```

8.9 Input and interaction functions

Category: Functions
Tags: input, inputsave, inputlist, confirm

Use `input()`, `inputlist()`, `confirm()` functions to create interactive vim scripts with user prompts.

Example

```
:let name = input('Enter name: ')         " prompt for input
:let choice = inputlist(['1. Red', '2. Blue', '3. Green'])
:let result = confirm('Save changes?', "&Yes\n&No\n&Cancel")
:echo "You chose: " . choice
```

8.10 Line and column functions

Category: Functions
Tags: line, col, getline, setline

Use `line()`, `col()`, `getline()`, `setline()` for precise cursor positioning and line manipulation.

Example

```
:echo line('.')         " current line number
:echo col('.')          " current column number
:echo getline('.')      " current line text
:call setline('.', 'new text')  " replace current line
```

8.11 List and dictionary functions

Category: Functions
Tags: len, empty, has_key, keys, values

Use `len()`, `empty()`, `has_key()`, `keys()`, `values()` for working with lists and dictionaries.

Example

```
:let mylist = [1, 2, 3]
:echo len(mylist)                          " length: 3
:echo empty(mylist)                        " false (0)
:let mydict = {'a': 1, 'b': 2}
:echo has_key(mydict, 'a')                " true (1)
:echo keys(mydict)                         " ['a', 'b']
```

8.12 Mathematical functions

Category: Functions
Tags: abs, pow, sqrt, sin, cos, math

Use built-in math functions like `abs()`, `pow()`, `sqrt()`, `sin()`, `cos()` for calculations in vim script.

Example

```
:echo abs(-5)                    " absolute value: 5
:echo pow(2, 3)                  " 2 to power of 3: 8
:echo sqrt(16)                   " square root: 4.0
:echo sin(3.14159/2)             " sine: ~1.0
:echo round(3.7)                 " round: 4
```

8.13 Path manipulation functions

Category: Functions
Tags: fnamemodify, resolve, simplify, path

Use `fnamemodify()` to manipulate file paths and `resolve()` to resolve symbolic links and shortcuts.

Example

```
:echo fnamemodify(expand('%'), ':p:h')     " full directory path
:echo fnamemodify(expand('%'), ':t:r')     " filename without extension
:echo resolve(expand('%'))                 " resolve symlinks
:echo simplify('../path/./file')           " normalize path
```

8.14 Register manipulation functions

Category: Functions
Tags: getreg, setreg, getregtype, registers

Use `getreg()`, `setreg()`, `getregtype()` to programmatically work with vim registers.

Example

```
:echo getreg('"')                " get default register content
:call setreg('a', 'hello world') " set register 'a'
:echo getregtype('a')            " get register type (v, V, or Ctrl-V)
:call setreg('+', @")            " copy default register to clipboard
```

8.15 Regular expression functions

Category: Functions
Tags: matchadd, matchdelete, matchlist, regex

Use `matchadd()`, `matchdelete()`, `matchlist()` for advanced pattern matching and highlighting.

Example

```
:let m = matchadd('Search', 'TODO')      " highlight all TODO
:call matchdelete(m)                     " remove highlighting
:echo matchlist('file.txt', '\\(.*\\)\\.\\(.*\\)')  " capture groups
:echo matchstr('hello123world', '\\d\\+') " extract digits: 123
```

8.16 Search and match functions

Category: Functions
Tags: search, searchpos, match, pattern

Use `search()`, `searchpos()`, and `match()` functions for programmatic searching without moving cursor.

Example

```
:echo search('pattern')                    " find pattern, return line
↪  number
:echo searchpos('pattern')                 " return [line, column]
:echo match('hello world', 'wor')          " find position in string (6)
:echo matchend('hello world', 'wor')       " end position (9)
```

8.17 String manipulation functions

Category: Functions
Tags: substitute, matchstr, split, string

Use `substitute()`, `matchstr()`, and `split()` functions for powerful string manipulation without changing buffers.

Example

```
:echo substitute("hello world", "world", "vim", "g")  " hello vim
:echo matchstr("file.txt", '\\.\\w\\+$')             " .txt
:echo split("a,b,c", ",")                            " ['a', 'b', 'c']
```

8.18 System and environment functions

Category: Functions
Tags: system, systemlist, environ, getenv

Use `system()` and `systemlist()` to execute shell commands and `getenv()` to access environment variables.

Example

```
:echo system('date')                    " execute shell command
:echo systemlist('ls -la')              " return as list
:echo getenv('HOME')                    " get environment variable
:echo exists('$EDITOR')                 " check if env var exists
```

8.19 Type checking functions

Category: Functions
Tags: type, islocked, exists, function

Use `type()`, `islocked()`, and `exists()` functions to check variable types and existence.

Example

```
:echo type(42)                  " 0 (Number)
:echo type("string")            " 1 (String)
:echo type([])                  " 3 (List)
:echo type({})                  " 4 (Dictionary)
:echo exists('g:my_var')        " check if variable exists
```

8.20 Window and tab functions

Category: Functions
Tags: winheight, winwidth, tabpagebuflist, winsaveview

Use window dimension and state functions to manage window layouts programmatically.

Example

```
:echo winheight(0)              " current window height
:echo winwidth(0)               " current window width
:let view = winsaveview()       " save cursor position and view
:call winrestview(view)         " restore saved view
:echo tabpagebuflist()          " list buffers in current tab
```

CHAPTER 9

Clever tricks

9.1 Alternative substitute delimiters

Category: Clever Tricks
Tags: substitute, delimiter, slash, alternative

Use any character as delimiter in substitute commands to avoid escaping slashes in paths.

Example

```
:s#/path/to/old#/path/to/new#g  " using # as delimiter
:s|/usr/bin|/usr/local/bin|g    " using | as delimiter
:s@old@new@g                    " using @ as delimiter
```

9.2 Auto-indent current block

Category: Clever Tricks
Tags: indent, block, braces, auto

Use `=%` when cursor is on opening brace to auto-indent entire block.

Example

```
=%  " auto-indent current block/braces
```

9.3 Auto-indent entire document

Category: Clever Tricks
Tags: indent, format, document, auto

Use `gg=G` to auto-indent entire document from top to bottom.

Example

```
gg=G  " auto-indent entire file
```

9.4 Calculation with expression register

Category: Clever Tricks
Tags: calculation, expression, register, math, evaluate

Use = register to evaluate mathematical expressions and insert results.

Example

```
" In insert mode:
Ctrl+r =2+3*4<Enter>    " inserts 14
Ctrl+r =sqrt(16)<Enter> " inserts 4.0
Ctrl+r =strftime("%Y")<Enter>  " inserts current year
```

9.5 Center line after jump

Category: Clever Tricks
Tags: center, jump, navigation

Append zz after navigation commands to center the line. Works with searches, line jumps, etc.

Example

```
42Gzz   " jump to line 42 and center
/foozz  " search for 'foo' and center
```

9.6 Change directory to current file

Category: Clever Tricks
Tags: directory, current, file, cd, path

Use `:cd %:h` to change directory to the directory of the current file.

Example

```
:cd %:h     " change to current file's directory
:pwd        " verify current directory
:lcd %:h    " change local directory for current window only
```

9.7 Change until character

Category: Clever Tricks
Tags: change, until, character

Use `ct{char}` to change text up to but not including character, or `cf{char}` to include the

character.

Example

```
ct;  " change until semicolon
cf;  " change including semicolon
```

9.8 Create word frequency table

Category: Clever Tricks
Tags: word, frequency, table, count, analysis

Create a word frequency analysis using Vim commands and external tools.

Example

```
" Create word frequency table:
:%s/\W\+/\r/g | sort | uniq -c | sort -nr
" Or using Vim's internal commands:
:g/./normal 0"ay$
```

9.9 Enhanced repeat with cursor positioning

Category: Clever Tricks
Tags: repeat, cursor, position, change, dot

Map . followed by ` to repeat last command and return cursor to start of change.

Example

```
" Add this mapping:
nnoremap <leader>. .`[

" Now after making a change:
<leader>.  " repeat change and go to start position
```

9.10 File encoding in status line

Category: Clever Tricks
Tags: encoding, status, line, file, format

Add file encoding to status line to see current file's character encoding.

Example

```
:set statusline=%f\ [%{&fileencoding?&fileencoding:&encoding}]\ %y
" Shows filename, encoding, and filetype
```

9.11 G-commands - Rot13 encoding

Category: Clever Tricks
Tags: rot13, encode, cipher, text

Use `g?{motion}` to apply Rot13 encoding to text (shifts letters by 13).

Example

```
g?iw  " apply Rot13 to word under cursor
g??   " apply Rot13 to current line
```

9.12 G-commands - case conversion

Category: Clever Tricks
Tags: case, convert, upper, lower

Use `gU{motion}` for uppercase, `gu{motion}` for lowercase, and `g~{motion}` to toggle case.

Example

```
gUw   " uppercase word
guu   " lowercase current line
g~iw  " toggle case of word under cursor
```

9.13 G-commands - display command output

Category: Clever Tricks
Tags: display, command, output, history

Use `g<` to display the output of the previous command.

Example

```
g<  " display previous command output
```

9.14 G-commands - execute application

Category: Clever Tricks
Tags: execute, application, file, system

Use `gx` to execute the default application for the file/URL under cursor.

Example

```
gx  " open file/URL under cursor with default app
```

9.15 G-commands - format keeping cursor

Category: Clever Tricks
Tags: format, cursor, position, text

Use `gw{motion}` to format text while keeping cursor position unchanged.

Example

```
gwap  " format paragraph, keep cursor position
```

9.16 G-commands - join without space

Category: Clever Tricks
Tags: join, line, space

Use `gJ` to join lines without inserting a space between them.

Example

```
gJ  " join lines without adding space
```

9.17 G-commands - mark navigation without jumplist

Category: Clever Tricks
Tags: mark, navigation, jumplist

Use `g'` and `g`` to jump to marks without changing the jumplist.

Example

```
g'a  " jump to mark 'a' without affecting jumplist
g`a  " jump to exact position of mark 'a' without jumplist
```

9.18 G-commands - middle of line

Category: Clever Tricks
Tags: middle, line, screen, text

Use `gm` to go to middle of screen line and `gM` to go to middle of text line.

Example

```
gm  " go to middle of screen line
gM  " go to middle of text line
```

9.19 G-commands - put and leave cursor

Category: Clever Tricks
Tags: put, paste, cursor, position

Use `gp` and `gP` to put text and leave cursor after the pasted text.

Example

```
gp  " put after and leave cursor at end
gP  " put before and leave cursor at end
```

9.20 G-commands - repeat substitute

Category: Clever Tricks
Tags: substitute, repeat, global, command

Use `g&` to repeat the last `:substitute` command on all lines.

Example

```
:s/old/new/   " substitute on current line
g&            " repeat substitute on all lines
```

9.21 G-commands - screen line movement

Category: Clever Tricks
Tags: screen, line, wrap, movement

Use `gj` and `gk` to move by screen lines when text is wrapped, `g0` and `g$` for screen line start/end.

Example

```
gj  " move down by screen line (with wrap)
gk  " move up by screen line (with wrap)
g0  " go to start of screen line
g$  " go to end of screen line
```

9.22 G-commands - search and select

Category: Clever Tricks
Tags: search, select, visual, pattern

Use `gn` to find and visually select next search match, `gN` for previous match.

Example

```
/pattern<Enter>  " search for pattern first
gn               " select next match
gN               " select previous match
```

9.23 G-commands - search variations

Category: Clever Tricks
Tags: search, variations, boundaries

Use `g*` and `g#` to search for word under cursor without word boundaries (matches partial words).

Example

```
g*  " search forward for word without boundaries
g#  " search backward for word without boundaries
```

9.24 G-commands - select modes

Category: Clever Tricks
Tags: select, mode, visual, block

Use `gh` for select mode, `gH` for select line mode, `g Ctrl+h` for select block mode.

Example

```
gh       " start select mode
gH       " start select line mode
g Ctrl+h " start select block mode
```

9.25 G-commands - sleep

Category: Clever Tricks
Tags: sleep, delay, pause

Use `gs` to make Neovim sleep for specified seconds (useful in scripts).

Example

```
3gs  " sleep for 3 seconds
gs   " sleep for 1 second (default)
```

9.26 G-commands - undo branches

Category: Clever Tricks
Tags: undo, branch, time, state

Use `g-` and `g+` to navigate through undo branches by time.

Example

```
g-  " go to older text state
g+  " go to newer text state
```

9.27 G-commands - virtual replace

Category: Clever Tricks
Tags: virtual, replace, mode, character

Use `gR` to enter virtual replace mode, `gr{char}` to replace character without affecting layout.

Example

```
gR    " enter virtual replace mode
grx   " replace character with 'x' virtually
```

9.28 Line completion in insert mode

Category: Clever Tricks
Tags: completion, line, insert, auto

Use `Ctrl+X Ctrl+L` in insert mode to complete entire lines from current buffer.

Example

```
" In insert mode:
Ctrl+X Ctrl+L  " complete entire line
```

9.29 List lines matching last search

Category: Clever Tricks
Tags: search, list, global, pattern, last

Use `:g//` to list all lines containing the last search pattern without specifying the pattern again.

Example

```
/function   " search for 'function'
:g//        " list all lines containing 'function'
:g//p       " same as above (print is default)
```

9.30 Open URL from current line

Category: Clever Tricks
Tags: url, open, browser, web, link

Use `gx` to open URL under cursor, or create mapping to open entire line as URL.

Example

```
gx  " open URL under cursor with default browser

" Custom mapping for entire line:
nnoremap <leader>o :!open <cWORD><CR>
```

9.31 Open file under cursor

Category: Clever Tricks
Tags: file, open, cursor, path

Use `gf` to open file whose name is under cursor. Use `gF` to go to specific line number.

Example

```
gf   " open file under cursor
gF   " open file and go to line number
```

9.32 Quick number increment

Category: Clever Tricks
Tags: number, increment, math

Use `Ctrl+a` to increment number under cursor, `Ctrl+x` to decrement. Works with deci-

mals and hex.

Example

```
Ctrl+a  " increment number
Ctrl+x  " decrement number
```

9.33 Quick substitute word

Category: Clever Tricks
Tags: substitute, word, replace

Use `ciw{newword}` to change inner word. Position cursor anywhere in word and type replacement.

Example

```
ciwfoo  " change word to 'foo'
```

9.34 Repeat last Ex command with @:

Category: Clever Tricks
Tags: repeat, ex, command, macro, colon

Use `@:` to repeat the last Ex command, similar to how `@@` repeats macros.

Example

```
:substitute/old/new/g
@:  " repeat the last substitute command
```

9.35 Save each line to separate files

Category: Clever Tricks
Tags: file, save, line, separate, export

Use `:g/^/exe` to save each line to a separate file with incremental names.

Example

```
:let i = 1 | g/^/exe 'w! line' . i . '.txt' | let i = i + 1
" Saves each line to line1.txt, line2.txt, etc.
```

9.36 Scroll windows together

Category: Clever Tricks
Tags: scroll, window, together, bind, sync

Use `:set scrollbind` in multiple windows to scroll them together synchronously.

Example

```
" In first window:
:set scrollbind

" In second window:
:set scrollbind

" Now both windows scroll together
" To disable:
:set noscrollbind
```

9.37 Search for lines NOT matching pattern

Category: Clever Tricks
Tags: search, not, matching, invert, negative

Use `:v/pattern/` or `:g!/pattern/` to work with lines that do NOT match a pattern.

Example

```
:v/TODO/d       " delete lines NOT containing TODO
:g!/function/p  " print lines NOT containing 'function'
:v/^$/d         " delete non-empty lines (keep only empty lines)
```

9.38 Split line at cursor

Category: Clever Tricks
Tags: split, line, break

Use `i` followed by `Enter` then `Esc`, or more efficiently `r` followed by `Enter` to break line at cursor.

Example

```
i<Enter><Esc>  " split line at cursor
```

9.39 Swap assignment statement sides

Category: Clever Tricks
Tags: swap, assignment, left, right, substitute

Use substitute with groups to swap left and right sides of assignment statements.

Example

```
" Swap variable assignment (a = b becomes b = a):
:%s/\(\w\+\)\s*=\s*\(\w\+\)/\2 = \1/g

" Swap in selected region:
:'<,'>s/\(\w\+\)\s*=\s*\(\w\+\)/\2 = \1/g
```

9.40 Swap two characters

Category: Clever Tricks
Tags: character, swap, transpose

Use `xp` to swap current character with next character.

Example

```
xp  " swap characters
```

9.41 Toggle text case inside a HTML tag

Category: Clever Tricks
Tags: edit, case, tag

Use `g~it` to change the case of the text inside a html tag. Cursor should be between opening and closing HTML tag.

Example

```
" turns <b>important</b> into <b>IMPORTANT</b>
g~it
```

9.42 Visual line selection shortcut

Category: Clever Tricks
Tags: visual, line, selection

Use `V` to select entire line immediately, then `j`/`k` to extend selection.

Example

```
Vjjj  " select current line + 3 below
```

9.43 Word count in selection or file

Category: Clever Tricks
Tags: word, count, selection, statistics, file

Use `g Ctrl+g` to show word count, or `:!wc -w %` for file word count.

Example

```
" Select text in visual mode, then:
g Ctrl+g     " show character, word, line count of selection

" For entire file:
:!wc -w %    " show word count of current file
```

9.44 Z-commands - spelling corrections

Category: Clever Tricks
Tags: spelling, correction, dictionary

Use `z=` for spelling suggestions, `zg` to add word to dictionary, `zw` to mark as misspelled, `zG`/`zW` for temporary marks.

Example

```
z=  " show spelling suggestions for word under cursor
zg  " add word to personal dictionary (good)
zw  " mark word as misspelled (wrong)
zG  " temporarily mark word as correct
zW  " temporarily mark word as incorrect
```

CHAPTER 10
Clipboard

10.1 GNU/Linux clipboard with xclip

Category: Clipboard
Tags: linux, clipboard, xclip, copy, paste

Use xclip utility for clipboard integration on GNU/Linux systems.

Example

```
" Copy/paste with xclip
vnoremap <C-c> :w !xclip -selection clipboard<CR><CR>
nnoremap <C-v> :r !xclip -selection clipboard -o<CR>

" Function-based approach
function! ClipboardYank()
  call system('xclip -i -selection clipboard', @@)
endfunction
```

10.2 Mac OS clipboard sharing

Category: Clipboard
Tags: macos, clipboard, pbcopy, pbpaste

Integrate Vim with macOS clipboard using pbcopy and pbpaste utilities.

Example

```
" macOS clipboard integration
vnoremap <C-c> :w !pbcopy<CR><CR>
nnoremap <C-v> :r !pbpaste<CR>

" Use system clipboard by default
set clipboard=unnamed
```

10.3 Preserve register when pasting over selection

Category: Clipboard
Tags: paste, register, yank, visual, black-hole

When pasting over a visual selection, the deleted text normally replaces the unnamed register. Use the black hole register to preserve your yanked content.

Example

```
-- Paste without losing the yanked content
vim.keymap.set("x", "p", '"_dP', { desc = "Paste without yanking", noremap =
↪  true })

-- Alternative: use a specific register
vim.keymap.set("x", "<leader>p", '"0p', { desc = "Paste from yank register",
↪  noremap = true })
```

Example

```
" In vimscript:
xnoremap p "_dP
" This deletes to black hole register (_) then pastes
```

10.4 Set system clipboard from Lua

Category: Clipboard
Tags: clipboard, lua, register

Use `vim.fn.setreg("+", "text")` to set system clipboard content from Lua.

Example

```
:lua vim.fn.setreg("+", "hello world")
```

10.5 System clipboard access with registers

Category: Clipboard
Tags: clipboard, system, copy, paste, register

Access system clipboard using + and * registers for seamless integration with other applications.

Example

```
" Copy to system clipboard
"+y         " yank to + register (desktop clipboard)
"*y         " yank to * register (mouse selection)
gg"+yG      " copy entire buffer to system clipboard

" Paste from system clipboard
"+p         " paste from + register
"*p         " paste from * register
```

10.6 System clipboard sync

Category: Clipboard
Tags: clipboard, system, sync

Use `vim.opt.clipboard="unnamedplus"` to sync yank/paste with system clipboard automatically.

Example

```
:lua vim.opt.clipboard = "unnamedplus"
```

10.7 System clipboard: handling yank and delete motions differently

Category: Clipboard
Tags: clipboard, copy, paste

Suppose that you want yank and delete motions to behave differently with respect to system clipboard. For example, you want all yanked text to be copied to system clipboard as well to unnamed internal register. But in case of delete motions, you don't want to affect system clipboard. The setup is fairly easy. Just add the following lines to your `init.lua` configuration file

Example

```
-- Avoid global clipboard hijacking
vim.opt.clipboard = {}
-- NOTE: Yank should copy to unnamed register AND system clipboard
-- Deleted text goes to unnamed register only without changing system
 ↪  clipboard
vim.keymap.set({ "n", "x" }, "y", '"+y', { desc = "Yank to clipboard",
 ↪  noremap = true })
vim.keymap.set("n", "yy", '"+yy', { desc = "Yank to clipboard", noremap =
 ↪  true })
```

CHAPTER 11
Command line

11.1 Command completion

Category: Command Line
Tags: command, completion, tab

Use `Tab` for command completion and `Ctrl+d` to list all possible completions.

Example

```
:ed<Tab>   " complete to :edit
:h vim<Tab>  " complete help topics
:set nu<Ctrl+d>  " list all options starting with 'nu'
```

11.2 Command line editing

Category: Command Line
Tags: command, edit, navigation

Use `Ctrl+b` to go to beginning of line, `Ctrl+e` to end, `Ctrl+h` to delete character, `Ctrl+w` to delete word.

Example

```
:Ctrl+b  " go to beginning of command line
:Ctrl+e  " go to end of command line
:Ctrl+h  " delete character backward
:Ctrl+w  " delete word backward
```

11.3 Command-line completion modes

Category: Command Line
Tags: command, completion, tab, modes

Use `Tab` for next completion, `Shift+Tab` for previous, `Ctrl+d` to list all, `Ctrl+a` to insert all matches, `Ctrl+l` for longest common part.

Example

```
" In command mode:
:e <Tab>        " complete filename
:e <Shift+Tab>  " previous completion
:set <Ctrl+d>   " list all completions
:b <Ctrl+a>     " insert all buffer matches
:help <Ctrl+l>  " complete to longest common part
```

11.4 Command-line cursor movement

Category: Command Line
Tags: command, cursor, movement, navigation

Use arrow keys or `Ctrl+b`/`Ctrl+e` for movement, `Shift+Left`/`Shift+Right` or `Ctrl+Left`/`Ctrl+Right` for word movement.

Example

```
" In command mode:
<Left>/<Right>     " move cursor by character
Ctrl+b/Ctrl+e      " move to beginning/end of line
Shift+Left/Right   " move by word
Ctrl+Left/Right    " move by word (alternative)
```

11.5 Command-line deletion operations

Category: Command Line
Tags: command, delete, backspace, clear

Use `Backspace` or `Ctrl+h` to delete character, `Del` to delete forward, `Ctrl+w` to delete word, `Ctrl+u` to clear line.

Example

```
" In command mode:
<BS>/Ctrl+h  " delete character backward
<Del>        " delete character forward
Ctrl+w       " delete word backward
Ctrl+u       " clear from cursor to beginning
```

11.6 Command-line history with filtering

Category: Command Line
Tags: command, history, filter, search

Use `Shift+Up`/`Shift+Down` or `PageUp`/`PageDown` to recall commands that start with current

text.

Example

```
" Type partial command, then:
:se<Shift+Up>    " find previous commands starting with 'se'
:ed<PageDown>    " find next commands starting with 'ed'
```

11.7 Command-line literal insertion

Category: Command Line
Tags: command, literal, insert, special

Use `Ctrl+v` or `Ctrl+q` to insert the next character literally (useful for special characters).

Example

```
" In command mode:
:echo "Ctrl+v<Tab>"   " insert literal tab character
:s/Ctrl+v<Esc>/x/g    " search for literal Esc character
```

11.8 Command-line mode switching

Category: Command Line
Tags: command, mode, switch, abandon

Use `Ctrl+c` or `Esc` to abandon command, `Ctrl+\ Ctrl+n` or `Ctrl+\ Ctrl+g` to go to normal mode.

Example

```
" In command mode:
Ctrl+c           " abandon command without executing
<Esc>            " abandon command (alternative)
Ctrl+\ Ctrl+n    " go to normal mode
Ctrl+\ Ctrl+g    " go to normal mode (alternative)
```

11.9 Command-line register insertion

Category: Command Line
Tags: command, register, insert, content

Use `Ctrl+r` followed by register name to insert register contents into command line.

Example

```
" In command mode:
:Ctrl+r "     " insert default register
:Ctrl+r a     " insert register 'a'
:Ctrl+r %     " insert current filename
:Ctrl+r :     " insert last command
:Ctrl+r /     " insert last search pattern
```

11.10 Command-line special insertions

Category: Command Line
Tags: command, insert, word, filename, line

Use `Ctrl+r` with special keys to insert current context: `Ctrl+w` for word, `Ctrl+f` for filename, `Ctrl+l` for line.

Example

```
" In command mode:
:Ctrl+r Ctrl+w  " insert word under cursor
:Ctrl+r Ctrl+f  " insert filename under cursor
:Ctrl+r Ctrl+p  " insert filename with path expansion
:Ctrl+r Ctrl+a  " insert WORD under cursor
:Ctrl+r Ctrl+l  " insert line under cursor
```

11.11 Command-line window access

Category: Command Line
Tags: command, window, edit, history

Use `Ctrl+f` to open command-line window for full editing, `Ctrl+o` to execute one normal mode command.

Example

```
" In command mode:
Ctrl+f  " open command-line window for editing
Ctrl+o  " execute one normal mode command and return
```

11.12 Command-line word manipulation

Category: Command Line
Tags: command, word, delete, kill, clear

Use `Ctrl+w` to delete word before cursor, `Ctrl+u` to delete from cursor to beginning of line.

Example

```
" In command mode:
Ctrl+w  " delete word before cursor
Ctrl+u  " delete from cursor to beginning
Ctrl+k  " delete from cursor to end of line
```

11.13 Insert word under cursor in command

Category: Command Line
Tags: command, word, cursor

Use `Ctrl+r Ctrl+w` to insert the word under cursor into command line.

Example

```
:Ctrl+r Ctrl+w  " insert word under cursor
```

11.14 Open command history

Category: Command Line
Tags: history, command, window

Use `q:` to open command history in a searchable window.

Example

```
q:  " open command history window
```

CHAPTER 12

Command line (advanced)

12.1 Command line abbreviations and shortcuts

Category: Command Line Advanced
Tags: abbreviation, shortcut, cabbrev, expand

Create command line abbreviations for frequently used commands.

Example

```
:cabbrev W w                 " expand W to w
:cabbrev Q q                 " expand Q to q
:cabbrev Wq wq               " expand Wq to wq
:cabbrev vsb vert sb         " expand vsb to 'vert sb'
:cabbrev today put =strftime('%Y-%m-%d')  " insert today's date
```

12.2 Command line advanced search operations

Category: Command Line Advanced
Tags: search, advanced, pattern, replace, scope

Perform sophisticated search operations from command line.

Example

```
:vimgrep /pattern/ **/*.js   " search in all JS files recursively
:lvimgrep /TODO/ %           " search in current file (location list)
:grep -r "pattern" --include="*.py" .  " external grep
:helpgrep pattern            " search help files
:g/pattern1/s/pattern2/replacement/g   " conditional substitute
```

12.3 Command line advanced substitution techniques

Category: Command Line Advanced
Tags: substitute, advanced, technique, pattern

Master advanced substitution patterns and techniques.

Example

```
:%s/\v(word1|word2)/\U\1/g   " uppercase specific words
:%s/\(.*\)\n\1/\1/           " remove duplicate consecutive lines
:%s/^\s*\(.*\S\)\s*$/\1/     " trim leading/trailing whitespace
:%s/\%V.*\%V/\=substitute(submatch(0), 'a', 'A', 'g')  " in visual selection
```

12.4 Command line buffer and window targeting

Category: Command Line Advanced
Tags: buffer, window, target, specific, operation

Target specific buffers and windows for command execution.

Example

```
:bufdo %s/old/new/ge         " execute in all buffers
:windo set number            " execute in all windows
:tabdo echo tabpagenr()      " execute in all tabs
:argdo %s/pattern/replace/ge " execute on argument list files
:cdo s/old/new/g             " execute on quickfix list items
```

12.5 Command line completion customization

Category: Command Line Advanced
Tags: completion, custom, wildmenu, wildmode

Customize command line completion behavior and appearance.

Example

```
:set wildmenu                " enable command completion menu
:set wildmode=longest:full,full  " completion behavior
:set wildignore=*.o,*.pyc,*.swp  " ignore patterns
:set wildoptions=pum         " use popup menu for completion
:set pumheight=15            " limit popup menu height
```

12.6 Command line conditional execution

Category: Command Line Advanced
Tags: conditional, execute, if, expression

Execute commands conditionally using expressions and logic.

Example

```
:if line('.') > 100 | echo "Large file" | endif
:execute line('.') > 50 ? 'echo "Past line 50"' : 'echo "Early in file"'
:silent! write               " suppress error messages
:try | source ~/.vimrc | catch | echo "Config error" | endtry
```

12.7 Command line custom command creation

Category: Command Line Advanced
Tags: command, custom, user, define, parameter

Create sophisticated custom commands with parameters and completion.

Example

```
" Command with file completion
:command! -nargs=1 -complete=file EditConfig edit ~/.config/<args>

" Command with custom completion
:command! -nargs=1 -complete=custom,MyComplete MyCmd echo <args>
function! MyComplete(ArgLead, CmdLine, CursorPos)
  return ['option1', 'option2', 'option3']
endfunction

" Range command with count
:command! -range=% -nargs=1 ReplaceAll <line1>,<line2>s/<args>/g
```

12.8 Command line debugging and inspection

Category: Command Line Advanced
Tags: debug, inspect, verbose, trace

Debug command execution and inspect Vim state from command line.

Example

```
:verbose map <leader>        " show where mapping was defined
:verbose set tabstop?        " show where option was last set
:function                    " list all user-defined functions
:scriptnames                 " list all sourced scripts
:messages                    " show message history
:redir @a | silent! command | redir END  " redirect output to register
```

12.9 Command line environment variable integration

Category: Command Line Advanced
Tags: environment, variable, expand, system

Work with environment variables and system integration.

Example

```
:echo $HOME                  " display environment variable
:let $MYVAR = 'value'       " set environment variable
:edit $HOME/.vimrc          " use environment variable in path
:!echo $PATH                " use in external command
:put =expand('$USER')       " insert environment variable value
```

12.10 Command line error handling

Category: Command Line Advanced
Tags: error, silent, try, catch, handling

Handle errors gracefully in command line operations.

Example

```
:silent! command             " suppress error messages
:try | risky_command | catch /^Vim/ | echo "Vim error" | endtry
:if exists(':SomeCommand') | SomeCommand | endif
:command! -bang MyCmd if <bang>0 | echo "Bang!" | else | echo "No bang" |
↪  endif
```

12.11 Command line expression evaluation

Category: Command Line Advanced
Tags: expression, evaluation, calculation, register

Use `Ctrl+r =` to evaluate expressions and insert results into command line.

Example

```
" In command line:
:echo <Ctrl+r⩾2*3<CR>        " insert 6
:edit /path/<Ctrl+r⩾strftime("%Y")<CR>/file.txt  " insert current year
:let var = <Ctrl+r⩾line('.')*2<CR>  " multiply current line by 2
```

Title: Insert word under cursor in command line # Category: Command Line Advanced # Tags: command-line, register, word, cursor, <C-r><C-w> — Use `<C-r><C-w>` in command-line mode to insert the word under the cursor, perfect for quick substitutions.

Example

```
" Position cursor on 'oldword' then:
:%s//<C-r><C-w>/g     " substitute oldword with word under cursor
:grep <C-r><C-w> **   " search for word under cursor in all files
:help <C-r><C-w>      " get help for word under cursor
```

12.12 Command line external command integration

Category: Command Line Advanced
Tags: external, command, shell, filter, system

Integrate external commands seamlessly with Vim command line.

Example

```
:r !date                      " insert date command output
:.,+5!sort                    " sort next 5 lines with external sort
:!ls                          " run ls and show output
:!!                           " repeat last external command
:.!tr '[:lower:]' '[:upper:]' " convert current line to uppercase
```

12.13 Command line filename completion variations

Category: Command Line Advanced
Tags: completion, filename, path, directory

Use different completion types for files, directories, and patterns.

Example

```
" In command line:
:edit <Ctrl+x><Ctrl+f>        " filename completion
:cd <Ctrl+x><Ctrl+d>          " directory completion
:help <Ctrl+x><Ctrl+v>        " Vim command completion
:set <Ctrl+x><Ctrl+o>         " option completion
```

12.14 Command line history search and filtering

Category: Command Line Advanced
Tags: history, search, filter, pattern

Search and filter command history with patterns and ranges.

Example

```
:history /pattern/            " search command history for pattern
:history : 10                 " show last 10 commands
:history / 5,10               " show search history items 5-10
:history =                    " show expression history
```

12.15 Command line job control and async

Category: Command Line Advanced
Tags: job, async, background, control

Control background jobs and asynchronous operations.

Example

```
:call jobstart(['ls', '-la'])           " start async job
:let job = jobstart('long_command', {'on_exit': 'MyHandler'})
:call jobstop(job)                      " stop job
:call jobwait([job], 5000)              " wait for job with timeout
```

12.16 Command line macro recording and playback

Category: Command Line Advanced
Tags: macro, record, playbook, command, automation

Record and replay command sequences for automation.

Example

```
:let @q = 'command sequence' " store command in register q
:normal @q                   " execute commands from register q
:g/pattern/normal @q         " execute macro on matching lines
:%normal @q                  " execute macro on all lines
```

12.17 Command line range shortcuts

Category: Command Line Advanced
Tags: range, shortcut, selection, lines

Use range shortcuts for efficient line selection in commands.

Example

```
:.,$d                         " delete from current line to end
:.,+5s/old/new/g             " substitute from current to +5 lines
:'a,'bs/foo/bar/g            " substitute from mark 'a' to mark 'b'
:/pattern/,/end/d            " delete from pattern to 'end'
:1,10!sort                   " sort lines 1-10 with external command
```

12.18 Command line register manipulation

Category: Command Line Advanced
Tags: register, insert, content, reference

Access and manipulate registers from command line efficiently.

Example

```
" In command line:
:<Ctrl+r>"                    " insert default register
:<Ctrl+r>a                    " insert register 'a'
:<Ctrl+r>%                    " insert current filename
:<Ctrl+r>#                    " insert alternate filename
:<Ctrl+r>:                    " insert last command
:<Ctrl+r>/                    " insert last search pattern
```

12.19 Command line script execution

Category: Command Line Advanced
Tags: script, execute, source, runtime

Execute scripts and source files with advanced options.

Example

```
:source %                     " source current file
:so $MYVIMRC                  " source vimrc
:runtime! plugin/**/*.vim     " source all plugins
:execute 'source' fnameescape(expand('~/.config/nvim/init.lua'))
:luafile %                    " execute current Lua file
```

12.20 Command line substitution flags and modifiers

Category: Command Line Advanced
Tags: substitute, flags, modifier, advanced

Use advanced substitution flags for precise control over replacements.

Example

```
:%s/old/new/gc                " global with confirmation
:%s/old/new/I                 " case sensitive (ignore ignorecase setting)
:%s/old/new/gn                " show matches without replacing
:%s//~/g                      " replace last search with last substitute
:%s/pattern/\=submatch(0)*2/g " use expression in replacement
```

12.21 Command line terminal integration

Category: Command Line Advanced
Tags: terminal, integration, shell, command

Integrate terminal operations seamlessly with command line.

Example

```
:terminal                    " open terminal in split
:vert terminal              " open vertical terminal
:terminal ++close grep pattern *.txt  " run command and close
:let @" = system('date')    " capture system command output
:put =system('whoami')      " insert system command result
```

12.22 Command line window operations

Category: Command Line Advanced
Tags: window, command, edit, history

Use command line window for advanced command editing and history.

Example

```
q:                           " open command history window
q/                           " open search history window
:<Ctrl+f>                    " switch to command line window from command
 ↪  line
" In command window: <CR> executes, <Ctrl+c> closes
```

CHAPTER 13
Community tips

13.1 Advanced completion shortcuts

Category: Completion
Tags: completion, ctrl-x, advanced, shortcuts

Use Ctrl+X completion modes for different types of intelligent completion in insert mode.

Example

```
" In insert mode:
Ctrl+x Ctrl+p  " word completion with suggestions
Ctrl+x Ctrl+l  " complete entire lines
Ctrl+x Ctrl+k  " dictionary word completion
Ctrl+x Ctrl+]  " tag-based completion
Ctrl+x Ctrl+f  " filename completion
Ctrl+x Ctrl+o  " omni completion (context-aware)
```

13.2 Buffer-specific settings

Category: Configuration
Tags: buffer, specific, settings, local

Use buffer-local settings and autocmds for file-type specific configurations and optimizations.

Example

```
:autocmd BufEnter *.lua setlocal tabstop=2 shiftwidth=2
:autocmd BufEnter *.py setlocal tabstop=4 shiftwidth=4
:autocmd BufEnter *.md setlocal textwidth=80 spell
" File-specific settings without global impact
```

13.3 Builtin completion without plugins

Category: Completion
Tags: builtin, completion, native, plugin-free

Use Neovim's built-in completion capabilities for intelligent code completion without external plugins.

Example

```
:set completeopt=menu,menuone,noselect,preview
:inoremap <Tab> <C-n>
:inoremap <S-Tab> <C-p>
" Ctrl+n/Ctrl+p for next/previous completion
" Ctrl+x Ctrl+o for omni completion (language-aware)
```

13.4 Command abbreviations

Category: Command Line
Tags: abbreviations, shortcuts, efficiency, typos

Use command abbreviations for frequently used commands and common typo corrections.

Example

```
:cabbrev W w
:cabbrev Wq wq
:cabbrev Q q
:cabbrev vsf vert sfind
:cabbrev ff find **/*
" Corrects common typos and creates shortcuts
```

13.5 Command-line window editing

Category: Command Line
Tags: command, window, editing, history

Use command-line window for advanced command history editing and complex command construction.

Example

```
q:  " open command history in editable window
q/  " open search history in editable window
q?  " open search history (backward) in editable window
" Edit commands like regular text, press Enter to execute
" Navigate with vim motions, make complex edits
```

13.6 Dynamic plugin management

Category: Configuration
Tags: lazy, plugin, dynamic, management

Use dynamic plugin installation and loading patterns inspired by TJ DeVries for self-bootstrapping configurations.

Example

```
local lazypath = vim.fn.stdpath("data") .. "/lazy/lazy.nvim"
if not vim.uv.fs_stat(lazypath) then
  vim.fn.system({
    "git", "clone", "--filter=blob:none",
    "https://github.com/folke/lazy.nvim.git",
    "--branch=stable", lazypath,
  })
end
vim.opt.rtp:prepend(lazypath)
```

13.7 Efficient whitespace cleanup

Category: Text Manipulation
Tags: whitespace, cleanup, trailing, efficiency

Use F-key mapping for instant trailing whitespace removal with user feedback across entire buffer.

Example

```
:noremap <F5> :%s/\s\+$//<CR>:echo 'All trailing whitespace removed.'<CR>
" One key press to clean entire file and confirm action
" Works in any mode, provides immediate feedback
```

13.8 Environment-aware configuration

Category: Configuration
Tags: environment, conditional, config, dotenv

Use environment variables and conditional loading for portable configurations across different machines.

Example

```
" Load local environment variables
if filereadable(expand('~/.config/nvim/.env'))
  for line in readfile(expand('~/.config/nvim/.env'))
    let env_var = split(line, '=')
```

```
    if len(env_var) ⩾ 2
      execute 'let $' . env_var[0] . '="' . join(env_var[1:], '=') . '"'
    endif
  endfor
endif
```

13.9 Help in new tab workflow

Category: Workflow
Tags: help, tab, workflow, reference

Use custom mapping to open help documentation in new tabs for better reference workflow during coding.

Example

```
:nnoremap <leader>h :tabnew<CR>:help<CR><C-w><C-w>:quit<CR>
" Opens help in new tab, focuses on help content, closes empty buffer
" Provides dedicated space for documentation reference
```

13.10 Insert mode line manipulation

Category: Editing
Tags: insert, line, manipulation, efficiency

Use Alt key combinations to add new lines above/below without leaving insert mode or changing cursor position.

Example

```
:inoremap <M-o> <Esc>o<Esc>a    " add line below, return to insert
:inoremap <M-O> <Esc>O<Esc>a    " add line above, return to insert
" Maintains flow during writing/coding without mode switches
```

13.11 Insert mode navigation

Category: Insert
Tags: insert, navigation, movement, efficiency

Use insert mode navigation keys for efficient editing without leaving insert mode frequently.

Example

```
<C-h>   " backspace (delete left)
<C-w>   " delete word left
<C-u>   " delete to beginning of line
<C-t>   " indent current line
<C-d>   " unindent current line
<C-o>   " execute one normal mode command
```

13.12 Mark-based navigation workflow

Category: Marks
Tags: marks, navigation, workflow, jumping

Use marks for efficient navigation between important locations in large files and projects.

Example

```
ma      " set mark 'a' at current position
'a      " jump to line of mark 'a'
`a      " jump to exact position of mark 'a'
:marks  " list all marks
mA      " set global mark 'A' (across files)
'A      " jump to global mark 'A'
```

13.13 Modular configuration loading

Category: Configuration
Tags: modular, import, require, organization

Use Lua's require system with custom import directories for organized, modular configuration management.

Example

```
-- Structure: ~/.config/nvim/lua/custom/
require('lazy').setup({
  { import = "custom.plugins" },  -- loads plugins from custom/plugins/
  { import = "custom.lsp" },      -- loads LSP configs from custom/lsp/
}, {
  change_detection = { notify = false }
})
```

13.14 Motion-based editing patterns

Category: Movement
Tags: motion, editing, patterns, efficiency

Use motion commands combined with operators for efficient text editing patterns and muscle memory.

Example

```
ci"     " change inside quotes
ca(     " change around parentheses
di}     " delete inside braces
ya]     " yank around brackets
viw     " visually select inner word
vap     " visually select around paragraph
```

13.15 Quick fold navigation

Category: Folding
Tags: fold, navigation, quick, movement

Use fold navigation commands for efficient code structure navigation and overview.

Example

```
zj  " move to next fold
zk  " move to previous fold
[z  " move to start of current fold
]z  " move to end of current fold
zv  " view cursor line (unfold if needed)
zx  " update folds
```

13.16 Register operations mastery

Category: Registers
Tags: registers, operations, advanced, clipboard

Use register operations for sophisticated copy-paste workflows and text manipulation chains.

Example

```
"ay5y    " yank 5 lines into register 'a'
"Ay3y    " append 3 lines to register 'a'
"ap      " paste contents of register 'a'
:reg a   " view contents of register 'a'
:let @a='new text'  " set register 'a' programmatically
```

13.17 Session workflow optimization

Category: Session
Tags: session, workflow, project, management

Use session commands for project-based workflow management and context switching.

Example

```
:mksession! ~/project.vim     " save current session
:source ~/project.vim         " load session
:SSave project_name           " save with plugin session manager
:SLoad project_name           " load named session
" Restore window layouts, open files, cursor positions
```

13.18 Split window mastery

Category: Windows
Tags: split, windows, mastery, layout

Use advanced window splitting and management for efficient multi-file editing and reference workflows.

Example

```
:vsplit file.txt              " vertical split
:split +/pattern file.txt     " split and search
<C-w>r                        " rotate windows
<C-w>H                        " move window to left
<C-w≥                         " equalize window sizes
<C-w>_                        " maximize window height
```

13.19 Tab-based workflow

Category: Tabs
Tags: tabs, workflow, organization, navigation

Use tabs for logical grouping of related files and context-based editing workflows.

Example

```
:tabnew file.lua              " open file in new tab
:tabonly                      " close all other tabs
gt / gT                       " navigate between tabs
<C-w>T                        " move current window to new tab
:tabmove 2                    " move tab to position 2
```

CHAPTER 14
Completion

14.1 API/Module completion

Category: Completion
Tags: completion, api, modules, vim.api

Build completion for Vim/Neovim API functions and modules.

Example

```
-- Cache API function names
local api_cache = nil

local function get_api_functions()
  if api_cache then
    return api_cache
  end

  local functions = {}

  -- Get all vim.api functions
  for name, _ in pairs(vim.api) do
    if type(vim.api[name]) == 'function' then
      table.insert(functions, {
        word = 'vim.api.' .. name,
        abbr = name,
        menu = '[API]',
        info = 'Neovim API function',
      })
    end
  end

  -- Get vim.fn functions (limited sample)
  for name, _ in pairs(vim.fn) do
    if type(vim.fn[name]) == 'function' then
      table.insert(functions, {
        word = 'vim.fn.' .. name,
        abbr = name,
        menu = '[Func]',
        info = 'Vim function',
      })
    end
  end
```

```
  api_cache = functions
  return functions
end

function ApiComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1

    -- Find start of vim.api. or vim.fn.
    while col > 0 and line:sub(col, col):match('[%w_.]') do
      col = col - 1
    end

    return col
  else
    local functions = get_api_functions()
    local matches = {}

    for _, func in ipairs(functions) do
      if func.word:find(vim.pesc(base)) then
        table.insert(matches, func)
      end
    end

    return matches
  end
end

-- Set for Lua files
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'lua',
  callback = function()
    vim.opt_local.completefunc = 'v:lua.ApiComplete'
  end,
})
```

14.2 Async completion with vim.schedule

Category: Completion
Tags: completion, async, performance, schedule

Use `vim.schedule()` to perform expensive completion operations without blocking the UI.

Example

```
local cached_items = nil
local cache_time = 0
local CACHE_DURATION = 5000  -- 5 seconds

function AsyncComplete(findstart, base)
```

```
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    local now = vim.loop.now()

    -- Use cache if fresh
    if cached_items and (now - cache_time) < CACHE_DURATION then
      return vim.tbl_filter(function(item)
        return item.word:find('^' .. vim.pesc(base))
      end, cached_items)
    end

    -- Schedule async update
    vim.schedule(function()
      -- Expensive operation (e.g., file system search, API call)
      local items = {}

      -- Example: Get all Lua functions in runtime path
      local lua_files = vim.api.nvim_get_runtime_file('**/*.lua', true)

      for _, file in ipairs(lua_files) do
        local basename = vim.fn.fnamemodify(file, ':t:r')
        table.insert(items, {
          word = basename,
          menu = '[Module]',
        })
      end

      -- Update cache
      cached_items = items
      cache_time = now
    end)

    -- Return empty or cached for now
    return cached_items or {}
  end
end

vim.opt.completefunc = 'v:lua.AsyncComplete'
```

14.3 Cached completion for performance

Category: Completion
Tags: completion, cache, performance, optimization

Implement smart caching to avoid expensive re-computation on every keystroke.

Example

```
local completion_cache = {}

local function get_cache_key(context)
  return string.format('%s:%s:%d',
    vim.bo.filetype,
    context,
    vim.b.changedtick or 0
  )
end

function CachedComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    local cache_key = get_cache_key('complete')

    -- Check cache
    if completion_cache[cache_key] then
      local cached = completion_cache[cache_key]

      -- Filter cached results by current base
      return vim.tbl_filter(function(item)
        return item.word:find('^' .. vim.pesc(base))
      end, cached)
    end

    -- Generate fresh completion items (expensive operation)
    local items = {}

    -- ... expensive computation here ...
    local all_words = {}
    for _, buf in ipairs(vim.api.nvim_list_bufs()) do
      if vim.api.nvim_buf_is_loaded(buf) then
        local lines = vim.api.nvim_buf_get_lines(buf, 0, -1, false)
        for _, line in ipairs(lines) do
          for word in line:gmatch('[%w_]+') do
            if #word > 3 then
              all_words[word] = true
            end
          end
        end
      end
    end

    for word in pairs(all_words) do
      table.insert(items, {word = word, menu = '[Cached]'})
    end

    -- Store in cache
```

```
    completion_cache[cache_key] = items

    -- Filter by base
    return vim.tbl_filter(function(item)
      return item.word:find('^' .. vim.pesc(base))
    end, items)
  end
end

-- Clear cache on significant changes
vim.api.nvim_create_autocmd({'BufWritePost', 'BufEnter'}, {
  callback = function()
    completion_cache = {}
  end,
})

vim.opt.completefunc = 'v:lua.CachedComplete'
```

14.4 Completion items with detailed info

Category: Completion
Tags: completion, items, menu, info, kind

Return completion items with additional metadata like menu text, info, and kind.

Example

```
function DetailedComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    -- Return items with metadata
    return {
      {
        word = "function",
        abbr = "func",           -- Show "func" in completion menu
        menu = "[Keyword]",       -- Category/source
        info = "Define a function", -- Preview window info
        kind = "f",              -- Single letter kind (f=function)
        icase = 1,               -- Case-insensitive matching
        dup = 1,                 -- Allow duplicates
      },
      {
        word = "variable",
        abbr = "var",
        menu = "[Keyword]",
        info = "Declare a variable",
```

```
        kind = "v",
      },
      {
        word = "return",
        abbr = "ret",
        menu = "[Keyword]",
        info = "Return from function",
        kind = "k",
      },
    }
  end
end

vim.opt.completefunc = 'v:lua.DetailedComplete'
```

14.5 Completion with external command

Category: Completion
Tags: completion, external, command, integration

Integrate external tools or commands into completion sources.

Example

```
function ExternalComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_-]') do
      col = col - 1
    end
    return col
  else
    -- Example: Git branch completion
    local handle = io.popen('git branch --list 2>/dev/null')
    if not handle then
      return {}
    end

    local result = handle:read('*a')
    handle:close()

    local matches = {}
    for branch in result:gmatch('[^\n]+') do
      -- Remove * and whitespace
      branch = branch:gsub('^%s*%*?%s*', '')

      if branch:find(vim.pesc(base)) then
        table.insert(matches, {
          word = branch,
          menu = '[Git Branch]',
        })
```

```
      end
    end

    return matches
  end
end

-- Use for git commit messages
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'gitcommit',
  callback = function()
    vim.opt_local.completefunc = 'v:lua.ExternalComplete'
  end,
})
```

14.6 Context-aware completion

Category: Completion
Tags: completion, context, smart, aware

Build completion that adapts based on cursor context like inside function, string, or comment.

Example

```
-- Get context at cursor
local function get_context()
  local node = vim.treesitter.get_node()

  while node do
    local type = node:type()

    if type == "string" then
      return "string"
    elseif type == "comment" then
      return "comment"
    elseif type == "function_declaration" or type == "function_definition"
    ↪  then
      return "function"
    end

    node = node:parent()
  end

  return "normal"
end

function ContextAwareComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
```

```
      col = col - 1
    end
    return col
  else
    local context = get_context()
    local items = {}

    if context == "string" then
      -- String-specific completions
      items = {
        {word = "format", menu = "[String Method]"},
        {word = "match", menu = "[String Method]"},
        {word = "gsub", menu = "[String Method]"},
      }
    elseif context == "function" then
      -- Function-specific completions
      items = {
        {word = "return", menu = "[Keyword]"},
        {word = "local", menu = "[Keyword]"},
        {word = "end", menu = "[Keyword]"},
      }
    else
      -- General completions
      items = {
        {word = "function", menu = "[Keyword]"},
        {word = "if", menu = "[Keyword]"},
        {word = "for", menu = "[Keyword]"},
      }
    end

    -- Filter by base
    local matches = {}
    for _, item in ipairs(items) do
      if item.word:find('^' .. vim.pesc(base)) then
        table.insert(matches, item)
      end
    end

    return matches
  end
end

vim.opt.completefunc = 'v:lua.ContextAwareComplete'
```

14.7 Custom completion function basics

Category: Completion
Tags: completion, completefunc, custom, omnifunc

Create custom completion functions using `completefunc` or `omnifunc` for domain-specific completions.

Example

```
-- Basic completion function signature
function MyCompleteFunc(findstart, base)
  if findstart == 1 then
    -- Return the column where completion starts
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1

    -- Find start of word
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end

    return col
  else
    -- Return list of matches for 'base'
    local matches = {}

    local words = {'hello', 'world', 'help', 'welcome'}
    for _, word in ipairs(words) do
      if word:find('^' .. base) then
        table.insert(matches, word)
      end
    end

    return matches
  end
end

-- Set as completion function
vim.api.nvim_create_user_command('SetMyComplete', function()
  vim.opt_local.completefunc = 'v:lua.MyCompleteFunc'
end, {})

-- Use with Ctrl-X Ctrl-U in insert mode
```

14.8 File path completion

Category: Completion
Tags: completion, files, paths, filesystem

Create custom file path completion that works in specific contexts.

Example

```
function FilePathComplete(findstart, base)
  if findstart == 1 then
    -- Find start of path
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1

    -- Look for path characters
```

```
    while col > 0 and line:sub(col, col):match('[%w_/%.~-]') do
      col = col - 1
    end

    return col
  else
    local matches = {}

    -- Expand ~ and environment variables
    base = vim.fn.expand(base)

    -- Get directory part
    local dir = vim.fn.fnamemodify(base, ':h')
    local partial = vim.fn.fnamemodify(base, ':t')

    -- Default to current directory
    if dir == base then
      dir = '.'
    end

    -- List files in directory
    local ok, files = pcall(vim.fn.readdir, dir)
    if not ok then
      return {}
    end

    for _, file in ipairs(files) do
      if file:find('^' .. vim.pesc(partial)) then
        local full_path = dir .. '/' .. file

        -- Check if directory
        local is_dir = vim.fn.isdirectory(full_path) == 1

        table.insert(matches, {
          word = file .. (is_dir and '/' or ''),
          menu = is_dir and '[Dir]' or '[File]',
          kind = is_dir and 'd' or 'f',
        })
      end
    end

    return matches
  end
end

-- Use for specific filetypes
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'lua', 'vim'},
  callback = function()
    vim.opt_local.completefunc = 'v:lua.FilePathComplete'
  end,
})
```

14.9 Fuzzy matching completion

Category: Completion
Tags: completion, fuzzy, matching, filter

Implement fuzzy matching for completion items for more flexible matching.

Example

```
-- Simple fuzzy match function
local function fuzzy_match(str, pattern)
  local pattern_idx = 1
  local str_idx = 1

  while pattern_idx ⩽ #pattern and str_idx ⩽ #str do
    if str:sub(str_idx, str_idx):lower() == pattern:sub(pattern_idx,
    ↪  pattern_idx):lower() then
      pattern_idx = pattern_idx + 1
    end
    str_idx = str_idx + 1
  end

  return pattern_idx > #pattern
end

-- Score fuzzy match (consecutive chars score higher)
local function fuzzy_score(str, pattern)
  if not fuzzy_match(str, pattern) then
    return 0
  end

  local score = 0
  local consecutive = 0

  local pattern_idx = 1
  for str_idx = 1, #str do
    if str:sub(str_idx, str_idx):lower() == pattern:sub(pattern_idx,
    ↪  pattern_idx):lower() then
      score = score + 1 + consecutive
      consecutive = consecutive + 1
      pattern_idx = pattern_idx + 1

      if pattern_idx > #pattern then
        break
      end
    else
      consecutive = 0
    end
  end

  return score
end

function FuzzyComplete(findstart, base)
  if findstart == 1 then
```

```
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    local all_items = {
      'apple', 'application', 'apply',
      'banana', 'band', 'background',
      'cherry', 'check', 'character',
    }

    -- Fuzzy match and score
    local matches = {}
    for _, word in ipairs(all_items) do
      local score = fuzzy_score(word, base)
      if score > 0 then
        table.insert(matches, {
          word = word,
          score = score,
        })
      end
    end

    -- Sort by score (highest first)
    table.sort(matches, function(a, b)
      return a.score > b.score
    end)

    return matches
  end
end

vim.opt.completefunc = 'v:lua.FuzzyComplete'
```

14.10 LSP-style completion with documentation

Category: Completion
Tags: completion, lsp, documentation, preview

Create rich completion items with documentation preview like LSP.

Example

```
local completion_items = {
  {
    label = "print",
    kind = "Function",
    detail = "print(...)",
    documentation = {
      kind = "markdown",
```

```
      value = [[
Print arguments to stdout.

**Parameters:**
- `...` - Values to print

**Example:**
```

lua print(”Hello”, ”World”) – Output: Hello World

Example

```
      ]],
    },
  },
  {
    label = "require",
    kind = "Function",
    detail = "require(modname)",
    documentation = {
      kind = "markdown",
      value = [[
Load and return a module.

**Parameters:**
- `modname` (string) - Module name

**Returns:**
- Module table

**Example:**
```

lua local mymod = require(’mymodule’)

Example

```
      ]],
    },
  },
}

function LspStyleComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    local matches = {}

    for _, item in ipairs(completion_items) do
```

```
      if item.label:find('^' .. vim.pesc(base)) then
        table.insert(matches, {
          word = item.label,
          abbr = item.label,
          menu = string.format('[%s]', item.kind),
          info = item.documentation.value,
          kind = item.kind:sub(1, 1):lower(),
        })
      end
    end

    return matches
  end
end

vim.opt.completefunc = 'v:lua.LspStyleComplete'

-- Enable preview window for completion
vim.opt.completeopt = {'menu', 'menuone', 'preview'}
```

14.11 Multi-source completion chaining

Category: Completion
Tags: completion, chaining, multiple-sources, aggregation

Combine multiple completion sources into a single completion function.

Example

```
-- Define multiple completion sources
local sources = {}

sources.keywords = function(base)
  local keywords = {'function', 'if', 'then', 'else', 'end', 'return',
  ↪  'local'}
  local matches = {}

  for _, kw in ipairs(keywords) do
    if kw:find('^' .. vim.pesc(base)) then
      table.insert(matches, {
        word = kw,
        menu = '[Keyword]',
        priority = 10,
      })
    end
  end

  return matches
end

sources.buffer_words = function(base)
  -- Get words from current buffer
```

```
  local words = {}
  local lines = vim.api.nvim_buf_get_lines(0, 0, -1, false)
  local text = table.concat(lines, '\n')

  for word in text:gmatch('[%w_]+') do
    if word:find('^' .. vim.pesc(base)) and #word > 3 then
      words[word] = true
    end
  end

  local matches = {}
  for word in pairs(words) do
    table.insert(matches, {
      word = word,
      menu = '[Buffer]',
      priority = 5,
    })
  end

  return matches
end

sources.api = function(base)
  if not base:match('^vim%.') then
    return {}
  end

  local matches = {}
  for name in pairs(vim.api) do
    local full = 'vim.api.' .. name
    if full:find(vim.pesc(base)) then
      table.insert(matches, {
        word = full,
        menu = '[API]',
        priority = 15,
      })
    end
  end

  return matches
end

function ChainedComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_.]') do
      col = col - 1
    end
    return col
  else
    local all_matches = {}

    -- Gather from all sources
    for name, source_func in pairs(sources) do
      local items = source_func(base)
```

```
      for _, item in ipairs(items) do
        table.insert(all_matches, item)
      end
    end

    -- Sort by priority (higher first)
    table.sort(all_matches, function(a, b)
      return (a.priority or 0) > (b.priority or 0)
    end)

    return all_matches
  end
end

vim.opt.completefunc = 'v:lua.ChainedComplete'
```

14.12 Snippet-style completion

Category: Completion
Tags: completion, snippets, templates, user-data

Create completion items that insert templates using `user_data` for custom behavior.

Example

```
function SnippetComplete(findstart, base)
  if findstart == 1 then
    local line = vim.api.nvim_get_current_line()
    local col = vim.fn.col('.') - 1
    while col > 0 and line:sub(col, col):match('[%w_]') do
      col = col - 1
    end
    return col
  else
    return {
      {
        word = "function",
        abbr = "func",
        menu = "[Snippet]",
        info = "Function template",
        user_data = {
          snippet = "function ${1:name}(${2:args})\n\t${0}\nend"
        }
      },
      {
        word = "if",
        abbr = "if",
        menu = "[Snippet]",
        info = "If statement",
        user_data = {
          snippet = "if ${1:condition} then\n\t${0}\nend"
        }
```

```
      },
      {
        word = "for",
        abbr = "for",
        menu = "[Snippet]",
        info = "For loop",
        user_data = {
          snippet = "for ${1:i}, ${2:v} in ipairs(${3:table})
          ↪  do\n\t${0}\nend"
        }
      },
    }
  end
end

-- Handle completion selection
vim.api.nvim_create_autocmd('CompleteDone', {
  callback = function()
    local completed_item = vim.v.completed_item

    if completed_item.user_data and
       type(completed_item.user_data) == 'table' and
       completed_item.user_data.snippet then

      -- Simple snippet expansion (replace ${n:text} with text)
      local snippet = completed_item.user_data.snippet
      local text = snippet:gsub('%${%d+:([^}]+)}', '%1')
      text = text:gsub('%${%d+}', '')

      -- Delete the completion word
      vim.cmd('normal! diw')

      -- Insert snippet
      local lines = vim.split(text, '\n')
      vim.api.nvim_put(lines, 'c', true, true)
    end
  end,
})

vim.opt.completefunc = 'v:lua.SnippetComplete'
```

CHAPTER 15
Configuration

15.1 Alternate Neovim startup configuration

Category: Configuration
Tags: startup, config, alternate, minimal, debug

Start Neovim with alternate configuration using `-u` flag for testing or minimal setups.

Example

```
" Start with minimal config:
nvim -u ~/.config/nvim/minimal.lua

" Start with no config:
nvim -u NONE

" Start with specific vimrc:
nvim -u ~/.vimrc.test
```

15.2 Append to option value

Category: Configuration
Tags: set, option, append

Use `:set option+=value` to append a value to an option.

Example

```
:set path+=./include  " add to search path
:set wildignore+=*.pyc  " ignore Python bytecode
```

15.3 Auto tab completion

Category: Configuration
Tags: completion, tab, autocomplete

Configure TAB to autocomplete words while preserving normal TAB functionality.

Example

```
function! Tab_Or_Complete()
  if col('.')>1 && strpart( getline('.'), col('.')-2, 3 ) =~ '^\w'
    return "\<C-N>"
  else
    return "\<Tab>"
  endif
endfunction
inoremap <Tab> <C-R>≥Tab_Or_Complete()<CR>
set dictionary="/usr/dict/words"
```

15.4 Auto-reload file changes

Category: Configuration
Tags: auto, reload, file, changes

Automatically reload file when it changes externally, with optional warning.

Example

```
set autoread
" Trigger autoread when cursor stops moving
au FocusGained,BufEnter * :silent! !
au FocusLost,WinLeave * :silent! w
" Or check periodically
au CursorHold * :silent! checktime
```

15.5 Check plugin key mapping usage

Category: Configuration
Tags: plugin, mapping, check, usage, debug

Use `echo maparg("key", "mode")` to check what key mapping is assigned in specific mode.

Example

```
:echo maparg("S", "v")      " check visual mode 'S' mapping
:echo maparg("<leader>f", "n") " check normal mode leader+f mapping
:echo maparg("<C-n>", "i")  " check insert mode Ctrl+n mapping
```

15.6 Enable 256 colors

Category: Configuration
Tags: colors, terminal, display

Configure terminal to support 256 colors with proper settings.

Example

```
set t_Co=256
set t_AB=^[[48;5;%dm
set t_AF=^[[38;5;%dm
" In shell profile:
export TERM='xterm-256color'
```

15.7 Environment variables in configuration

Category: Configuration
Tags: environment, variable, conditional, config, lua

Use `os.getenv()` in Lua configuration to conditionally set options based on environment variables.

Example

```
-- In init.lua:
if os.getenv("MACHINE") == "work" then
  -- Work-specific configuration
  vim.opt.colorcolumn = "80"
else
  -- Personal configuration
  vim.opt.colorcolumn = "120"
end
```

15.8 Ex commands - autocmds and events

Category: Configuration
Tags: ex, autocmd, event, pattern, command

Use `:autocmd` to set up automatic commands, `:autocmd!` to clear, `:doautocmd` to trigger events.

Example

```
:autocmd BufWritePost *.py !python %  " run python after save
:autocmd! BufRead       " clear all BufRead autocmds
:doautocmd BufRead      " trigger BufRead event
:autocmd FileType python setlocal ts=4  " Python-specific settings
```

15.9 Ex commands - highlight and syntax

Category: Configuration
Tags: ex, highlight, syntax, color, group

Use `:highlight` to set colors, `:syntax` for syntax highlighting, `:colorscheme` to change themes.

Example

```
:highlight Comment ctermfg=green   " set comment color
:syntax on                         " enable syntax highlighting
:syntax off                        " disable syntax highlighting
:colorscheme desert                " change color scheme
:highlight clear                   " clear all highlighting
```

15.10 Ex commands - mappings and abbreviations

Category: Configuration
Tags: ex, map, abbrev, shortcut, key

Use `:map` for mappings, `:abbrev` for abbreviations, `:unmap` and `:unabbrev` to remove.

Example

```
:map <F2> :w<CR>        " map F2 to save
:imap <F3> <Esc>:w<CR>  " insert mode mapping
:abbrev teh the         " abbreviation for typo
:unmap <F2>             " remove mapping
:unabbrev teh           " remove abbreviation
```

15.11 Ex commands - option with values

Category: Configuration
Tags: ex, set, value, assignment, string

Use `:set option=value` to assign value, `:set option+=value` to append, `:set option-=value` to remove.

Example

```
:set tabstop=4        " set tab width to 4
:set path+=/usr/include  " add to path
:set path-=/tmp       " remove from path
:set suffixes+=.bak   " add .bak to suffixes
```

15.12 Ex commands - runtime and sourcing

Category: Configuration
Tags: ex, source, runtime, script, load

Use `:source` to load script, `:runtime` to load from runtime path, `:scriptnames` to list loaded scripts.

Example

```
:source ~/.vimrc        " load configuration file
:runtime! plugin/**/*.vim  " load all plugins
:scriptnames            " list all loaded scripts
:source %               " reload current file as script
```

15.13 Ex commands - set options

Category: Configuration
Tags: ex, set, option, toggle, query

Use `:set option` to enable, `:set nooption` to disable, `:set option?` to query, `:set option&` to reset to default.

Example

```
:set number        " enable line numbers
:set nonumber      " disable line numbers
:set number?       " check if line numbers are enabled
:set number&       " reset to default value
```

15.14 Execute command with pipe separator

Category: Configuration
Tags: execute, command, pipe, separator, multiple

Use `:execute` to allow `|` pipe character to separate multiple commands in mappings.

Example

```
" Without execute, | ends the mapping:
nnoremap <F5> :w | echo "Saved"<CR>  " Wrong - | ends mapping

" With execute, | separates commands:
nnoremap <F5> :execute "w \| echo 'Saved'"<CR>  " Correct
```

15.15 Hidden buffers option

Category: Configuration
Tags: hidden, buffer, switch, unsaved, edit

Use `:set hidden` to allow switching between files without saving changes, preventing "No write since last change" errors.

Example

```
:set hidden        " allow unsaved buffer switching
:set nohidden      " require saving before switching (default)
" Now you can use :edit, :next, etc. without saving first
```

15.16 Home key smart mapping

Category: Configuration
Tags: home, key, mapping, smart, navigation

Map Home key to toggle between beginning of line and first non-blank character.

Example

```
" Smart Home key mapping:
nnoremap <expr> <Home> (col('.') == 1 ? '^' : '0')
inoremap <expr> <Home> (col('.') == 1 ? '<C-o>^' : '<C-o>0')

" Alternative version:
nnoremap <silent> <Home> :call SmartHome()<CR>
function! SmartHome()
  let curcol = col('.')
  normal! ^
  if col('.') == curcol
    normal! 0
  endif
endfunction
```

15.17 Markdown code block syntax highlighting

Category: Configuration
Tags: markdown, syntax, highlighting, fenced, languages

Configure syntax highlighting for fenced code blocks in markdown files by setting supported languages.

Example

```
-- In init.lua
vim.g.markdown_fenced_languages = {
  "html",
  "javascript",
  "typescript",
  "css",
  "scss",
  "lua",
  "vim",
  "python",
  "bash"
}
```

15.18 Per-project configuration with .nvim.lua

Category: Configuration
Tags: project, config, local, exrc, security

Enable per-project configuration by creating `.nvim.lua` files in project directories. This allows project-specific settings without security risks of `.exrc`.

Example

```
-- In init.lua, enable exrc option:
vim.opt.exrc = true

-- Create .nvim.lua in project root:
-- .nvim.lua
vim.opt.tabstop = 2
vim.opt.shiftwidth = 2
vim.opt.colorcolumn = "80"

-- Project-specific LSP settings
vim.lsp.start({
  name = "project-lsp",
  cmd = {"my-lsp-server"},
  root_dir = vim.fs.dirname(vim.fs.find({"package.json"}, { upward = true
  ↪  })[1])
})
```

15.19 Remove from option value

Category: Configuration
Tags: set, option, remove

Use `:set option-=value` to remove a value from an option.

Example

```
:set path-=./include  " remove from search path
:set wildignore-=*.pyc  " stop ignoring Python bytecode
```

15.20 Restore cursor position

Category: Configuration
Tags: cursor, position, session, restore

Automatically restore cursor position when reopening files.

Example

```
function! ResCur()
  if line("'\"") <= line("$")
    normal! g`"
    return 1
  endif
endfunction

augroup resCur
  autocmd!
  autocmd BufWinEnter * call ResCur()
augroup END

" Enable viminfo
set viminfo='10,\"100,:20,%,n~/.viminfo
```

15.21 Sandbox mode for safe testing

Category: Configuration
Tags: sandbox, safe, testing, command, :sandbox

Use `:sandbox` to execute commands safely without side effects like persistent undo entries or autocommands.

Example

```
:sandbox set number           " test setting without permanent change
:sandbox echo expand('%')     " safely test expressions
:sandbox source unsafe.vim    " test configuration safely
```

15.22 Set color scheme based on time

Category: Configuration
Tags: color, scheme, time, automatic

Automatically switch between light and dark color schemes based on time of day.

Example

```
if strftime("%H") < 18 && strftime("%H") > 6
  colorscheme morning
else
  colorscheme evening
endif
```

15.23 Speed up vimgrep with noautocmd

Category: Configuration
Tags: vimgrep, speed, autocmd, performance, search

Use `:noautocmd vimgrep` to speed up vimgrep by disabling autocmds during search.

Example

```
:noautocmd vimgrep /pattern/ **/*.txt  " faster vimgrep
:noautocmd bufdo %s/old/new/ge         " faster buffer operations
```

15.24 Toggle paste mode

Category: Configuration
Tags: paste, toggle, indent, clipboard

Set up paste toggle to prevent auto-indenting when pasting from clipboard in terminal.

Example

```
set pastetoggle=<F2>
nnoremap <F2> :set invpaste paste?<CR>
set showmode
" Use F2 before and after pasting external text
```

15.25 Verbose mapping information

Category: Configuration
Tags: verbose, mapping, script, source, debug

Use `:verbose map <key>` to see which script defined a mapping and where.

Example

```
:verbose map <F1>     " show where F1 mapping was defined
:verbose imap <Tab>   " show insert mode Tab mapping source
:verbose map          " show all mappings with sources
```

15.26 View runtime paths

Category: Configuration
Tags: runtime, path, debug

Use `:echo &runtimepath` to see all runtime paths Neovim is using.

Example

```
:echo &runtimepath  " show runtime paths
```

CHAPTER 16

Cut and paste

16.1 Cut/delete word

Category: Cut and Paste
Tags: cut, delete, word

Use `dw` to delete from cursor to start of next word, `de` to delete to end of current word, or `db` to delete to start of current word.

Example

```
dw  " delete to next word
de  " delete to end of word
db  " delete to start of word
```

16.2 Paste text

Category: Cut and Paste
Tags: paste, put, text

Use `p` to paste after cursor/line and `P` to paste before cursor/line.

Example

```
p  " paste after
P  " paste before
```

16.3 Paste with automatic indentation

Category: Cut and Paste
Tags: paste, indent, automatic

Use `[p` and `[P` to paste and automatically adjust indentation to match current line.

Example

```
[p  " paste after with auto-indent
[P  " paste before with auto-indent
]p  " paste after with auto-indent
]P  " same as [P
```

16.4 Yank line

Category: Cut and Paste
Tags: yank, copy, line

Use `yy` to yank (copy) the current line, or `{number}yy` to yank multiple lines.

Example

```
yy   " yank current line
3yy  " yank 3 lines
```

16.5 Yank word

Category: Cut and Paste
Tags: yank, copy, word

Use `yw` to yank from cursor to start of next word, `ye` to yank to end of current word.

Example

```
yw  " yank to next word
ye  " yank to end of word
```

CHAPTER 17

Diagnostics

17.1 Configure diagnostic display with virtual text

Category: Diagnostics
Tags: lsp, diagnostic, virtual-text, configuration, display

Configure how LSP diagnostics are displayed using virtual text, virtual lines, or floating windows.

Example

```
-- In init.lua, configure diagnostic display:
vim.diagnostic.config({
  virtual_text = {
    prefix = '●',  -- Could be '■', '▎', etc.
    source = "if_many",  -- Show source if multiple sources
  },
  signs = true,
  underline = true,
  update_in_insert = false,
  severity_sort = true,
  float = {
    border = 'rounded',
    source = 'always',
    header = '',
    prefix = '',
  },
})

-- Disable virtual text, use virtual lines instead:
vim.diagnostic.config({
  virtual_text = false,
  virtual_lines = { only_current_line = true }
})

-- Toggle diagnostics on/off:
vim.keymap.set('n', '<leader>dt', function()
  vim.diagnostic.enable(not vim.diagnostic.is_enabled())
end, { desc = 'Toggle diagnostics' })
```

17.2 Find mapping source

Category: Diagnostics
Tags: mapping, verbose, source

Use `:verbose map <key>` to see where a specific mapping was defined.

Example

```
:verbose map <leader>f  " see where <leader>f was mapped
```

17.3 Find option source

Category: Diagnostics
Tags: option, verbose, source

Use `:verbose set option?` to see where a specific option was last set.

Example

```
:verbose set number?  " see where 'number' option was set
```

17.4 Health diagnostics

Category: Diagnostics
Tags: health, check, diagnostics

Use `:checkhealth` to run health diagnostics for your Neovim setup.

Example

```
:checkhealth  " run health diagnostics
```

17.5 View messages

Category: Diagnostics
Tags: messages, log, history

Use `:messages` to view past messages and notifications.

Example

```
:messages  " view past messages
```

CHAPTER 18

Display

18.1 Conceal text with syntax highlighting

Category: Display
Tags: conceal, hide, text, syntax, conceallevel

Use `:set conceallevel=2` to hide concealed text and `:syntax match` with `conceal` to define what to hide.

Example

```
:set conceallevel=2       " hide concealed text completely
:set conceallevel=0       " show all text normally
:syntax match htmlTag '<[^>]*>' conceal  " hide HTML tags
" Toggle conceal on/off
nnoremap <leader>c :let &conceallevel = (&conceallevel == 2) ? 0 : 2<CR>
```

18.2 Ex commands - display and UI settings

Category: Display
Tags: ex, display, ui, show, list

Use `:set list` to show whitespace, `:set wrap` for line wrapping, `:set ruler` for cursor position, `:set showcmd` for command display.

Example

```
:set list         " show whitespace characters
:set listchars=tab:>-,trail:·  " customize whitespace display
:set wrap         " enable line wrapping
:set nowrap       " disable line wrapping
:set ruler        " show cursor position
:set showcmd      " show partial commands
```

18.3 Ex commands - folding display

Category: Display
Tags: ex, fold, display, column, text

Use `:set foldcolumn` to show fold column, `:set foldtext` for custom fold text, `:set fillchars` for fill characters.

Example

```
:set foldcolumn=4     " show fold indicators in 4-char column
:set fillchars=fold:.,vert:|  " customize fill characters
:set foldtext=MyFoldText()    " custom fold text function
```

18.4 Ex commands - line numbers and columns

Category: Display
Tags: ex, line, number, column, relative

Use `:set number` for line numbers, `:set relativenumber` for relative numbers, `:set colorcolumn` for guide column.

Example

```
:set number           " show line numbers
:set relativenumber   " show relative line numbers
:set number relativenumber  " show both
:set colorcolumn=80   " highlight column 80
:set textwidth=72     " set text width
```

18.5 Ex commands - scrolling and viewport

Category: Display
Tags: ex, scroll, viewport, offset, bind

Use `:set scrolloff` for scroll offset, `:set sidescrolloff` for horizontal offset, `:set scrollbind` to bind scrolling.

Example

```
:set scrolloff=5      " keep 5 lines above/below cursor
:set sidescrolloff=8  " keep 8 columns left/right of cursor
:set scrollbind       " bind scrolling between windows
:set noscrollbind     " unbind scrolling
```

18.6 Ex commands - status line and tabs

Category: Display
Tags: ex, status, line, tab, label

Use `:set laststatus` for status line, `:set showtabline` for tab line, `:set statusline` for

custom status.

Example

```
:set laststatus=2     " always show status line
:set showtabline=2    " always show tab line
:set statusline=%f\ %m%r%h%w\ [%Y]\ [%{&ff}]\ %=%l,%c\ %p%%
```

18.7 Toggle cursor line highlight

Category: Display
Tags: cursorline, highlight, toggle

Use `:set cursorline!` to toggle highlighting of the current cursor line.

Example

```
:set cursorline!  " toggle cursor line highlight
```

18.8 Toggle invisible characters

Category: Display
Tags: invisible, characters, toggle

Use `:set list!` to toggle display of invisible characters (tabs, spaces, etc.).

Example

```
:set list!  " toggle invisible characters
```

CHAPTER 19
Edit

19.1 Adding prefix/suffix to multiline text easily

Category: Edit
Tags: text, object, advanced

Suppose that you have multiple lines of text. You want to put `<b>` and `</b>` tags around each line:

- Position the cursor in normal mode over the first char of the frist line - Enter visual block mode: `<C-v>` - Select all first characters in all lines under the first one by using normal cursor keys - Switch to insert mode: `I` - Start changing the first line by typing `<b>` - When you are done, press `<ESC>` and all lines will have the same `<b>` prefix. - Now let's add `</b>` to the end of each line. Press `gv` - the first column gets selected. - Press `$` to go to the end of the line. - Now type `A` to append to the line - The cursor goes to the end of the top line. Enter the closing tag `</b>`. - Now pres `ESC` to leave the insert mode and you are done!

Credits: [Henry Misc](https://www.youtube.com/watch?v=RdyfT2dbt78https://www.youtube.com

19.2 Common operators with motions

Category: Edit
Tags: operator, motion, combination, examples

Operators like `d` (delete), `c` (change), `y` (yank), `>` (indent) work with any motion or text object.

Example

```
dw    " delete word
c3j   " change 3 lines down
y$    " yank to end of line
>i{   " indent inside braces
=ap   " format around paragraph
```

19.3 Operator-pending mode - cancel operations

Category: Edit
Tags: operator, pending, cancel, abort, undo

Use `Esc` to abandon the operator or `Backspace` to undo/cancel the operator in pending mode.

Example

```
d<Esc>          " abandon delete operation
c<Backspace>    " cancel change operation
```

19.4 Operator-pending mode - force operation type

Category: Edit
Tags: operator, pending, force, characterwise, linewise, blockwise

Use `v` for characterwise, `V` for linewise, `Ctrl+v` for blockwise operation after an operator.

Example

```
dvw   " force characterwise delete word
dV    " force linewise delete (whole line)
dCtrl+v}  " force blockwise delete to closing brace
```

19.5 Operator-pending mode basics

Category: Edit
Tags: operator, pending, mode, motion

After typing an operator (d, c, y, etc.), you enter operator-pending mode where you can provide motion or text object to complete the operation.

Example

```
d    " delete operator (enters pending mode)
dw   " delete word (operator + motion)
ci(  " change inside parentheses (operator + text object)
```

19.6 Operator-pending mode with text objects

Category: Edit
Tags: operator, text, object, combination

All text objects (iw, aw, i(, a(, ip, ap, etc.) work in operator-pending mode for precise text manipulation.

Example

```
ciw   " change inside word
da(   " delete around parentheses
yi"   " yank inside quotes
>ap   " indent around paragraph
=i{   " format inside braces
```

19.7 Redraw screen

Category: Edit
Tags: redraw, screen, refresh, display

Use `Ctrl+l` to redraw the screen, useful when display gets corrupted or needs refreshing.

Example

```
Ctrl+l  " redraw screen
```

19.8 Repeat last change

Category: Edit
Tags: repeat, change, command

Use `.` (dot) to repeat the last change command.

Example

```
.  " repeat last change
```

19.9 Show file information

Category: Edit
Tags: file, info, position, status

Use `Ctrl+g` to display current file name, cursor position, and buffer information.

Example

```
Ctrl+g  " show file info and cursor position
```

19.10 Substitute characters

Category: Edit
Tags: substitute, character, delete, insert

Use `s` to substitute (delete character under cursor and enter insert mode) and `S` to substitute entire line.

Example

```
s   " substitute character under cursor
S   " substitute entire line
5s  " substitute 5 characters
```

19.11 Time-based undo navigation

Category: Edit
Tags: undo, time, history

Use `:earlier 10m` to revert buffer to state 10 minutes ago, or `:later 5m` to go forward 5 minutes.

Example

```
:earlier 10m  " revert to 10 minutes ago
:later 5m     " go forward 5 minutes
```

CHAPTER 20
Editing

20.1 Advanced sort options

Category: Editing
Tags: sort, advanced, numeric, reverse, case

Use advanced `:sort` options for more complex sorting needs like numeric sorting, reverse sorting, and case-insensitive sorting.

Example

```
:sort!        " reverse sort (Z to A)
:sort i       " case-insensitive sort
:sort n       " numeric sort (by first number in line)
:sort u       " unique - sort and remove duplicates
:sort! n      " reverse numeric sort
:sort f       " float sort (for decimal numbers)

" Combine options:
:sort in      " case-insensitive numeric sort
:sort! u      " reverse sort and remove duplicates
```

20.2 Calculate expressions

Category: Editing
Tags: calculate, math, expression, replace

Use `<C-r>=` in insert mode to calculate mathematical expressions and insert the result.

Example

```
" In insert mode:
<C-r>=2+2<CR>       " inserts '4'
<C-r>=16*1024<CR>   " inserts '16384'
```

20.3 Capitalize words easily

Category: Editing
Tags: capitalize, words, case, format

Use `guw~` to make word lowercase then capitalize first letter, or create mapping for title case.

Example

```
guw~        " lowercase word then capitalize first letter
" Or map for convenience:
nnoremap <leader>tc guw~
```

20.4 Copy and move lines to marks

Category: Editing
Tags: copy, move, mark, line, range

Use `:t` to copy lines to marks, `:.t'a` to copy current line to mark 'a', `:152,154t.` to copy range to current position.

Example

```
ma           " set mark 'a' at current line
:.t'a        " copy current line to mark 'a'
:5,10t'b     " copy lines 5-10 to mark 'b'
:'<,'>t'a    " copy visual selection to mark 'a'
:152,154t.   " copy lines 152-154 to current position
```

20.5 Delete words in different way

Category: Editing
Tags: delete, word, alternative, whitespace

Use `daw` to delete word including surrounding whitespace, `diw` for word only, `dW` for WORD including punctuation.

Example

```
daw  " delete a word (including spaces)
diw  " delete inner word (no spaces)
dW   " delete WORD (including punctuation)
daW  " delete a WORD (including spaces)
```

20.6 Edit file at specific line

Category: Editing
Tags: file, line, open, position, jump

Use `:edit +{line} {file}` to open file and jump directly to specified line number.

Example

```
:edit +25 config.vim  " open config.vim at line 25
:edit +/pattern file.txt  " open file.txt at first line matching pattern
:edit +$ log.txt      " open log.txt at last line
vim +42 file.txt      " from command line: open at line 42
```

20.7 Enhanced undo and redo

Category: Editing
Tags: undo, redo, changes, history, time

Use advanced undo/redo with time-based navigation and undo tree features.

Example

```
u           " undo last change
<C-r>       " redo last undone change
U           " restore line to original state
:earlier 5m " undo changes from 5 minutes ago
:later 10m  " redo changes up to 10 minutes later

" Navigate undo tree
:undolist   " show undo history
g-          " go to older text state
g+          " go to newer text state
```

20.8 Ex commands - joining and splitting

Category: Editing
Tags: ex, join, split, lines, combine

Use `:join` or `:j` to join lines, with count to join multiple lines.

Example

```
:join         " join current line with next
:j            " short form of join
:5,8join      " join lines 5-8
:join!        " join without inserting spaces
```

20.9 Ex commands - line operations

Category: Editing
Tags: ex, line, delete, copy, move, range

Use `:d` to delete lines, `:y` to yank, `:m` to move, `:co` or `:t` to copy, with ranges like `1,5` or `%`.

Example

```
:5d           " delete line 5
:1,10d        " delete lines 1-10
:%d           " delete all lines
:%delete      " same as above
ggdG          " same as above
:5,10m 20     " move lines 5-10 to after line 20
:1,5co 10     " copy lines 1-5 to after line 10
```

20.10 Ex commands - marks and jumps

Category: Editing
Tags: ex, marks, jump, position, navigate

Use `:mark` to set mark, `:jumps` to show jump list, `:changes` for change list, `:delmarks` to delete marks.

Example

```
:mark a       " set mark 'a' at current line
:marks        " show all marks
:jumps        " show jump list
:changes      " show change list
:delmarks a   " delete mark 'a'
:delmarks!    " delete all lowercase marks
```

20.11 Ex commands - sorting and formatting

Category: Editing
Tags: ex, sort, format, center, left, right

Use `:sort` to sort lines, `:center` to center text, `:left` and `:right` for alignment.

Example

```
:%sort        " sort all lines
:5,15sort     " sort lines 5-15
:sort u       " sort and remove duplicates
:center 80    " center text in 80 columns
:left 5       " left align with 5 space indent
```

```
:right 70     " right align to column 70
```

20.12 Ex commands - undo and redo

Category: Editing
Tags: ex, undo, redo, earlier, later

Use `:undo` and `:redo` for undo/redo, `:earlier` and `:later` for time-based undo.

Example

```
:undo         " undo last change
:redo         " redo last undone change
:earlier 10m  " go back 10 minutes
:later 5s     " go forward 5 seconds
:earlier 10f  " go back 10 file states
```

20.13 Execute normal commands without mappings

Category: Editing
Tags: normal, command, mapping, script, execute

Use `normal!` in scripts to execute normal-mode commands without triggering user mappings.

Example

```
" In a script or function:
normal! dd       " delete line without triggering dd mapping
normal! yy       " yank line without triggering yy mapping
execute "normal! \<C-v>j"  " block select down
```

20.14 Global command with normal mode operations

Category: Editing
Tags: global, normal, command, pattern, batch

Use `:g/pattern/ normal {commands}` to execute normal mode commands on all matching lines.

Example

```
:g/console.log/ normal gcc    " comment all lines with 'console.log'
:g/TODO/ normal dw            " delete first word on lines with 'TODO'
:g/function/ normal >>        " indent all lines containing 'function'
```

20.15 Increment search results

Category: Editing
Tags: increment, search, replace, counter, sequential

Use global command with counter to incrementally replace search results with sequential numbers.

Example

```
" Replace all '2.gif' with incremental numbers:
:let idx=0 | g/2\.gif/ let idx += 1 | s//\= idx . '.gif'/

" Replace 'item' with numbered items:
:let n=1 | g/item/ s//\='item' . n/ | let n=n+1
```

20.16 Insert at beginning/end

Category: Editing
Tags: insert, beginning, end

Use `I` to insert at beginning of line, `A` to append at end of line.

Example

```
I  " insert at line start
A  " append at line end
```

20.17 Insert multiple lines

Category: Editing
Tags: insert, lines, multiple, batch

Use `o<Esc>` followed by repeat count, or `{count}o` to insert multiple empty lines at once.

Example

```
5o<Esc>     " insert 5 empty lines below
5O<Esc>     " insert 5 empty lines above
" Or in normal mode:
o<Esc>4.    " insert line, then repeat 4 times
```

20.18 Insert newline without entering insert mode

Category: Editing
Tags: newline, insert, normal, mode

Use `o<Esc>` to insert line below or `O<Esc>` to insert line above without staying in insert mode.

Example

```
o<Esc>      " insert empty line below, stay in normal mode
O<Esc>      " insert empty line above, stay in normal mode
" Or map for convenience:
nnoremap <leader>o o<Esc>
nnoremap <leader>O O<Esc>
```

20.19 Insert single character

Category: Editing
Tags: insert, character, single, quick

Use `i{char}<Esc>` or create mapping with `s` to quickly insert single character without staying in insert mode.

Example

```
" Insert single character and return to normal mode
nnoremap <leader>i i_<Esc>r
" Or use s to substitute character:
s{char}<Esc>  " replace character under cursor
```

20.20 Move line to end of paragraph

Category: Editing
Tags: move, line, paragraph, end, motion

Use `:m'}-1` to move current line to end of current paragraph.

Example

```
:m'}-1        " move current line to end of paragraph
:m'}          " move current line after end of paragraph
:m'{-1        " move current line to start of paragraph
```

20.21 Move lines to marks

Category: Editing
Tags: move, marks, line, navigation

Use `:m'a` to move current line to mark 'a', or `:.m'b` to move current line to mark 'b'. Useful when target is not visible on screen.

Example

```
ma      " mark current line as 'a'
:.m'a   " move current line to mark 'a'
:5m'b   " move line 5 to mark 'b'
```

20.22 Number lines with commands

Category: Editing
Tags: number, line, sequence, increment, script

Add line numbers to text using substitute command with expression.

Example

```
:%s/^/\=line('.') . '. '/  " add line numbers with dots
:%s/^/\=printf("%3d: ", line('.'))/ " formatted line numbers
:let i=1 | g/^/s//\=i . '. '/ | let i=i+1  " alternative method
```

20.23 Omni completion setup

Category: Editing
Tags: completion, omni, smart, autocomplete

Configure and use intelligent omni completion for programming languages.

Example

```
" Enable omni completion
filetype plugin on
set omnifunc=syntaxcomplete#Complete

" Use omni completion
<C-x><C-o>        " trigger omni completion in insert mode
<C-n>             " navigate completion menu down
<C-p>             " navigate completion menu up

" Enable for specific languages
autocmd FileType python setlocal omnifunc=pythoncomplete#Complete
autocmd FileType javascript setlocal omnifunc=javascriptcomplete#CompleteJS
```

20.24 Open new line

Category: Editing
Tags: open, newline, insert

Use `o` to open new line below cursor, `O` to open new line above cursor.

Example

```
o  " open line below
O  " open line above
```

20.25 Put (paste) operations

Category: Editing
Tags: put, paste, clipboard

Use `p` to paste after cursor, `P` to paste before cursor.

Example

```
p  " paste after cursor
P  " paste before cursor
```

20.26 Put text above or below current line

Category: Editing
Tags: put, paste, above, below, line

Use `:pu` to paste below current line, `:pu!` to paste above current line, regardless of cursor position.

Example

```
:pu         " paste register contents below current line
:pu!        " paste register contents above current line
:pu a       " paste register 'a' below current line
:pu! a      " paste register 'a' above current line
```

20.27 Return to last exit position

Category: Editing
Tags: position, exit, return, mark, jump

Use mark `"0` to jump to position where Vim was last exited from current file.

Example

```
`"0     " jump to last exit position
'"0     " jump to last exit position (line start)
:normal `"0  " execute from script/mapping
```

20.28 Select non-uniform strings across lines

Category: Editing
Tags: select, yank, append, register, pattern

Use normal mode with append register to collect text from multiple lines into one register.

Example

```
" Yank text inside {} from multiple lines to register A:
:'<,'>norm "Ayi{

" Yank word under cursor from multiple lines:
:g/pattern/ normal "Ayiw

" Clear register first:
qAq
:'<,'>norm "Ayi{
```

20.29 Substitute character

Category: Editing
Tags: substitute, character, change

Use `s` to substitute character (delete and enter insert mode), `S` for entire line.

Example

```
s  " substitute character
S  " substitute line
```

20.30 Substitute entire line and start insert

Category: Editing
Tags: substitute, line, insert, indentation

Use `S` to delete the entire line and start insert mode with proper indentation.

Example

```
S  " delete line and start insert at correct indentation
```

20.31 Substitute in all buffers

Category: Editing
Tags: substitute, buffer, all, bufdo, global

Use `:bufdo %s/old/new/ge` to substitute in all open buffers, `e` flag suppresses errors.

Example

```
:bufdo %s/old/new/ge    " substitute in all buffers
:bufdo %s/TODO/DONE/ge  " replace TODO with DONE in all buffers
:bufdo update           " save all modified buffers
```

20.32 Wrap text in HTML tags

Category: Editing
Tags: html, tag, wrap, surround, format

Use visual selection and substitute to wrap text in HTML tags.

Example

```
" Select text in visual mode, then:
:'<,'>s/.*/\<p>&\<\/p>/   " wrap lines in <p> tags
:'<,'>s/.*/\<li>&\<\/li>/ " wrap lines in <li> tags
:'<,'>s/\(.*\)/\<strong>\1\<\/strong>/ " wrap in <strong> tags
```

20.33 Yank (copy) operations

Category: Editing
Tags: yank, copy, clipboard

Use `yy` to yank entire line, `yw` to yank word, `y$` to yank to end of line.

Example

```
yy  " yank entire line
yw  " yank word
y$  " yank to end of line
```

CHAPTER 21

Ex commands (advanced)

21.1 Advanced substitute flags

Category: Search Replace
Tags: substitute, flags, options, advanced, ex

Use various flags with `:substitute` for advanced replacement options.

Example

```
:s/old/new/I        " case-sensitive (ignore ignorecase)
:s/old/new/i        " case-insensitive
:s/old/new/e        " no error if pattern not found
:s/old/new/n        " report matches but don't substitute
:s/old/new/p        " print line after substitution
```

21.2 Append text after line

Category: Text Editing
Tags: append, insert, text, add, ex

Use `:append` or `:a` to enter text entry mode, adding lines after specified line.

Example

```
:5a                 " append after line 5
:append             " append after current line
This is new text
.                   " end with dot on empty line
```

21.3 Browse with file dialog

Category: File Operations
Tags: browse, dialog, file, gui, ex

Use `:browse` to open file dialog for commands (GUI Vim only).

Example

```
:browse edit        " browse for file to edit
:browse saveas      " browse for save location
:browse read        " browse for file to read
:browse source      " browse for script to source
```

21.4 Center align text

Category: Formatting
Tags: center, align, format, text, ex

Use `:center` or `:ce` to center-align text within specified width.

Example

```
:center             " center current line (default width)
:ce 80              " center in 80-character width
:1,5ce 60           " center lines 1-5 in 60 chars
```

21.5 Change lines with text entry

Category: Text Editing
Tags: change, replace, lines, text, ex

Use `:change` or `:c` to replace line range with new text.

Example

```
:5c                 " change line 5
:1,3c               " change lines 1-3
New replacement text
.                   " end with dot on empty line
```

21.6 Change tab settings and convert

Category: Formatting
Tags: retab, tabs, spaces, convert, ex

Use `:retab` to convert tabs to spaces or vice versa based on current settings.

Example

```
:retab              " convert tabs using current tabstop
:retab 4            " convert using 4-space tabs
:retab!             " also change tab settings
:1,10retab 2        " retab lines 1-10 with 2-space tabs
```

21.7 Command history navigation

Category: Command Line
Tags: history, navigate, command, previous, ex

Use history navigation to recall and modify previous Ex commands.

Example

```
:<Up>               " previous command in history
:<Down>             " next command in history
:<C-p>              " previous command (alternative)
:<C-n>              " next command (alternative)
:his                " show command history
```

21.8 Execute on non-matching lines

Category: Text Manipulation
Tags: vglobal, inverse, global, exclude, ex

Use `:vglobal` or `:v` to execute commands on lines NOT matching pattern.

Example

```
:v/pattern/d        " delete lines NOT containing pattern
:vglobal/TODO/p     " print lines without TODO
:5,10v/^$/d         " delete non-empty lines in range 5-10
```

Opposite of `:global` - executes on non-matching lines.

21.9 Global with range

Category: Text Manipulation
Tags: global, range, lines, scope, ex

Use `:global` with line ranges to limit scope of global operations.

Example

```
:5,50g/pattern/d    " delete matching lines only in range 5-50
:.,+10g/TODO/p      " print TODO lines from current to +10 lines
:'<,'>g/^#/s/#/\/\//  " in visual selection, change # to //
```

21.10 Insert text before line

Category: Text Editing
Tags: insert, text, before, add, ex

Use `:insert` or `:i` to enter text entry mode, adding lines before specified line.

Example

```
:5i                 " insert before line 5
:insert             " insert before current line
New text here
.                   " end with dot on empty line
```

21.11 Join lines together

Category: Text Editing
Tags: join, lines, merge, combine, ex

Use `:join` or `:j` to join lines, removing line breaks.

Example

```
:join               " join current and next line
:5,10j              " join lines 5 through 10
:j!                 " join without inserting spaces
:j 3                " join current line with next 2 lines
```

21.12 Keep jump list during operation

Category: Navigation
Tags: keepjumps, jumps, preserve, navigation, ex

Use `:keepjumps` to prevent commands from adding entries to jump list.

Example

```
:keepjumps normal! G     " go to end without jump entry
:keepjumps /pattern      " search without jump entry
```

21.13 Keep marks during operation

Category: Marks
Tags: keepmarks, marks, preserve, maintain, ex

Use `:keepmarks` to preserve mark positions during range operations.

Example

```
:keepmarks 1,5d     " delete lines 1-5 but keep marks
:kee s/old/new/g    " substitute preserving marks
```

21.14 Left align text

Category: Formatting
Tags: left, align, format, text, ex

Use `:left` or `:le` to left-align text, removing leading whitespace.

Example

```
:left               " left-align current line
:le 4               " left-align with 4-space indent
:1,10left 0         " remove all leading whitespace from lines 1-10
```

21.15 List old files

Category: File History
Tags: oldfiles, recent, history, files, ex

Use `:oldfiles` to show list of recently edited files.

Example

```
:oldfiles           " show recently edited files
:ol                 " shorter version
:browse oldfiles    " browse old files with dialog (GUI)
```

Files are numbered; use ‘:e #< to edit by number.

21.16 Load saved view

Category: View Management
Tags: loadview, view, restore, position, ex

Use `:loadview` to restore previously saved window view.

Example

```
:loadview           " load view with automatic name
:lo 1               " load from view slot 1
:loadview ~/my.vim  " load view from specific file
```

21.17 Lock marks during operation

Category: Marks
Tags: lockmarks, marks, preserve, lock, ex

Use `:lockmarks` to prevent commands from changing mark positions.

Example

```
:lockmarks normal! dd    " delete line without affecting marks
:loc s/old/new/g         " substitute without moving marks
```

21.18 Make session file

Category: Session Management
Tags: mksession, session, save, workspace, ex

Use `:mksession` to save current editing session to file.

Example

```
:mksession          " create Session.vim in current dir
:mks ~/my.vim       " save session to specific file
:mks!               " overwrite existing session file
```

Restore with `:source Session.vim` or `nvim -S Session.vim`.

21.19 Nested global commands

Category: Text Manipulation
Tags: global, nested, complex, pattern, ex

Use nested `:global` commands for complex pattern operations.

Example

```
:g/function/+1,/^}/g/TODO/p   " find TODO in function bodies
:g/class/.,/^$/v/def/d        " delete non-def lines in classes
```

Inner global operates on lines found by outer global.

21.20 Put register contents

Category: Registers
Tags: put, paste, register, insert, ex

Use `:put` to insert register contents after current line.

Example

```
:put                " put default register after current line
:put a              " put register 'a' after current line
:5put               " put after line 5
:put!               " put before current line
:put +              " put system clipboard
```

21.21 Quit with error code

Category: Exit
Tags: cquit, quit, error, code, ex

Use `:cquit` or `:cq` to quit Vim with error exit code.

Example

```
:cquit              " quit with error code
:cq                 " shorter version
:cq 2               " quit with specific error code
```

Useful in shell scripts to indicate failure.

21.22 Range with patterns

Category: Text Manipulation
Tags: range, pattern, search, scope, ex

Use patterns as range specifiers in Ex commands.

Example

```
:/pattern1/,/pattern2/d        " delete from first to second pattern
:/function/+1,/^}/s/old/new/g  " substitute in function body
:?#include?,/main/p            " print from include backward to main
```

21.23 Return to normal mode

Category: Mode Switching
Tags: visual, normal, mode, return, ex

Use `:visual` or `:vi` to return to Normal mode from Ex mode.

Example

```
:visual             " return to Normal mode
:vi                 " shorter version
```

Historical command, rarely needed in modern Neovim.

21.24 Right align text

Category: Formatting
Tags: right, align, format, text, ex

Use `:right` or `:ri` to right-align text within specified width.

Example

```
:right              " right-align current line
:ri 80              " right-align in 80-character width
:1,5ri 60           " right-align lines 1-5 in 60 chars
```

21.25 Save current view

Category: View Management
Tags: mkview, view, save, position, ex

Use `:mkview` to save current window view (cursor position, folds, etc.).

Example

```
:mkview             " save view with automatic name
:mkv 1              " save to view slot 1
:mkview ~/my.vim    " save view to specific file
```

21.26 Sort lines alphabetically

Category: Text Manipulation
Tags: sort, alphabetical, lines, order, ex

Use `:sort` to sort lines in various ways.

Example

```
:sort               " sort current buffer alphabetically
:5,10sort           " sort lines 5-10
:sort!              " reverse sort (descending)
:sort n             " numeric sort
:sort u             " remove duplicates while sorting
```

```
:sort i             " case-insensitive sort
:%!sort
```

21.27 Substitute confirmation

Category: Search Replace
Tags: substitute, confirm, interactive, replace, ex

Use the `c` flag with `:substitute` for interactive confirmation of each replacement.

Example

```
:s/old/new/gc       " substitute with confirmation
:%s/foo/bar/gc      " replace in whole file with prompts
:5,10s/x/y/gc       " replace in range with confirmation
```

Prompts: `yes`, `no`, `all`, `quit`, `last`.

21.28 Substitute with backreferences

Category: Search Replace
Tags: substitute, backreference, capture, group, ex

Use `\(` and `\)` to capture groups, reference with `\1`, `\2`, etc.

Example

```
:s/\(word1\) \(word2\)/\2 \1/    " swap two words
:s/\(\w\+\)\s\+\(\w\+\)/\2, \1/  " swap and add comma
:%s/\(.*\)/"\1"/                 " quote all lines
```

21.29 Substitute with expressions

Category: Search Replace
Tags: substitute, expression, function, dynamic, ex

Use `\=` in replacement to evaluate Vim expressions.

Example

```
:s/\d\+/\=submatch(0)*2/g    " double all numbers
:s/.*/\=line('.').' '.submatch(0)/  " add line number prefix
:%s/$/\=', line: '.line('.')/       " add line number suffix
```

21.30 Write all and quit all

Category: File Operations
Tags: wqall, write, quit, all, buffers, ex

Use `:wqall` to write all modified buffers and quit all windows.

Example

```
:wqall              " write all modified and quit all
:wqa                " shorter version
:xa                 " write all modified and exit (alternative)
```

21.31 Write and exit

Category: File Operations
Tags: xit, write, exit, save, quit, ex

Use `:xit` or `:x` to write file only if modified, then exit.

Example

```
:xit                " write if modified and exit
:x                  " shorter version
:5,10x file.txt     " write lines 5-10 to file and exit
```

More efficient than `:wq` since it only writes when necessary.

21.32 Yank lines to register

Category: Registers
Tags: yank, copy, register, lines, ex

Use `:yank` or `:y` to copy lines to a register.

Example

```
:yank               " yank current line to default register
:5y                 " yank line 5
:1,3y a             " yank lines 1-3 to register 'a'
:y 5                " yank 5 lines starting from current
```

CHAPTER 22

Ex commands (comprehensive)

22.1 Buffer list navigation

Category: Buffer Management
Tags: buffer, list, navigation, ex

Use `:ls` or `:buffers` to show all buffers with their numbers and status indicators.

Example

```
:ls                 " list all buffers
:buffers           " same as :ls (alternative)
:ls!               " list all buffers including unlisted
```

Status indicators: `%` current, `#` alternate, `+` modified, `x` read errors.

22.2 Close all windows except current

Category: Window Management
Tags: window, close, only, layout, ex

Use `:only` or `:on` to close all windows except the current one.

Example

```
:only              " close all other windows
:on                " shorter version
```

Useful for cleaning up complex window layouts quickly.

22.3 Copy lines to another location

Category: Text Manipulation
Tags: copy, lines, move, range, ex

Use `:copy` or `:co` to copy lines to another location in the file.

Example

```
:5copy10          " copy line 5 after line 10
:1,3co$           " copy lines 1-3 to end of file
:.co0             " copy current line to beginning
:co.              " copy current line after itself (duplicate)
```

22.4 Create new empty buffer

Category: Buffer Management
Tags: buffer, new, empty, enew, ex

Use `:enew` to create a new empty buffer in current window.

Example

```
:enew             " create new empty buffer
:new              " create new buffer in split window
:vnew             " create new buffer in vertical split
```

22.5 Delete buffers

Category: Buffer Management
Tags: buffer, delete, close, ex

Use `:bdelete` or `:bd` to remove buffer from list, `:bwipeout` to completely wipe buffer.

Example

```
:bdelete          " delete current buffer
:bd 3             " delete buffer number 3
:bd file.txt      " delete buffer by name
:bwipeout         " completely wipe current buffer
:1,3bd            " delete buffers 1 through 3
```

22.6 Delete specific lines

Category: Text Manipulation
Tags: delete, lines, range, remove, ex

Use `:delete` or `:d` to delete specific lines or line ranges.

Example

```
:5delete          " delete line 5
:1,3d             " delete lines 1 through 3
:.,$d             " delete from current line to end
```

```
:g/pattern/d       " delete all lines containing pattern
```

22.7 Execute normal mode commands

Category: Command Execution
Tags: normal, execute, mode, command, ex

Use `:normal commands` to execute normal mode commands from Ex mode.

Example

```
:normal dd         " delete current line
:5,10normal A;     " append semicolon to lines 5-10
:%normal I//       " comment all lines with //
```

Use `!` to avoid mappings: `:normal! dd`

22.8 Find file in path

Category: File Operations
Tags: find, path, file, search, ex

Use `:find filename` to search for file in 'path' option directories and edit it.

Example

```
:find config.vim   " find and edit config.vim in path
:fin *.py          " find Python files (with tab completion)
```

Set your path with `:set path+=directory` to include custom directories.

22.9 First and last files in argument list

Category: File Navigation
Tags: first, last, file, argument, list, ex

Use `:first` and `:last` to jump to first or last file in argument list.

Example

```
:first             " edit first file in argument list
:rewind            " same as :first
:last              " edit last file in argument list
```

22.10 Go to specific buffer by number

Category: Buffer Management
Tags: buffer, number, navigation, ex

Use `:buffer N` or `:b N` to switch to buffer number N from the buffer list.

Example

```
:buffer 3          " go to buffer number 3
:b 3               " shorter version
:b filename        " go to buffer by partial filename match
```

22.11 Internal grep with vimgrep

Category: Search
Tags: vimgrep, search, internal, pattern, ex

Use `:vimgrep` to search using Vim's internal grep (works with Vim patterns).

Example

```
:vimgrep /pattern/j *.py   " search in Python files
:vim /\cTODO/ **/*.js      " case-insensitive recursive search
:vimgrep /\<word\>/ %      " search for whole word in current file
```

Use `j` flag to avoid jumping to first match immediately.

22.12 Jump to tag definition

Category: Navigation
Tags: tag, jump, definition, ctags, ex

Use `:tag tagname` to jump to tag definition (requires tags file).

Example

```
:tag function_name " jump to tag definition
:ta MyClass        " jump to MyClass tag
:tag /pattern      " search for tags matching pattern
```

Generate tags with `ctags -R .` in your project root.

22.13 List all sourced scripts

Category: Configuration
Tags: scripts, source, debug, files, ex

Use `:scriptnames` to list all sourced Vim script files with their script IDs.

Example

```
:scriptnames       " list all sourced scripts
:scr               " shorter version
```

Useful for debugging configuration issues and seeing load order.

22.14 Location list navigation

Category: Search
Tags: location, list, navigate, lnext, ex

Use `:lnext`, `:lprev` to navigate location list (window-local quickfix).

Example

```
:lnext             " go to next item in location list
:lprev             " go to previous item
:lfirst            " go to first item
:llast             " go to last item
:lopen             " open location list window
```

22.15 Move lines to another location

Category: Text Manipulation
Tags: move, lines, cut, range, ex

Use `:move` or `:m` to move lines to another location in the file.

Example

```
:5move10           " move line 5 after line 10
:1,3m$             " move lines 1-3 to end of file
:.m0               " move current line to beginning
:m+1               " move current line down one position
```

22.16 Next file in argument list

Category: File Navigation
Tags: next, file, argument, list, ex

Use `:next` or `:n` to edit next file in argument list.

Example

```
:next               " edit next file
:n                  " shorter version
:2next              " skip 2 files forward
```

See argument list with `:args`, set with `nvim file1 file2 file3`.

22.17 Previous file in argument list

Category: File Navigation
Tags: previous, file, argument, list, ex

Use `:previous` or `:prev` to edit previous file in argument list.

Example

```
:previous           " edit previous file
:prev               " shorter version
:2prev              " skip 2 files backward
```

22.18 Previous tag in stack

Category: Navigation
Tags: tag, previous, stack, back, ex

Use `:pop` or `:po` to go back to previous location in tag stack.

Example

```
:pop                " go back in tag stack
:po                 " shorter version
:2pop               " go back 2 positions
```

Use after jumping to tags with `:tag` or `Ctrl+]`.

22.19 Quickfix list navigation

Category: Search
Tags: quickfix, navigate, error, jump, ex

Use `:cnext`, `:cprev` to navigate quickfix list (global error list).

Example

```
:cnext              " go to next quickfix item
:cprev              " go to previous item
```

```
:cfirst            " go to first item
:clast             " go to last item
:copen             " open quickfix window
:cclose            " close quickfix window
```

22.20 Quit all windows/buffers

Category: File Operations
Tags: quit, all, exit, buffers, ex

Use `:qall` or `:qa` to quit all windows, `:qa!` to quit without saving.

Example

```
:qall              " quit all windows/buffers
:qa                " shorter version
:qa!               " quit all without saving changes
:wqa               " save all and quit
```

22.21 Recover file from swap

Category: File Recovery
Tags: recover, swap, file, crash, ex

Use `:recover filename` or `:rec` to recover file from swap file after crash.

Example

```
:recover file.txt  " recover file from swap
:rec               " recover current file
```

Vim creates `.swp` files for crash recovery. Use this after unexpected shutdowns.

22.22 Repeat last Ex command

Category: Command History
Tags: repeat, last, command, history, ex

Use `@:` to repeat the last Ex command, `@@` to repeat last `@` command.

Example

```
@:                 " repeat last Ex command
5@:                " repeat last Ex command 5 times
@@                 " repeat the last @ command
```

Useful for repeating complex commands without retyping.

22.23 Run grep and jump to matches

Category: Search
Tags: grep, search, quickfix, external, ex

Use `:grep pattern files` to run external grep and jump to first match.

Example

```
:grep TODO *.py    " search for TODO in Python files
:grep -r "function" src/  " recursive search in src/
:grep! pattern *   " run grep but don't jump to first match
```

Results appear in quickfix list. Use `:cn` and `:cp` to navigate.

22.24 Save all modified buffers

Category: File Operations
Tags: save, all, buffers, write, wa, ex

Use `:wall` or `:wa` to save all modified buffers at once.

Example

```
:wall              " write all modified buffers
:wa                " shorter version
:wqa               " write all and quit
:xa                " write all modified and exit
```

22.25 Set local options

Category: Configuration
Tags: set, local, options, buffer, window, ex

Use `:setlocal` to set options only for current buffer/window.

Example

```
:setlocal number   " show line numbers in current buffer only
:setl ts=2         " set tabstop to 2 for current buffer
:setlocal ft=python " set filetype for current buffer
```

22.26 Show argument list

Category: File Navigation
Tags: args, argument, list, files, ex

Use `:args` to display current argument list with current file highlighted.

Example

```
:args              " show argument list
:args *.py         " set argument list to all Python files
:args **/*.js      " recursively find all JavaScript files
```

22.27 Show version information

Category: System Information
Tags: version, info, build, features, ex

Use `:version` to display Neovim version, build info, and compiled features.

Example

```
:version           " show full version information
:ve                " shorter version
```

Shows version number, build date, features, and compilation options.

22.28 Source Vim scripts

Category: Configuration
Tags: source, script, load, runtime, ex

Use `:source` to execute Vim script file, `:runtime` to source from runtime path.

Example

```
:source ~/.vimrc   " source specific file
:so %              " source current file
:runtime plugin/myplugin.vim  " source from runtime path
:ru syntax/python.vim         " load Python syntax
```

CHAPTER 23

Ex commands (extended)

23.1 Add buffer to list

Category: Buffer Management
Tags: badd, buffer, add, list, ex

Use `:badd` to add file to buffer list without editing it.

Example

```
:badd file.txt     " add file to buffer list
:badd *.py         " add all Python files
```

Useful for preparing a list of files to work with.

23.2 AutoGroup management

Category: Scripting
Tags: augroup, autocmd, group, event, ex

Use `:augroup` to group autocommands and manage them collectively.

Example

```
:augroup MyGroup    " start group definition
:autocmd!           " clear existing autocmds in group
:autocmd BufRead * echo "File read"
:augroup END        " end group definition
```

23.3 Call functions

Category: Scripting
Tags: call, function, execute, script, ex

Use `:call` to execute functions and discard their return value.

Example

```
:call search('pattern')  " call search function
:call setline('.', 'new text')  " replace current line
:call cursor(10, 5)      " move cursor to line 10, column 5
```

23.4 Change working directory

Category: Navigation
Tags: directory, cd, path, working, ex

Use `:cd` to change current working directory, `:pwd` to show current directory.

Example

```
:cd ~/projects      " change to home/projects directory
:cd %:h             " change to directory of current file
:pwd                " show current working directory
:cd -               " change to previous directory
```

23.5 Check file path existence

Category: File Operations
Tags: checkpath, include, path, files, ex

Use `:checkpath` to verify all files in include path can be found.

Example

```
:checkpath          " check all included files
:checkp!            " show files that cannot be found
```

Useful for debugging include paths in C/C++ projects.

23.6 Conditional execution

Category: Scripting
Tags: if, condition, branch, script, ex

Use `:if`, `:else`, `:endif` for conditional execution in scripts.

Example

```
:if &filetype == 'python'
  echo "Python file"
:else
  echo "Not Python"
```

```
:endif
```

23.7 Create abbreviations

Category: Text Input
Tags: abbreviate, abbr, shortcut, expand, ex

Use `:abbreviate` or `:abbr` to create text shortcuts that expand automatically.

Example

```
:abbr teh the       " auto-correct 'teh' to 'the'
:abbr @@ john@example.com  " expand @@ to email
:iabbr <buffer> fn function  " buffer-local abbreviation
:unabbr teh         " remove abbreviation
```

23.8 Define variables

Category: Scripting
Tags: let, variable, assign, define, ex

Use `:let` to define and assign values to variables.

Example

```
:let g:my_var = 'value'    " global variable
:let b:buffer_var = 123    " buffer-local variable
:let &tabstop = 4          " set option value
:unlet g:my_var            " delete variable
```

23.9 Echo text and expressions

Category: Scripting
Tags: echo, print, expression, debug, ex

Use `:echo` to print text or evaluate expressions in command line.

Example

```
:echo "Hello World" " print text
:echo &tabstop      " show value of tabstop option
:echo expand('%')   " show current filename
:echo line('.')     " show current line number
```

23.10 File information

Category: Information
Tags: file, info, status, buffer, ex

Use `:file` to show file information and optionally rename buffer.

Example

```
:file              " show file info (name, lines, position)
:file newname.txt  " rename current buffer
:f                 " shorter version
```

23.11 Function definition

Category: Scripting
Tags: function, define, script, procedure, ex

Use `:function` to define custom functions in Vim script.

Example

```
:function! MyFunc()
  echo "Hello from function"
endfunction

:call MyFunc()      " call the function
:delfunc MyFunc     " delete function
```

23.12 Help search

Category: Help
Tags: helpgrep, help, search, documentation, ex

Use `:helpgrep` to search through all help files for patterns.

Example

```
:helpgrep pattern   " search help for pattern
:helpg autocmd      " search for autocmd info
:cn                 " next help match
:cp                 " previous help match
```

23.13 Include jump

Category: Navigation
Tags: ijump, include, file, search, ex

Use `:ijump` to jump to first line containing identifier in include files.

Example

```
:ijump printf       " jump to printf definition in includes
:ij MyFunc          " jump to MyFunc in include files
```

23.14 Include list

Category: Navigation
Tags: ilist, include, search, show, ex

Use `:ilist` to list all lines containing identifier in include files.

Example

```
:ilist printf       " list all printf occurrences
:il /pattern/       " list lines matching pattern
:il! MyFunc         " list including header files
```

23.15 Introduction screen

Category: Interface
Tags: intro, welcome, screen, startup, ex

Use `:intro` to show the Neovim introduction/welcome screen.

Example

```
:intro              " show introduction screen
```

Useful after clearing the screen or when you want to see version info.

23.16 Key mapping

Category: Mapping
Tags: map, key, mapping, shortcut, ex

Use `:map` to create key mappings, `:noremap` for non-recursive mappings.

Example

```
:map <F2> :w<CR>          " map F2 to save
:nmap <leader>q :q<CR>    " normal mode mapping
:imap <C-s> <Esc>:w<CR>a  " insert mode mapping
:unmap <F2>               " remove mapping
```

23.17 Language settings

Category: Configuration
Tags: language, locale, encoding, international, ex

Use `:language` to set language for messages and time.

Example

```
:language messages en_US.UTF-8  " set message language
:language time C                " set time language
:language                       " show current settings
```

23.18 Make and build

Category: Development
Tags: make, build, compile, external, ex

Use `:make` to run external make command and capture errors.

Example

```
:make              " run make command
:make clean        " run make with clean target
:make -j4          " run make with 4 parallel jobs
```

Errors appear in quickfix list. Use `:cn` to navigate.

23.19 Match highlighting

Category: Display
Tags: match, highlight, pattern, visual, ex

Use `:match` to highlight patterns with specific colors in current window.

Example

```
:match ErrorMsg /TODO/     " highlight TODO in red
:match Search /\<word\>/   " highlight whole word
:match none                " clear all matches
```

```
:2match Comment /pattern/  " second match group
```

23.20 Menu creation

Category: Interface
Tags: menu, gui, interface, create, ex

Use `:menu` to create menu items (GUI mode).

Example

```
:menu File.Save :w<CR>           " create Save menu item
:menu Edit.Find :promptfind<CR>  " create Find menu item
:unmenu File.Save                " remove menu item
```

23.21 Neovim health check

Category: Diagnostics
Tags: checkhealth, health, diagnostic, status, ex

Use `:checkhealth` to run diagnostic checks on Neovim installation and plugins.

Example

```
:checkhealth        " check all health
:checkhealth nvim   " check only Neovim core health
:checkhealth vim    " check Vim compatibility
```

23.22 Print lines

Category: Display
Tags: print, lines, show, output, ex

Use `:print` or `:p` to print lines to command area.

Example

```
:print             " print current line
:1,5p              " print lines 1 through 5
:.,$p              " print from current line to end
:p #               " print with line numbers
```

23.23 Runtime file loading

Category: Configuration
Tags: runtime, load, path, script, ex

Use `:runtime` to load script files from runtime path directories.

Example

```
:runtime! plugin/**/*.vim  " load all plugins
:runtime syntax/python.vim " load Python syntax
:ru macros/matchit.vim     " load matchit macro
```

23.24 Show all marks

Category: Navigation
Tags: marks, list, show, position, ex

Use `:marks` to display all marks with their line numbers and text.

Example

```
:marks              " show all marks
:marks aB           " show only marks 'a' and 'B'
:delmarks a-z       " delete lowercase marks
:delmarks!          " delete all marks for current buffer
```

23.25 Show all messages

Category: Information
Tags: messages, history, errors, warnings, ex

Use `:messages` to display message history including errors and warnings.

Example

```
:messages           " show all messages
:mes                " shorter version
:messages clear     " clear message history
```

23.26 Show digraphs

Category: Text Input
Tags: digraphs, special, characters, unicode, ex

Use `:digraphs` to show available two-character combinations for special characters.

Example

```
:digraphs           " show all digraphs
:dig                " shorter version
```

In insert mode, use `Ctrl+k` followed by two characters (e.g., `Ctrl+k a'` for á).

23.27 Show jump list

Category: Navigation
Tags: jumps, list, history, navigation, ex

Use `:jumps` to display jump list with positions you can return to.

Example

```
:jumps              " show jump list
:ju                 " shorter version
```

Use `Ctrl+o` to go back, `Ctrl+i` to go forward in jump list.

23.28 Show registers content

Category: Registers
Tags: registers, show, content, clipboard, ex

Use `:registers` or `:reg` to display contents of all registers.

Example

```
:registers          " show all registers
:reg abc            " show only registers a, b, and c
:reg "              " show default register
:reg +              " show system clipboard register
```

23.29 Spell checking commands

Category: Text Editing
Tags: spell, check, dictionary, correction, ex

Use spell-related commands to manage spell checking.

Example

```
:spell             " enable spell checking
:set spell         " same as :spell
:spellgood word    " add word to good word list
```

```
:spellwrong word   " add word as wrong word
:spelldump         " show all spell words
```

23.30 Tag selection

Category: Navigation
Tags: tselect, tag, multiple, choose, ex

Use `:tselect` when multiple tag matches exist to choose from a list.

Example

```
:tselect function   " show list of function tags
:ts MyClass         " show list of MyClass tags
:tnext              " go to next tag match
:tprev              " go to previous tag match
```

23.31 Unlet variables

Category: Scripting
Tags: unlet, variable, delete, remove, ex

Use `:unlet` to delete variables and free memory.

Example

```
:unlet g:my_var    " delete global variable
:unlet! b:temp     " delete if exists (no error)
:unlet $HOME       " delete environment variable
```

CHAPTER 24

Exit

24.1 Quit Vim

Category: Exit
Tags: quit, exit, close

Use `:q` to quit, `:q!` to quit without saving, `:wq` or `:x` to write and quit, `ZZ` to save and exit, 'ZQ' to quit withut saving.

Example

```
:q   " quit
:q!  " quit without saving
:wq  " write and quit
ZZ   " save and exit
ZQ   " quit without saving
```

CHAPTER 25

Extmarks

25.1 Add sign column signs with extmarks

Category: Extmarks
Tags: extmarks, signs, sign-column, gutter

Use extmarks with `sign_text` to place signs in the sign column. More flexible than traditional signs.

Example

```
local ns_id = vim.api.nvim_create_namespace('my_signs')

-- Add a sign to line 10
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  sign_text = "⚠",
  sign_hl_group = "WarningMsg",
})

-- Add sign with number column highlight
vim.api.nvim_buf_set_extmark(0, ns_id, 5, 0, {
  sign_text = "✓",
  sign_hl_group = "DiffAdd",
  number_hl_group = "DiffAdd",  -- highlight line number too
  line_hl_group = "DiffAdd",     -- highlight entire line
})
```

25.2 Add virtual text with extmarks

Category: Extmarks
Tags: extmarks, virtual-text, inline-hints

Use extmarks with `virt_text` to display text that isn't actually in the buffer. Perfect for inline hints, diagnostics, or annotations.

Example

```
local ns_id = vim.api.nvim_create_namespace('hints')

-- Add virtual text at end of line
```

```
vim.api.nvim_buf_set_extmark(0, ns_id, 5, 0, {
  virt_text = {{"← This is a hint", "Comment"}},
  virt_text_pos = "eol",  -- end of line
})

-- Add virtual text inline (after specific column)
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 15, {
  virt_text = {{"[TODO]", "WarningMsg"}},
  virt_text_pos = "inline",
})

-- Add virtual text as overlay (replaces text visually)
vim.api.nvim_buf_set_extmark(0, ns_id, 3, 0, {
  end_col = 6,  -- covers first 6 characters
  virt_text = {{"FIXME", "ErrorMsg"}},
  virt_text_pos = "overlay",
})
```

25.3 Build a simple word abbreviation system

Category: Extmarks
Tags: extmarks, abbreviations, overlay, practical

Create a complete system to visually abbreviate long keywords using extmarks overlays.

Example

```
-- Abbreviation system
local M = {}
local ns_id = vim.api.nvim_create_namespace('abbreviations')

local abbreviations = {
  ["function"] = "fn",
  ["return"] = "ret",
  ["require"] = "req",
  ["local"] = "lcl",
}

function M.apply_abbreviations(bufnr)
  bufnr = bufnr or 0

  -- Clear old abbreviations
  vim.api.nvim_buf_clear_namespace(bufnr, ns_id, 0, -1)

  local lines = vim.api.nvim_buf_get_lines(bufnr, 0, -1, false)

  for lnum, line in ipairs(lines) do
    for word, abbrev in pairs(abbreviations) do
      local col = 0
      while true do
        local start, finish = string.find(line, word, col, true)
        if not start then break end
```

```
        -- Check if it's a whole word (not part of another word)
        local before = start > 1 and line:sub(start - 1, start - 1) or " "
        local after = finish < #line and line:sub(finish + 1, finish + 1) or
        ↪  " "

        if before:match("[%W_]") and after:match("[%W_]") then
          vim.api.nvim_buf_set_extmark(bufnr, ns_id, lnum - 1, start - 1, {
            end_col = finish,
            virt_text = {{abbrev, "Keyword"}},
            virt_text_pos = "overlay",
          })
        end

        col = finish + 1
      end
    end
  end
end

function M.toggle()
  local bufnr = vim.api.nvim_get_current_buf()
  local marks = vim.api.nvim_buf_get_extmarks(bufnr, ns_id, 0, -1, {})

  if #marks > 0 then
    -- Remove abbreviations
    vim.api.nvim_buf_clear_namespace(bufnr, ns_id, 0, -1)
    print("Abbreviations disabled")
  else
    -- Apply abbreviations
    M.apply_abbreviations(bufnr)
    print("Abbreviations enabled")
  end
end

-- Setup
vim.api.nvim_create_user_command('ToggleAbbrev', M.toggle, {})

-- Auto-apply on buffer changes (optional)
vim.api.nvim_create_autocmd({"TextChanged", "TextChangedI"}, {
  callback = function()
    M.apply_abbreviations()
  end,
})

return M
```

25.4 Clear extmarks from buffer

Category: Extmarks
Tags: extmarks, clear, cleanup, namespace

Use `nvim_buf_clear_namespace()` to remove extmarks. Essential for cleanup and re-

freshing decorations.

Example

```
local ns_id = vim.api.nvim_create_namespace('my_marks')

-- Clear all extmarks in namespace from current buffer
vim.api.nvim_buf_clear_namespace(0, ns_id, 0, -1)

-- Clear extmarks only in line range (lines 10-20)
vim.api.nvim_buf_clear_namespace(0, ns_id, 10, 20)

-- Clear specific extmark by id
vim.api.nvim_buf_del_extmark(0, ns_id, mark_id)
```

25.5 Create basic extmark

Category: Extmarks
Tags: extmarks, api, decoration, namespace

Use `vim.api.nvim_buf_set_extmark()` to place an extmark in a buffer. Extmarks are positions that move with text edits and can have decorations.

Example

```
-- Create a namespace (do this once, usually at plugin setup)
local ns_id = vim.api.nvim_create_namespace('my_plugin')

-- Place an extmark at line 0 (1st line), column 0
local mark_id = vim.api.nvim_buf_set_extmark(0, ns_id, 0, 0, {
  -- Empty options means just a position marker
})

-- Get extmark position later
local pos = vim.api.nvim_buf_get_extmark_by_id(0, ns_id, mark_id, {})
print("Extmark is at line:", pos[1], "col:", pos[2])
```

25.6 Create custom highlighting with extmark priorities

Category: Extmarks
Tags: extmarks, priorities, highlight, layering

Use extmark priorities to control which highlights are visible when they overlap.

Example

```
local ns_id = vim.api.nvim_create_namespace('highlights')

-- Low priority background highlight
```

```
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  end_col = 50,
  hl_group = "CursorLine",
  priority = 100,
})

-- Medium priority for syntax
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 10, {
  end_col = 20,
  hl_group = "String",
  priority = 150,
})

-- High priority for errors (will show on top)
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 15, {
  end_col = 18,
  hl_group = "ErrorMsg",
  priority = 200,
})

-- Note: Default priority is 4096
-- Treesitter uses priority 100
-- LSP semantic tokens use priority 125
```

25.7 Create inline diagnostics with extmarks

Category: Extmarks
Tags: extmarks, diagnostics, lsp, inline

Combine extmarks features to create rich inline diagnostics like modern IDEs.

Example

```
local ns_id = vim.api.nvim_create_namespace('diagnostics')

-- Underline error and show virtual text
local function show_diagnostic(line, col, end_col, message, severity)
  local hl_group = severity == "error" and "DiagnosticUnderlineError"
                   or "DiagnosticUnderlineWarn"
  local virt_hl = severity == "error" and "DiagnosticVirtualTextError"
                  or "DiagnosticVirtualTextWarn"
  local sign = severity == "error" and "×" or "⚠"

  vim.api.nvim_buf_set_extmark(0, ns_id, line, col, {
    end_col = end_col,
    hl_group = hl_group,
    virt_text = {{" " .. sign .. " " .. message, virt_hl}},
    virt_text_pos = "eol",
    sign_text = sign,
    sign_hl_group = virt_hl,
  })
end
```

```
-- Usage
show_diagnostic(10, 5, 15, "Undefined variable", "error")
show_diagnostic(15, 0, 10, "Unused import", "warn")
```

25.8 Create line number decorations with extmarks

Category: Extmarks
Tags: extmarks, line-numbers, number-column

Use extmarks to customize line number appearance for specific lines.

Example

```
local ns_id = vim.api.nvim_create_namespace('line_numbers')

-- Highlight line number for important lines
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  number_hl_group = "ErrorMsg",  -- red line number
  line_hl_group = "CursorLine",  -- subtle line highlight
})

-- Combine with sign for visual emphasis
vim.api.nvim_buf_set_extmark(0, ns_id, 15, 0, {
  sign_text = "►",
  sign_hl_group = "Search",
  number_hl_group = "Search",
  line_hl_group = "Visual",
})
```

25.9 Create virtual lines with extmarks

Category: Extmarks
Tags: extmarks, virtual-lines, inline-diagnostics

Use `virt_lines` to add entire virtual lines above or below a line without modifying the buffer. Perfect for diagnostics or documentation.

Example

```
local ns_id = vim.api.nvim_create_namespace('virtual_lines')

-- Add a virtual line below line 10
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  virt_lines = {
    {{"⚠ Warning: This function is deprecated", "WarningMsg"}},
    {{"  Use new_function() instead", "Comment"}},
  },
  virt_lines_above = false,  -- below the line (default)
```

```
})

-- Add virtual line above line 5
vim.api.nvim_buf_set_extmark(0, ns_id, 5, 0, {
  virt_lines = {
    {{"━━━━━━━━━━━━━━━━━━━━━━━━━━━━", "Comment"}},
    {{"Section: Helper Functions", "Title"}},
  },
  virt_lines_above = true,
})
```

25.10 Get all extmarks in range

Category: Extmarks
Tags: extmarks, query, list, range

Use `nvim_buf_get_extmarks()` to query extmarks in a buffer. Useful for finding and updating existing decorations.

Example

```
local ns_id = vim.api.nvim_create_namespace('my_marks')

-- Get all extmarks in namespace
local marks = vim.api.nvim_buf_get_extmarks(0, ns_id, 0, -1, {})

for _, mark in ipairs(marks) do
  local mark_id, row, col = mark[1], mark[2], mark[3]
  print(string.format("Mark %d at line %d, col %d", mark_id, row, col))
end

-- Get extmarks with details
local marks_detailed = vim.api.nvim_buf_get_extmarks(0, ns_id, 0, -1, {
  details = true,
})

-- Get extmarks in specific range (lines 10-20)
local range_marks = vim.api.nvim_buf_get_extmarks(0, ns_id, {10, 0}, {20,
↪  -1}, {})
```

25.11 Hide text with extmark conceal

Category: Extmarks
Tags: extmarks, conceal, hide, text

Use extmarks with `conceal` to visually hide text without removing it from the buffer. Useful for hiding verbose syntax or formatting markers.

Example

```
local ns_id = vim.api.nvim_create_namespace('conceal')

-- Hide the word "function" on line 5, starting at column 0
vim.api.nvim_buf_set_extmark(0, ns_id, 5, 0, {
  end_col = 8,  -- length of "function"
  conceal = "",  -- hide completely (empty string)
})

-- Replace "function" with "fn" visually
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  end_col = 8,
  conceal = "fn",  -- show "fn" instead
})

-- Note: Make sure conceallevel is set
vim.opt.conceallevel = 2
```

25.12 Highlight text ranges with extmarks

Category: Extmarks
Tags: extmarks, highlight, ranges, decoration

Use extmarks with `hl_group` to highlight specific text ranges. More powerful than matchadd() and survives buffer edits.

Example

```
local ns_id = vim.api.nvim_create_namespace('highlights')

-- Highlight a word (line 5, columns 10-15)
vim.api.nvim_buf_set_extmark(0, ns_id, 5, 10, {
  end_col = 15,
  hl_group = "Search",
})

-- Highlight entire line
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  end_line = 10,
  end_col = 0,
  hl_group = "DiffAdd",
  hl_eol = true,  -- highlight to end of line
})

-- Highlight with priority (higher = more visible)
vim.api.nvim_buf_set_extmark(0, ns_id, 3, 0, {
  end_col = 5,
  hl_group = "ErrorMsg",
  priority = 200,  -- default is 4096
})
```

25.13 Make extmarks persistent across edits

Category: Extmarks
Tags: extmarks, persistent, tracking, right-gravity

Use extmark options to control how they behave during text edits. Essential for maintaining decorations.

Example

```
local ns_id = vim.api.nvim_create_namespace('persistent')

-- Extmark stays at start of edit (left gravity - default)
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 5, {
  right_gravity = false,  -- default
})

-- Extmark moves to end of inserted text (right gravity)
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 5, {
  right_gravity = true,
})

-- Extmark spans a range and updates with edits
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 5, {
  end_row = 10,
  end_col = 20,
  -- Range automatically adjusts as text is edited
})

-- Disable automatic updates (static position)
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 5, {
  invalidate = true,  -- mark becomes invalid on edit
})
```

25.14 Monitor extmark changes with events

Category: Extmarks
Tags: extmarks, events, autocmd, tracking

Use the `nvim_buf_attach` API to watch for extmark changes in real-time.

Example

```
local ns_id = vim.api.nvim_create_namespace('watched')

-- Attach to buffer to watch changes
vim.api.nvim_buf_attach(0, false, {
  on_lines = function(_, bufnr, _, first_line, last_line_old, last_line_new)
    print(string.format(
      "Lines changed: %d to %d (was %d lines, now %d)",
      first_line, last_line_new,
```

```
      last_line_old - first_line,
      last_line_new - first_line
    ))

    -- Refresh extmarks in changed range
    -- ... update logic here

    return false  -- don't detach
  end,

  on_detach = function()
    print("Buffer detached")
  end,
})
```

25.15 Replace keywords visually with extmarks

Category: Extmarks
Tags: extmarks, overlay, replace, keywords

Use extmarks with `virt_text_pos = "overlay"` to visually replace text like ”return” with ”ret” without changing the buffer.

Example

```
local ns_id = vim.api.nvim_create_namespace('abbreviations')

-- Find all "return" keywords and replace visually with "ret"
local lines = vim.api.nvim_buf_get_lines(0, 0, -1, false)
for lnum, line in ipairs(lines) do
  local col = 0
  while true do
    local start, finish = string.find(line, "return", col, true)
    if not start then break end

    vim.api.nvim_buf_set_extmark(0, ns_id, lnum - 1, start - 1, {
      end_col = finish,
      virt_text = {{"ret", "Keyword"}},
      virt_text_pos = "overlay",
    })
    col = finish + 1
  end
end
```

CHAPTER 26

File operations

26.1 Browse for files with dialog

Category: File Operations
Tags: browse, dialog, gui, file, open

Use `:browse {command}` to open file browser dialog for commands that take filenames (GUI only).

Example

```
:browse edit      " open file browser to edit file
:browse saveas    " open save-as dialog
:browse read      " browse to read file into buffer
:browse source    " browse to source a script file
```

26.2 Check file existence in scripts

Category: File Operations
Tags: file, exist, check, script, function

Use `filereadable()` to check if file exists and is readable, `readfile()` to read all lines.

Example

```
" In Vim script:
if filereadable('config.vim')
  source config.vim
endif

" Read file into list:
let lines = readfile('data.txt')
```

26.3 Ex commands - file permissions and attributes

Category: File Operations
Tags: ex, file, permission, readonly, modifiable

Use `:set readonly` to make read-only, `:set nomodifiable` to prevent changes, `:set fileformat` for line endings.

Example

```
:set readonly      " make buffer read-only
:set nomodifiable  " prevent any modifications
:set fileformat=unix  " set Unix line endings
:set fileformat=dos   " set DOS line endings
```

26.4 Ex commands - read and write operations

Category: File Operations
Tags: ex, read, write, append, output

Use `:read` or `:r` to read file into buffer, `:write` range to write part of buffer, `:.,$w` for current to end.

Example

```
:r file.txt     " read file into current buffer
:read !date     " read output of command
:1,10w part.txt " write lines 1-10 to file
:.,$w end.txt   " write from current line to end
```

26.5 File names with spaces

Category: File Operations
Tags: file, name, space, isfname, path

Use `:set isfname+=32` to allow opening file names containing spaces with `gf` command.

Example

```
:set isfname+=32    " add space (ASCII 32) to filename chars
" Now gf works on: /path/to/file with spaces.txt
:set isfname-=32    " remove space from filename chars
```

26.6 Handle different file formats

Category: File Operations
Tags: file, format, mac, dos, unix, encoding

Use `:e ++ff=mac` to reload file with Mac format, `++ff=dos` for DOS, `++ff=unix` for Unix.

Example

```
:e ++ff=mac       " reload with Mac line endings
:e ++ff=dos       " reload with DOS line endings
:e ++ff=unix      " reload with Unix line endings
:set ff=unix      " change current file format
```

26.7 Insert current date

Category: File Operations
Tags: date, insert, command

Use `:r !date` to insert current date at cursor position.

Example

```
:r !date  " insert current date
```

26.8 Insert file contents

Category: File Operations
Tags: insert, file, read

Use `:r filename` to insert contents of another file at cursor position.

Example

```
:r file.txt  " insert contents of file.txt
```

26.9 Path separator conversion

Category: File Operations
Tags: path, separator, backslash, forward, slash

Use `:s` commands to easily convert between backslash and forward slash in file paths.

Example

```
" Convert backslashes to forward slashes:
:%s/\\/\//g

" Convert forward slashes to backslashes:
:%s/\//\\/g

" Or use built-in function:
:echo substitute(@%, '\\', '/', 'g')
```

26.10 Reload file from disk

Category: File Operations
Tags: reload, file, refresh

Use `:e!` to reload current file from disk, discarding unsaved changes.

Example

```
:e!  " reload file from disk
```

26.11 Save as

Category: File
Tags: save, file

Use `:sav[eas] filepath` to save file under a different name

Example

```
:sav ~/tmp/work.txt
```

26.12 Save file

Category: File Operations
Tags: file, save, write

Use `:w` to save current file, `:w {file}` to save as new file, or `:wall` to save all files.

Example

```
:w             " save current file
:w newfile.txt " save as new file
:wall          " save all files
```

26.13 Save multiple files at once

Category: File Operations
Tags: file, save, multiple, wall, xa

Use `:wa` to save all modified files, `:xa` to save all and exit, `:wqa` to save all and quit.

Example

```
:wa      " write (save) all modified files
:xa      " write all modified files and exit
:wqa     " write all and quit all windows
:qa!     " quit all without saving
```

26.14 Update file only if changed

Category: File Operations
Tags: file, update, save, changed, conditional

Use `:update` to save file only if it has been modified, more efficient than `:write`.

Example

```
:update           " save only if file is modified
:map <F2> :update<CR>  " map F2 to conditional save
```

26.15 Write file and create all directories form the full file path

Category: File Operations
Tags: file, save, write

Use this command to write file if the full path contains non-existent directories. All directories that do not exist will be created before the save:

Example

```
:write ++p
```

CHAPTER 27

Filetype specific tips

27.1 Binary and hex file editing

Category: File Type Specific
Tags: binary, hex, xxd, hexedit, file

Edit binary files using xxd hex editor integration.

Example

```
" Binary file detection and hex mode
:autocmd BufNewFile,BufRead *.bin setfiletype xxd

" Enter hex mode
command! HexMode :%!xxd
command! HexModeReverse :%!xxd -r

" Auto hex mode for binary files
:autocmd BufReadPost *.bin silent %!xxd
:autocmd BufWritePre *.bin %!xxd -r
:autocmd BufWritePost *.bin silent %!xxd
:autocmd BufReadPost *.bin set filetype=xxd
```

27.2 C/C++ header and implementation switching

Category: File Type Specific
Tags: c, cpp, header, implementation, switch

Navigate between C/C++ header and implementation files efficiently.

Example

```
" Switch between header and implementation
function! SwitchSourceHeader()
  let extension = expand('%:e')
  let base = expand('%:r')

  if extension ==# 'h' || extension ==# 'hpp'
    " Switch to implementation
    for ext in ['c', 'cpp', 'cc', 'cxx']
```

```
      if filereadable(base . '.' . ext)
        execute 'edit ' . base . '.' . ext
        return
      endif
    endfor
  else
    " Switch to header
    for ext in ['h', 'hpp', 'hxx']
      if filereadable(base . '.' . ext)
        execute 'edit ' . base . '.' . ext
        return
      endif
    endfor
  endif
endfunction

:autocmd FileType c,cpp nnoremap <leader>a :call SwitchSourceHeader()<CR>
```

27.3 CSS and SCSS productivity shortcuts

Category: File Type Specific
Tags: css, scss, sass, style, property

Speed up CSS/SCSS development with smart shortcuts and settings.

Example

```
" CSS-specific settings
:autocmd FileType css,scss setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType css,scss setlocal iskeyword+=-

" Quick property completion
:autocmd FileType css inoremap : :<Space>
:autocmd FileType css inoremap ; ;<CR>

" Color hex value highlighting
:autocmd FileType css,scss syntax match cssColor /#\x\{6\}/
```

27.4 Configuration file syntax highlighting

Category: File Type Specific
Tags: config, conf, ini, properties, syntax

Enable proper syntax highlighting for various configuration formats.

Example

```
" Auto-detect config file types
:autocmd BufNewFile,BufRead *.conf setfiletype conf
```

```
:autocmd BufNewFile,BufRead *.ini setfiletype dosini
:autocmd BufNewFile,BufRead *.properties setfiletype jproperties
:autocmd BufNewFile,BufRead .env* setfiletype sh
:autocmd BufNewFile,BufRead *.toml setfiletype toml

" Config file settings
:autocmd FileType conf,dosini setlocal commentstring=#\ %s
:autocmd FileType conf,dosini setlocal tabstop=4 shiftwidth=4 expandtab
```

27.5 Docker and container file editing

Category: File Type Specific
Tags: docker, dockerfile, container, build, syntax

Optimize editing Docker-related files with proper syntax and shortcuts.

Example

```
" Dockerfile settings
:autocmd BufNewFile,BufRead Dockerfile* setfiletype dockerfile
:autocmd BufNewFile,BufRead *.dockerfile setfiletype dockerfile

:autocmd FileType dockerfile setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType dockerfile setlocal commentstring=#\ %s

" Docker build shortcut
:autocmd FileType dockerfile nnoremap <leader>db :!docker build -t my-image
↪  .<CR>

" Quick Dockerfile templates
:autocmd FileType dockerfile nnoremap <leader>df iFROM <CR>RUN <CR>COPY
↪  <CR>CMD
```

27.6 Git commit message formatting

Category: File Type Specific
Tags: git, commit, message, format, conventional

Improve git commit message writing with templates and formatting.

Example

```
" Git commit settings
:autocmd FileType gitcommit setlocal textwidth=72
:autocmd FileType gitcommit setlocal spell spelllang=en_us
:autocmd FileType gitcommit setlocal colorcolumn=50,72

" Start in insert mode for commit messages
:autocmd FileType gitcommit startinsert
```

```
" Conventional commit templates
:autocmd FileType gitcommit nnoremap <leader>gf ifeat:
:autocmd FileType gitcommit nnoremap <leader>gb ifix:
:autocmd FileType gitcommit nnoremap <leader>gd idocs:
:autocmd FileType gitcommit nnoremap <leader>gr irefactor:
```

27.7 Go language specific features

Category: File Type Specific
Tags: go, golang, gofmt, import, build

Configure Go development workflow with formatting and building.

Example

```
" Go-specific settings
:autocmd FileType go setlocal tabstop=4 shiftwidth=4 noexpandtab
:autocmd FileType go setlocal listchars=tab:\ \ ,trail:·

" Go formatting on save
:autocmd BufWritePre *.go lua vim.lsp.buf.format()

" Quick Go build and run
:autocmd FileType go nnoremap <leader>gr :!go run %<CR>
:autocmd FileType go nnoremap <leader>gb :!go build<CR>
:autocmd FileType go nnoremap <leader>gt :!go test<CR>
```

27.8 HTML and XML tag manipulation

Category: File Type Specific
Tags: html, xml, tag, element, markup

Use specialized commands for HTML/XML tag editing and navigation.

Example

```
" Tag matching with %
:autocmd FileType html,xml set matchpairs+=<:>

" Surround word with HTML tags
:autocmd FileType html nnoremap <leader>t ysiw<

" Quick tag completion in insert mode
:autocmd FileType html inoremap <lt>/ </<C-X><C-O>

" Format HTML/XML
:autocmd FileType html,xml nnoremap <leader≥ gg=G
```

27.9 JSON formatting and validation

Category: File Type Specific
Tags: json, format, validate, pretty, minify

Work efficiently with JSON files using formatting and validation tools.

Example

```
" JSON settings
:autocmd FileType json setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType json setlocal conceallevel=0

" Format JSON
:autocmd FileType json nnoremap <leader>jf :%!jq '.'<CR>
:autocmd FileType json nnoremap <leader>jm :%!jq -c '.'<CR>  " minify

" Validate JSON
:autocmd FileType json nnoremap <leader>jv :!jq . % > /dev/null<CR>

" Quick JSON template
:autocmd FileType json nnoremap <leader>jt i{<CR>"key": "value"<CR>}<Esc>
```

27.10 Java class and package navigation

Category: File Type Specific
Tags: java, class, package, import, navigation

Streamline Java development with class and package utilities.

Example

```
" Java settings
:autocmd FileType java setlocal tabstop=4 shiftwidth=4 expandtab
:autocmd FileType java setlocal suffixesadd+=.java

" Quick class template
:autocmd FileType java nnoremap <leader>jc :0r
↪  ~/.config/nvim/templates/java-class.java<CR>

" Import statement insertion
:autocmd FileType java nnoremap <leader>ji ggOimport

" Quick main method
:autocmd FileType java nnoremap <leader>jm ipublic static void main(String[]
↪  args) {<CR>}<Esc>O
```

27.11 JavaScript/TypeScript development setup

Category: File Type Specific
Tags: javascript, typescript, js, ts, node, format

Optimize settings for JavaScript and TypeScript development.

Example

```
" JavaScript/TypeScript settings
:autocmd FileType javascript,typescript setlocal tabstop=2 shiftwidth=2
↪  expandtab
:autocmd FileType javascript,typescript setlocal suffixesadd+=.js,.ts
:autocmd FileType javascript,typescript setlocal
↪  include=^\\s*import\\s*.*\\s*from

" Quick console.log insertion
:autocmd FileType javascript nnoremap <leader>cl oconsole.log();<Left><Left>
:autocmd FileType typescript nnoremap <leader>cl oconsole.log();<Left><Left>
```

27.12 Log file analysis and navigation

Category: File Type Specific
Tags: log, analysis, navigation, search, timestamp

Navigate and analyze log files efficiently with specialized commands.

Example

```
" Log file detection and settings
:autocmd BufNewFile,BufRead *.log setfiletype log
:autocmd FileType log setlocal nowrap
:autocmd FileType log setlocal readonly

" Log navigation shortcuts
:autocmd FileType log nnoremap <leader>le /ERROR<CR>     " find errors
:autocmd FileType log nnoremap <leader>lw /WARN<CR>      " find warnings
:autocmd FileType log nnoremap <leader>lt /\d\{4\}-\d\{2\}-\d\{2\}<CR>  "
↪  find timestamps

" Highlight log levels
:autocmd FileType log syntax match logError /ERROR/
:autocmd FileType log syntax match logWarn /WARN/
:autocmd FileType log syntax match logInfo /INFO/
```

27.13 Lua script configuration

Category: File Type Specific
Tags: lua, script, neovim, config, development

Configure Lua development for Neovim scripting and general development.

Example

```
" Lua settings
:autocmd FileType lua setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType lua setlocal suffixesadd+=.lua

" Quick Neovim Lua testing
:autocmd FileType lua nnoremap <leader>ll :luafile %<CR>
:autocmd FileType lua vnoremap <leader>ll :lua <CR>

" Lua function template
:autocmd FileType lua nnoremap <leader>lf ilocal function ()<CR>end<Esc>kf(a

" Quick vim namespace
:autocmd FileType lua nnoremap <leader>lv ivim.
```

27.14 Markdown writing and formatting

Category: File Type Specific
Tags: markdown, md, writing, format, preview

Enhance Markdown writing experience with formatting and navigation.

Example

```
" Markdown settings
:autocmd FileType markdown setlocal textwidth=80
:autocmd FileType markdown setlocal wrap linebreak
:autocmd FileType markdown setlocal spell spelllang=en_us

" Quick formatting
:autocmd FileType markdown nnoremap <leader>mb ysiw**    " bold
:autocmd FileType markdown nnoremap <leader>mi ysiw*     " italic
:autocmd FileType markdown nnoremap <leader>mc ysiw`     " code

" Header navigation
:autocmd FileType markdown nnoremap <leader>h1 I# <Esc>
:autocmd FileType markdown nnoremap <leader>h2 I## <Esc>
:autocmd FileType markdown nnoremap <leader>h3 I### <Esc>
```

27.15 Python indentation and formatting

Category: File Type Specific
Tags: python, indent, format, pep8, filetype

Configure Python-specific settings for proper indentation and formatting.

Example

```
" Set Python-specific options
:autocmd FileType python setlocal tabstop=4 shiftwidth=4 expandtab
:autocmd FileType python setlocal textwidth=79
:autocmd FileType python setlocal autoindent smartindent

" Python folding
:autocmd FileType python setlocal foldmethod=indent
:autocmd FileType python setlocal foldlevel=2
```

27.16 Rust development optimization

Category: File Type Specific
Tags: rust, cargo, rustfmt, clippy, build

Set up efficient Rust development environment and shortcuts.

Example

```
" Rust settings
:autocmd FileType rust setlocal tabstop=4 shiftwidth=4 expandtab
:autocmd FileType rust setlocal textwidth=100

" Rust cargo commands
:autocmd FileType rust nnoremap <leader>rc :!cargo check<CR>
:autocmd FileType rust nnoremap <leader>rb :!cargo build<CR>
:autocmd FileType rust nnoremap <leader>rr :!cargo run<CR>
:autocmd FileType rust nnoremap <leader>rt :!cargo test<CR>

" Format on save
:autocmd BufWritePre *.rs lua vim.lsp.buf.format()
```

27.17 SQL query formatting and execution

Category: File Type Specific
Tags: sql, query, format, database, execute

Enhance SQL development with formatting and execution capabilities.

Example

```
" SQL settings
:autocmd FileType sql setlocal tabstop=2 shiftwidth=2 expandtab

" SQL keyword formatting (uppercase)
:autocmd FileType sql nnoremap <leader>su
 ↪  :s/\<\(select\|from\|where\|join\|group\|order\|by\)\>/\U&/g<CR>

" Format SQL query
```

```
:autocmd FileType sql nnoremap <leader>sf :%!sqlformat --reindent --keywords
↪  upper --identifiers lower -<CR>

" Quick SQL templates
:autocmd FileType sql nnoremap <leader>ss iSELECT <CR>FROM <CR>WHERE
:autocmd FileType sql nnoremap <leader>si iINSERT INTO  () VALUES ();<Esc>
```

27.18 Shell script development

Category: File Type Specific
Tags: shell, bash, sh, script, executable

Streamline shell script development with proper settings and shortcuts.

Example

```
" Shell script settings
:autocmd FileType sh setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType sh setlocal textwidth=100

" Make executable on save
:autocmd BufWritePost *.sh silent !chmod +x %

" Shell check linting
:autocmd FileType sh nnoremap <leader>sc :!shellcheck %<CR>

" Quick shebang insertion
:autocmd FileType sh nnoremap <leader>sb ggO#!/bin/bash<Esc>
```

27.19 Template file creation

Category: File Type Specific
Tags: template, skeleton, file, creation, boilerplate

Automatically insert templates for new files based on file type.

Example

```
" Template insertion for new files
:autocmd BufNewFile *.py 0r ~/.config/nvim/templates/python.py
:autocmd BufNewFile *.js 0r ~/.config/nvim/templates/javascript.js
:autocmd BufNewFile *.html 0r ~/.config/nvim/templates/html5.html
:autocmd BufNewFile *.css 0r ~/.config/nvim/templates/styles.css
:autocmd BufNewFile *.sh 0r ~/.config/nvim/templates/bash.sh

" Replace template variables
:autocmd BufNewFile * %s/{{FILENAME}}/\=expand('%:t:r')/g
:autocmd BufNewFile * %s/{{DATE}}/\=strftime('%Y-%m-%d')/g
:autocmd BufNewFile * %s/{{AUTHOR}}/\=system('git config user.name')[:-2]/g
```

27.20 XML and configuration file handling

Category: File Type Specific
Tags: xml, config, plist, format, validate

Handle XML and various configuration file formats effectively.

Example

```
" XML settings
:autocmd FileType xml setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType xml setlocal foldmethod=syntax
:autocmd FileType xml setlocal omnifunc=xmlcomplete#CompleteTags

" Format XML
:autocmd FileType xml nnoremap <leader>xf :%!xmllint --format -<CR>

" Validate XML
:autocmd FileType xml nnoremap <leader>xv :!xmllint --noout %<CR>

" Quick CDATA section
:autocmd FileType xml nnoremap <leader>cd i<![CDATA[]]><Esc>3hi
```

27.21 YAML configuration editing

Category: File Type Specific
Tags: yaml, yml, config, indent, validate

Optimize YAML editing with proper indentation and validation.

Example

```
" YAML settings
:autocmd FileType yaml setlocal tabstop=2 shiftwidth=2 expandtab
:autocmd FileType yaml setlocal indentkeys-=<:>
:autocmd FileType yaml setlocal foldmethod=indent

" YAML validation
:autocmd FileType yaml nnoremap <leader>yv :!yamllint %<CR>

" Quick list item
:autocmd FileType yaml nnoremap <leader>yl o-
:autocmd FileType yaml inoremap <C-l> <CR>-
```

CHAPTER 28
Folding

28.1 Create fold from selection

Category: Folding
Tags: fold, create, selection

Use `zf` to create a fold from visual selection or with motion (e.g., `zf5j` to fold 5 lines down).

Example

```
zf5j  " create fold 5 lines down
zf    " create fold from visual selection
```

28.2 Fold by indentation

Category: Folding
Tags: fold, indent, automatic, method

Automatically fold code based on indentation levels using foldmethod=indent.

Example

```
set foldmethod=indent   " fold based on indentation
set foldlevelstart=1    " start with some folds open
set foldnestmax=3       " limit nested fold depth
```

28.3 Fold levels

Category: Folding
Tags: fold, level, depth

Use `zm` to increase fold level (close more folds) and `zr` to reduce fold level (open more folds).

Example

```
zm  " increase fold level
zr  " reduce fold level
```

28.4 Keep folds when inserting

Category: Folding
Tags: fold, insert, preserve, maintain

Configure Vim to maintain fold state when entering insert mode.

Example

```
" Prevent folds from opening when inserting
set foldopen-=insert

" Mapping to toggle fold with F9
nnoremap <F9> za
vnoremap <F9> zf
```

28.5 Open and close all folds

Category: Folding
Tags: fold, all, global

Use `zR` to open all folds in buffer and `zM` to close all folds in buffer.

Example

```
zR  " open all folds
zM  " close all folds
```

28.6 Syntax-based folding

Category: Folding
Tags: fold, syntax, automatic, language

Use syntax-aware folding for programming languages that support fold markers in syntax files.

Example

```
set foldmethod=syntax   " fold based on file syntax
set foldlevel=2         " set initial fold level
```

28.7 Toggle fold

Category: Folding
Tags: fold, toggle, code

Use `za` to toggle fold under cursor open/closed.

Example

```
za  " toggle fold under cursor
```

28.8 Z-commands - create folds

Category: Folding
Tags: fold, create, lines

Use `zF` to create fold for N lines or `zf{motion}` to create fold with motion.

Example

```
5zF   " create fold for 5 lines
zf3j  " create fold from cursor down 3 lines
zfip  " create fold for inner paragraph
```

CHAPTER 29
Formatting

29.1 Automatic paragraph formatting

Category: Formatting
Tags: paragraph, textwidth, reflow

Automatically format paragraphs to specified width using textwidth and format commands.

Example

```
:set textwidth=60      " set line width to 60 characters
:set fo=aw2tq          " format options for auto-formatting
gqip                   " reformat inner paragraph
```

29.2 Automatic text width formatting

Category: Formatting
Tags: text, width, format, autowrap, textwidth

Use `:set textwidth=80` to automatically wrap lines at 80 characters while typing.

Example

```
:set textwidth=80   " wrap at 80 characters
:set textwidth=0    " disable automatic wrapping
:set formatoptions+=t  " enable automatic text wrapping
gqap                " manually format current paragraph to textwidth
```

29.3 Comment lines by filetype

Category: Formatting
Tags: comment, filetype, toggle

Automatically comment/uncomment lines based on current file type.

Example

```
function CommentIt()
  if &filetype == "vim"
    vmap +# :s/^/"/<CR>
    vmap -# :s/^"//<CR>
  elseif &filetype == "python"
    vmap +# :s/^/#/<CR>
    vmap -# :s/^#//<CR>
  endif
endfunction
autocmd BufEnter * call CommentIt()
```

29.4 Format with Treesitter

Category: Formatting
Tags: treesitter, format, syntax

Use `=ap` to format syntax-aware regions using Treesitter (when available).

Example

```
=ap  " format around paragraph with Treesitter
```

29.5 Poor men's JSON formatter

Category: Formatting
Tags: text, format, json

A poor men's json formatter using `vim.json.decode` + `vim.json.encode`:

Example

```
function _G.json_formatter()
        -- from $VIMRUNTIME/lua/vim/lsp.lua
        if vim.list_contains({ 'i', 'R', 'ic', 'ix' }, vim.fn.mode()) then
                return 1
        end
        local indent = vim.bo.expandtab and '\t' or string.rep(' ',
        ↪  vim.o.tabstop)

        local lines = vim.api.nvim_buf_get_lines(0, vim.v.lnum - 1,
        ↪  vim.v.count, true)
        local deco = vim.json.decode(table.concat(lines, '\n'))
        local enco = vim.json.encode(deco, { indent = indent })
        local split = vim.split(enco, '\n')
        vim.api.nvim_buf_set_lines(0, vim.v.lnum - 1, vim.v.count, true,
        ↪  split)

        return 0
```

```
end

vim.bo.formatexpr = 'v:lua.json_formatter()'
```

You can put it in `ftplugin/json.lua`. Only works for the whole file, e.g. with `gggqG`
[Credits: yochem](https://github.com/neovim/neovim/discussions/35683)

CHAPTER 30

Fun

30.1 Flip a coin

Category: Fun
Tags: fun

Quick decision-making inside Neovim. Could help settle coding debates (“Tabs or spaces?”).

Example

```
:echo ["Heads","Tails"][rand() % 2]
```

30.2 Funny event

Category: Fun
Tags: easter egg, fun

As you already know, Neovim emits various events that you can handle with your own code. There is one particular event that is not found in any other app: `UserGettingBored`. To find out more about this event, type:

Example

```
:h UserGettingBored
```

It turns out that this event is not implemented yet, help text is there just for fun. But... if you install plugin [mikesmithgh/ugbi](https://github.com/mikesmithgh/ugbi), you can actually see this event triggered in the funniest way possible. Plugin description is also a masterpiece on its own. Definitely the best plugin ever, especially in the ”Useless” category.

30.3 Help!

Category: Fun
Tags: easter egg, fun

If you are really, really desperate:

Example

```
:help!
```

30.4 Holy Grail

Category: Fun
Tags: easter egg, fun

Find the Holy Grail by typing the following command:

Example

```
:help holy-grail
```

30.5 Matrix like effect

Category: Fun
Tags: fun

Poor man's matrix screen :)

Example

```
:!yes | awk '{print int(rand()*10)}' | pv -qL 100
```

30.6 Random quote generator:

Category: Fun
Tags: fun, file

Paste random quote into your buffer with the following command:

Example

```
:lua local q = vim.fn.readfile("quotes.txt"); vim.api.nvim_buf_set_lines(0,
↪  vim.api.nvim_win_get_cursor(0)[1], vim.api.nvim_win_get_cursor(0)[1],
↪  false, { q[math.random(#q)] })
```

30.7 Reverse lines in file

Category: Fun
Tags: fun, edit, reverse

Moves every line to the top — effectively reversing the buffer. Great for experimenting or trolling your own file.

Example

```
:g/^/m0
```

30.8 Show all whitespace, but nicely

Category: Fun
Tags: fun

Use the following commands:

Example

```
:highlight ExtraWhitespace ctermbg=red guibg=red
:match ExtraWhitespace /\s\+$/
```

Highlights trailing spaces in red. Instant "why did I leave that space there?" effect.

30.9 Shuffle lines

Category: Fun
Tags: fun, shuffle, edit

Use the following commands to randomly rearrange all lines:

Example

```
:lua local lines=vim.api.nvim_buf_get_lines(0,0,-1,false); for i=#lines,2,-1
↪  do local j=math.random(i); lines[i],lines[j]=lines[j],lines[i]; end;
↪  vim.api.nvim_buf_set_lines(0,0,-1,false,lines)
```

Perfect for testing or chaos.

30.10 Smile!

Category: Fun
Tags: easter egg, fun

Check this easter egg:

Example

```
:smile
```

30.11 Sort randomly

Category: Fun
Tags: fun, sort

Sort lines randomly by using this simple command:

Example

```
:sort!
```

Works for small files only.

30.12 Speaking French?

Category: Fun
Tags: easter egg, fun

Well, translate this:

Example

```
:h |
```

30.13 Surprise yourself!

Category: Fun
Tags: fun

The following command will print `Neovim is great` until you stop it with `Ctrl+C`:

Example

```
:!yes "Neovim is great"
```

30.14 What is the meaning of life, the universe and everything

Category: Fun
Tags: easter egg, fun

Find the answer by typing the following command:

Example

```
:help 42
```

CHAPTER 31

G commands

31.1 Change case with gu and gU

Category: Text
Tags: g-commands, case, uppercase, lowercase

Use `gu` and `gU` to change the case of text with motion or in visual mode.

Example

```
" In normal mode:
guw   " lowercase word
gUw   " uppercase word
guiw  " lowercase inner word
gU$   " uppercase to end of line

" In visual mode:
gu    " lowercase selection
gU    " uppercase selection
```

31.2 Edit file under cursor with gf

Category: Navigation
Tags: g-commands, file, navigation, cursor

Use `gf` to edit the file whose path is under the cursor. Useful for navigating to files in code.

Example

```
" Place cursor on a file path and press:
gf  " open file in current window

" Related commands:
CTRL-W f  " open file in new window
CTRL-W gf " open file in new tab
```

31.3 Format text with gq

Category: Formatting
Tags: g-commands, format, text, wrap

Use `gq` to format text according to 'textwidth' and other formatting options.

Example

```
gqap  " format around paragraph
gqq   " format current line
gq}   " format until end of paragraph

" In visual mode:
gq    " format selection
```

31.4 Format without moving cursor with gw

Category: Formatting
Tags: g-commands, format, cursor, text

Use `gw` to format text like `gq` but keep the cursor position.

Example

```
gwap  " format around paragraph, cursor stays
gww   " format current line, cursor stays
```

31.5 Go to previous/next tab with gT and gt

Category: Tabs
Tags: g-commands, tabs, navigation

Use `gt` and `gT` to navigate between tabs.

Example

```
gt   " go to next tab
gT   " go to previous tab
{n}gt " go to tab number n
```

31.6 Increment numbers in sequence with g CTRL-A

Category: Editing
Tags: g-commands, numbers, increment, sequence

Use `g CTRL-A` in visual mode to increment numbers sequentially (1, 2, 3, ...).

Example

```
" Select multiple lines with same number:
1
1
1

" Press g CTRL-A to get:
1
2
3

" Similarly, g CTRL-X decrements sequentially
```

31.7 Insert at line start with gI

Category: Insert
Tags: g-commands, insert, column, line-start

Use `gI` to insert text at the first column of the line (before any indentation).

Example

```
gI  " insert at column 1 (ignoring indentation)
I   " insert before first non-blank character (respects indentation)
```

31.8 Join lines without spaces with gJ

Category: Text
Tags: g-commands, join, lines

Use `gJ` to join lines without inserting spaces between them.

Example

```
gJ   " join current line with next, no space added
J    " join with space added (standard join)
```

31.9 Move to middle of screen line with gm

Category: Navigation
Tags: g-commands, navigation, screen, middle

Use `gm` to move the cursor to the middle of the screen line.

Example

```
gm  " jump to middle of screen line
```

31.10 Move to screen line positions with g0, g^, g$

Category: Navigation
Tags: g-commands, navigation, screen-lines, position

Use `g0`, `g^`, and `g$` to move to positions within screen lines (for wrapped text).

Example

```
g0  " move to first character of screen line
g^  " move to first non-blank of screen line
g$  " move to last character of screen line

" Compare with regular commands:
0   " first character of actual line
^   " first non-blank of actual line
$   " last character of actual line
```

31.11 Navigate screen lines with gj and gk

Category: Navigation
Tags: g-commands, navigation, screen-lines, wrapped

Use `gj` and `gk` to move by screen lines instead of actual lines (useful for wrapped text).

Example

```
gj  " move down one screen line
gk  " move up one screen line

" Compare with:
j   " move down one actual line
k   " move up one actual line
```

31.12 Open URL or file with gx

Category: Navigation
Tags: g-commands, url, file, open, external

Use `gx` to open the file or URL under the cursor with the system's default application.

Example

```
" Place cursor on URL or file path:
gx  " open with default application (browser for URLs, etc.)
```

31.13 Repeat last command with g.

Category: Editing
Tags: g-commands, repeat, command, redo

Use `g.` to jump to the position of the last change.

Example

```
g;  " go to position of previous change (older)
g,  " go to position of next change (newer)
```

31.14 Reselect last visual selection with gv

Category: Visual
Tags: g-commands, visual, selection, reselect

Use `gv` to reselect the last visual selection. Useful for reapplying operations.

Example

```
" After making a visual selection and returning to normal mode:
gv  " reselect the same area
```

31.15 Return to last insert position with gi

Category: Navigation
Tags: g-commands, insert, navigation, jump

Use `gi` to jump to the last position where you were in INSERT mode and enter INSERT mode.

Example

```
gi  " jump to last insertion point and enter INSERT mode
```

31.16 Select last search match with gn

Category: Search
Tags: g-commands, search, visual, selection

Use `gn` to visually select the next match of the last search pattern. Powerful for targeted changes.

Example

```
" After searching with /pattern:
gn    " select next match
cgn   " change next match (then use . to repeat)
dgn   " delete next match
```

31.17 Show ASCII value with ga

Category: Utilities
Tags: g-commands, ascii, character, info

Use `ga` to display the ASCII/Unicode value of the character under the cursor.

Example

```
" Place cursor on a character:
ga  " shows decimal, hex, and octal values
```

31.18 Swap character case with g~

Category: Text
Tags: g-commands, case, swap, toggle

Use `g~` to swap the case of text (uppercase ↔ lowercase).

Example

```
g~w   " swap case of word
g~iw  " swap case of inner word
g~$   " swap case to end of line

" In visual mode:
g~    " swap case of selection
```

CHAPTER 32

Global

32.1 Global command - advanced patterns

Category: Global
Tags: global, regex, pattern, advanced

Combine global command with advanced patterns for complex operations.

Example

```
:g/^\s*$/d                " delete all blank lines (whitespace only)
:g/pattern1/.,/pattern2/d " delete from pattern1 to pattern2
:g/^/m0                   " reverse all lines in file
:g/^/t.                   " duplicate every line
```

32.2 Global command - copy matching lines

Category: Global
Tags: global, copy, pattern, duplicate

Use `:g` with `:t` (copy) to duplicate matching lines to a location.

Example

```
:g/error/t$               " copy all error lines to end of file
:g/TODO/t0                " copy all TODO lines to top of file
```

32.3 Global command - execute on matching lines

Category: Global
Tags: global, command, pattern, ex

Use `:g/pattern/command` to execute a command on all lines matching a pattern. One of Vim's most powerful features.

Example

```
:g/TODO/d           " delete all lines containing TODO
:g/function/p       " print all lines with 'function'
:g/error/norm gUU   " uppercase lines containing 'error'
:g!/pattern/d       " delete lines NOT matching pattern (inverse)
:v/pattern/d        " same as :g! (v for inVerse)
```

32.4 Global command - move matching lines

Category: Global
Tags: global, move, pattern, reorder

Use `:g` with `:m` to move all matching lines to a specific location.

Example

```
:g/TODO/m$                " move all lines with TODO to end of file
:g/^import/m0             " move all import lines to top of file
:g/function/m'a           " move all lines with 'function' to mark 'a'
```

32.5 Global command with line ranges

Category: Global
Tags: global, range, lines, pattern

Combine `:g` with line ranges to limit where the global command operates.

Example

```
:10,50g/pattern/d         " delete matching lines only in range 10-50
:'<,'>g/pattern/norm A!   " append ! to matching lines in visual selection
:1,$g/^$/d                " delete all empty lines in file
```

32.6 Global command with normal mode commands

Category: Global
Tags: global, normal, command, pattern

Use `:g/pattern/norm <commands>` to execute normal mode commands on matching lines.

Example

```
:g/TODO/norm A !!!          " append !!! to lines with TODO
:g/^#/norm >>               " indent all lines starting with #
:g/function/norm dwelp      " swap first two words on lines with 'function'
```

```
:g/console\.log/norm gcc    " comment out console.log lines (with
↪  commentary.vim)
```

32.7 Open documentation for word under the cursor

Category: Global
Tags: man pages, documentation, help

Use `K` to open a man page or other type of available documentation for the word under the cursor.

Example

```
K
```

32.8 Open terminal

Category: Global
Tags: terminal

Use `:ter[minal]` to open a terminal window. When the window shows up, press `i` to enter the insert mode and start typing shell commands. Type `exit` to close the terminal window.

TIP: Once in terminal, type `vimtutor` for a nice vim tutorial, excellent for starters.

Example

```
:ter
```

CHAPTER 33

Help

33.1 Ex commands - help and documentation

Category: Help
Tags: ex, help, documentation, version, info

Use `:version` for version info, `:intro` for intro message, `:messages` for message history, `:checkhealth` for diagnostics.

Example

```
:version      " show Neovim version and build info
:intro        " show introduction message
:messages     " show message history
:checkhealth  " run health diagnostics
```

33.2 Ex commands - help navigation

Category: Help
Tags: ex, help, navigation, tag, jump

Use `:helpgrep` to search help, `:ptag` for preview, `:pop` to go back, `:tag` to jump to tag.

Example

```
:helpgrep pattern   " search all help for pattern
:ptag function      " preview tag in preview window
:pop                " go back in tag stack
:tag function       " jump to tag
:tags               " show tag stack
```

33.3 Man pages

Category: Help
Tags: man, help, documentation

Use `:Man` to open a man-page in a read-only mode. You can navigate the page like any

other Neovim content. You can jump arround, copy parts of it, search for a word and perform all other standard Neovim operations. Once you have a man-page on the screen, you can generate its content by using stadnard `gO` command in normal mode.

Example

```
:Man ls   "Opens man-page for linux ls command
gO        "Generates TOC in normal mode for the selected man-page
```

33.4 Master help index

Category: Help
Tags: help, index, reference

Use `:h index.txt` to access the master help index with all available commands.

Example

```
:h index.txt  " master help index
```

33.5 Search help by pattern

Category: Help
Tags: help, search, pattern

Use `:help pattern` to search help documentation for specific keywords or patterns.

Example

```
:help pattern  " search help for 'pattern'
```

CHAPTER 34

Indentation

34.1 Auto indent

Category: Indentation
Tags: indent, auto, format

Use == to auto-indent current line, or {number}== to auto-indent multiple lines.

Example

```
==   " auto-indent current line
5==  " auto-indent 5 lines
```

34.2 Indent lines

Category: Indentation
Tags: indent, shift, format

Use >> to indent current line right, << to indent left, or use with numbers for multiple lines.

Example

```
>>   " indent line right
<<   " indent line left
3>>  " indent 3 lines right
```

CHAPTER 35

Insert

35.1 Adjust indentation in insert mode

Category: Insert
Tags: indent, indentation, shift

Use `Ctrl+t` to add one shiftwidth of indentation and `Ctrl+d` to remove one shiftwidth of indentation while in insert mode.

Example

```
" In insert mode:
Ctrl+t  " increase indent
Ctrl+d  " decrease indent
```

35.2 Control undo granularity in insert mode

Category: Insert
Tags: undo, granularity, control

Use `Ctrl+g u` to start a new undoable edit and `Ctrl+g U` to prevent the next cursor movement from breaking the undo sequence.

Example

```
" In insert mode:
Ctrl+g u  " start new undo block
Ctrl+g U  " don't break undo with next movement
```

35.3 Copy character from line above/below

Category: Insert
Tags: copy, character, above, below

Use `Ctrl+y` to copy the character above the cursor and `Ctrl+e` to copy the character below the cursor while in insert mode.

Example

```
" In insert mode:
Ctrl+y  " copy character from line above
Ctrl+e  " copy character from line below
```

35.4 Exit insert mode alternatives

Category: Insert
Tags: exit, escape, mode

Use `Ctrl+c` to quit insert mode without checking abbreviations, or `Ctrl+[` as an alternative to Escape key.

Example

```
" In insert mode:
Ctrl+c  " quit insert mode (no abbreviation check)
Ctrl+[  " same as Escape key
```

35.5 Insert above cursor

Category: Insert
Tags: insert, above, cursor

Use `gO` to insert a line above current line without moving cursor position, useful for adding code above current line.

Example

```
gO  " insert line above without moving cursor
```

35.6 Insert calculation result

Category: Insert
Tags: calculation, expression, register

Use `Ctrl+r =` to insert the result of an expression calculation in insert mode.

Example

```
" In insert mode:
Ctrl+r =2+3<Enter>  " inserts 5
Ctrl+r =strftime("%Y-%m-%d")<Enter>  " inserts current date
```

35.7 Insert character by decimal value

Category: Insert
Tags: character, decimal, value, ascii, unicode

Use `Ctrl+v` followed by decimal numbers to insert characters by their ASCII/Unicode decimal value.

Example

```
" In insert mode:
Ctrl+v 65    " insert 'A' (ASCII 65)
Ctrl+v 169   " insert '©' (copyright symbol)
Ctrl+v 8364  " insert '€' (euro symbol)
```

35.8 Insert digraphs

Category: Insert
Tags: digraph, special, characters, unicode

Use `Ctrl+k` followed by two characters to insert digraphs (special characters). Use `:digraphs` to see available combinations.

Example

```
" In insert mode:
Ctrl+k a:    " insert ä
Ctrl+k <<    " insert «
:digraphs    " show all digraphs
```

35.9 Insert mode completion

Category: Insert
Tags: completion, autocomplete, popup

Use `Ctrl+n` for next completion match and `Ctrl+p` for previous completion match. Use `Ctrl+x Ctrl+f` for filename completion.

Example

```
" In insert mode:
Ctrl+n        " next completion
Ctrl+p        " previous completion
Ctrl+x Ctrl+f " filename completion
```

35.10 Insert mode completion subcommands

Category: Insert
Tags: completion, submode, advanced

After `Ctrl+x`, use `Ctrl+d` for defined identifiers, `Ctrl+f` for filenames, `Ctrl+e` to scroll up in completion menu, `Ctrl+y` to scroll down.

Example

```
" In insert mode:
Ctrl+x Ctrl+d  " complete defined identifiers
Ctrl+x Ctrl+f  " complete filenames
Ctrl+x Ctrl+e  " scroll up in completion
Ctrl+x Ctrl+y  " scroll down in completion
```

35.11 Insert mode cursor movement with insertion point

Category: Insert
Tags: cursor, movement, insertion, point

Use `Ctrl+g j` to move cursor down to the column where insertion started and `Ctrl+g k` to move cursor up to that column.

Example

```
" In insert mode:
Ctrl+g j  " move down to insertion start column
Ctrl+g k  " move up to insertion start column
```

35.12 Insert mode line break

Category: Insert
Tags: line, break, split

Use `Ctrl+j` or `Ctrl+m` to create a new line in insert mode (equivalent to pressing Enter).

Example

```
" In insert mode:
Ctrl+j  " new line
Ctrl+m  " new line (alternative)
```

35.13 Insert tab character alternatives

Category: Insert
Tags: tab, character, indent

Use `Ctrl+i` as an alternative to the Tab key for inserting tab characters in insert mode.

Example

```
" In insert mode:
Ctrl+i  " insert tab character (same as Tab key)
```

35.14 New line insertion

Category: Insert
Tags: insert, line, editing

Use `o` to open new line below current line and `O` to open new line above current line.

Example

```
o  " new line below
O  " new line above
```

35.15 Paste in insert mode

Category: Insert
Tags: paste, insert, register, clipboard

Use `Ctrl+r "` to paste from default register, or `Ctrl+r a` to paste from register 'a' while in insert mode.

Example

```
" In insert mode:
Ctrl+r "     " paste from default register
Ctrl+r a     " paste from register 'a'
```

35.16 Paste in insert mode with register

Category: Insert
Tags: paste, insert, register, yank

Use `Ctrl+r 0` to paste yanked text in insert mode, or `Ctrl+r "` for default register.

Example

```
" In insert mode:
Ctrl+r 0  " paste from yank register
Ctrl+r "  " paste from default register
Ctrl+r +  " paste from system clipboard
```

35.17 Repeat last inserted text

Category: Insert
Tags: repeat, insert, text, previous

Use `Ctrl+a` to insert previously inserted text, or `Ctrl+@` to insert previously inserted text and immediately exit insert mode.

Example

```
" In insert mode:
Ctrl+a  " insert previously typed text
Ctrl+@  " insert previous text and exit insert mode
```

35.18 Replace mode

Category: Insert
Tags: replace, overwrite, mode

Use `R` to enter replace mode where typed characters overwrite existing text. Use `gR` for virtual replace mode.

Example

```
R   " enter replace mode
gR  " enter virtual replace mode
```

35.19 Scroll window in insert mode

Category: Insert
Tags: scroll, window, view, insert

Use `Ctrl+x Ctrl+e` to scroll the window down and `Ctrl+x Ctrl+y` to scroll the window up without leaving insert mode.

Example

```
" In insert mode:
Ctrl+x Ctrl+e  " scroll window down
```

```
Ctrl+x Ctrl+y  " scroll window up
```

35.20 Trigger abbreviation manually

Category: Insert
Tags: abbreviation, trigger, expand

Use `Ctrl+]` to manually trigger abbreviation expansion in insert mode.

Example

```
" In insert mode (after setting abbreviation):
:iab teh the
teh<Ctrl+]>  " expands to 'the'
```

CHAPTER 36

Insert mode (advanced)

36.1 Advanced completion modes

Category: Insert Mode Advanced
Tags: completion, advanced, keyword, line

Use specialized completion commands for different contexts.

Example

```
" In insert mode:
Ctrl+x Ctrl+l   " complete whole lines
Ctrl+x Ctrl+n   " complete keywords in current file
Ctrl+x Ctrl+k   " complete from dictionary
Ctrl+x Ctrl+t   " complete from thesaurus
Ctrl+x Ctrl+f   " complete file names
Ctrl+x Ctrl+i   " complete from included files
Ctrl+x Ctrl+]   " complete from tags
Ctrl+x Ctrl+s   " spelling suggestions
Ctrl+x Ctrl+u   " user defined completion
Ctrl+x Ctrl+v   " vim command-line completion
Ctrl+x Ctrl+o   " omni completion
```

36.2 Completion menu navigation

Category: Insert Mode Advanced
Tags: completion, menu, navigate, popup

Navigate and control completion popup menus effectively.

Example

```
" When completion menu is open:
Ctrl+n          " next item in menu
Ctrl+p          " previous item in menu
Ctrl+y          " accept current selection
Ctrl+e          " close menu without selecting
<Enter>         " accept current selection
<Esc>           " close menu and exit insert mode
```

36.3 Delete operations in insert mode

Category: Insert Mode Advanced
Tags: delete, backspace, word, line

Use various delete operations without leaving insert mode.

Example

```
" In insert mode:
<BS>            " delete character before cursor
<Del>           " delete character under cursor
Ctrl+h          " delete character before cursor (same as backspace)
Ctrl+w          " delete word before cursor
Ctrl+u          " delete from cursor to beginning of line
```

36.4 Insert mode abbreviation control

Category: Insert Mode Advanced
Tags: abbreviation, control, expand, prevent

Control abbreviation expansion in insert mode.

Example

```
" In insert mode:
Ctrl+v <Space>  " insert space without expanding abbreviation
Ctrl+]          " manually expand abbreviation
```

After setting `:iabbrev teh the`, typing "teh " expands to "the ".

36.5 Insert mode buffer operations

Category: Insert Mode Advanced
Tags: buffer, file, operation, switch

Perform buffer operations without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o :w       " save current buffer
Ctrl+o :e       " reload current buffer
Ctrl+o Ctrl+^   " switch to alternate buffer
Ctrl+r Ctrl+f   " insert filename under cursor
```

36.6 Insert mode case conversion

Category: Insert Mode Advanced
Tags: case, conversion, upper, lower, toggle

Convert case of text without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o ~        " toggle case of character under cursor
Ctrl+o guw      " lowercase word under cursor
Ctrl+o gUw      " uppercase word under cursor
Ctrl+o g~w      " toggle case of word under cursor
```

36.7 Insert mode folding control

Category: Insert Mode Advanced
Tags: fold, unfold, toggle, code

Control code folding without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o za       " toggle fold at cursor
Ctrl+o zo       " open fold at cursor
Ctrl+o zc       " close fold at cursor
Ctrl+o zR       " open all folds
Ctrl+o zM       " close all folds
```

36.8 Insert mode formatting and alignment

Category: Insert Mode Advanced
Tags: format, align, text, paragraph

Format and align text without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o gqap     " format around paragraph
Ctrl+o =ap      " indent around paragraph
Ctrl+o >ap      " increase indent of paragraph
Ctrl+o <ap      " decrease indent of paragraph
```

36.9 Insert mode macro operations

Category: Insert Mode Advanced
Tags: macro, record, replay, register

Work with macros without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o @a       " replay macro 'a'
Ctrl+o @@       " replay last macro
Ctrl+o qa       " start recording macro (then exit insert)
```

Note: Recording typically requires exiting insert mode first.

36.10 Insert mode marks and jumps

Category: Insert Mode Advanced
Tags: mark, jump, position, navigation

Set marks and jump to positions without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o ma       " set mark 'a' at current position
Ctrl+o 'a       " jump to line of mark 'a'
Ctrl+o `a       " jump to exact position of mark 'a'
Ctrl+o ''       " jump to position before last jump
```

36.11 Insert mode navigation without exiting

Category: Insert Mode Advanced
Tags: navigation, cursor, movement, arrow

Use arrow keys or `Ctrl+h` (left), `Ctrl+j` (down), `Ctrl+k` (up), `Ctrl+l` (right) to navigate in insert mode.

Example

```
" In insert mode:
<Left>/<Right>  " move by character
<Up>/<Down>     " move by line
Ctrl+h          " move left (backspace)
Ctrl+l          " move right
```

36.12 Insert mode register shortcuts

Category: Insert Mode Advanced
Tags: register, shortcut, special, paste

Use special register shortcuts for common insert operations.

Example

```
" In insert mode:
Ctrl+r %        " insert current filename
Ctrl+r #        " insert alternate filename
Ctrl+r :        " insert last command line
Ctrl+r /        " insert last search pattern
Ctrl+r .        " insert last inserted text
```

36.13 Insert mode search operations

Category: Insert Mode Advanced
Tags: search, find, pattern, navigate

Perform searches without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o /pattern " search forward for pattern
Ctrl+o ?pattern " search backward for pattern
Ctrl+o n        " repeat last search
Ctrl+o N        " repeat last search in opposite direction
Ctrl+o *        " search for word under cursor
```

36.14 Insert mode terminal integration

Category: Insert Mode Advanced
Tags: terminal, command, external, shell

Execute external commands and insert their output.

Example

```
" In insert mode:
Ctrl+r !date    " insert output of date command
Ctrl+r !whoami  " insert current username
Ctrl+r !pwd     " insert current directory
Ctrl+o :r !ls   " read output of ls into buffer
```

36.15 Insert mode text objects

Category: Insert Mode Advanced
Tags: text, object, change, delete

Use text objects with `Ctrl+o` to operate on text without leaving insert mode.

Example

```
" In insert mode:
Ctrl+o diw      " delete inner word
Ctrl+o ciw      " change inner word (stay in insert)
Ctrl+o yiw      " yank inner word
Ctrl+o daw      " delete a word (including spaces)
```

36.16 Insert mode window operations

Category: Insert Mode Advanced
Tags: window, scroll, operation, view

Use window operations without leaving insert mode.

Example

```
" In insert mode:
Ctrl+x Ctrl+e   " scroll window down one line
Ctrl+x Ctrl+y   " scroll window up one line
Ctrl+o zz       " center current line
Ctrl+o zt       " move current line to top
Ctrl+o zb       " move current line to bottom
```

36.17 Line movement in insert mode

Category: Insert Mode Advanced
Tags: line, movement, beginning, end

Use `Ctrl+o` to execute one normal command, then return to insert mode at same position.

Example

```
" In insert mode:
Ctrl+o ^        " go to first non-blank character
Ctrl+o $        " go to end of line
Ctrl+o gg       " go to first line
Ctrl+o G        " go to last line
```

36.18 Literal character insertion

Category: Insert Mode Advanced
Tags: literal, character, special, escape

Use `Ctrl+v` to insert special characters literally in insert mode.

Example

```
" In insert mode:
Ctrl+v <Tab>    " insert literal tab character
Ctrl+v <Esc>    " insert literal escape character
Ctrl+v <Enter>  " insert literal newline
Ctrl+v Ctrl+m   " insert carriage return
```

36.19 Smart indentation in insert mode

Category: Insert Mode Advanced
Tags: indent, smart, automatic, programming

Control automatic indentation behavior in insert mode.

Example

```
" In insert mode:
Ctrl+f          " re-indent current line
0 Ctrl+d        " remove all indent from current line
^ Ctrl+d        " remove indent to previous shiftwidth
Ctrl+o >>       " indent current line
Ctrl+o <<       " unindent current line
```

36.20 Temporary normal mode from insert mode

Category: Insert Mode Advanced
Tags: normal, mode, temporary, <C-o>, command

Use `<C-o>` in insert mode to execute a single normal mode command and return immediately to insert mode.

Example

```
" In insert mode:
<C-o>dd        " delete current line and return to insert
<C-o>j         " move down one line and continue inserting
<C-o>ciw       " change inner word and stay in insert mode
<C-o>A         " go to end of line and continue inserting
```

36.21 Word movement in insert mode

Category: Insert Mode Advanced
Tags: word, movement, navigation, shift

Use `Shift+Left` and `Shift+Right` to move by words in insert mode.

Example

```
" In insert mode:
Shift+Left      " move to beginning of previous word
Shift+Right     " move to beginning of next word
Ctrl+Left       " alternative word movement left
Ctrl+Right      " alternative word movement right
```

CHAPTER 37

Integration tips

37.1 API and webhook integration

Category: Integration
Tags: api, webhook, http, rest, automation

Integrate with APIs and webhooks for automation and data exchange.

Example

```
" Webhook notifications
function! NotifyWebhook(message)
  let payload = '{"text": "' . a:message . '"}'
  call system('curl -X POST -H "Content-Type: application/json" -d '
    \ . shellescape(payload) . ' https://hooks.example.com/webhook')
endfunction

" API data fetching
:r !curl -s "https://api.github.com/repos/owner/repo/issues"
:r !curl -H "Authorization: token TOKEN" "https://api.github.com/user"

" Auto-update from API
autocmd BufWritePost *.md call NotifyWebhook("Documentation updated")
```

37.2 Browser and documentation integration

Category: Integration
Tags: browser, documentation, help, external, web

Open external documentation and web resources from Neovim.

Example

```
" Open URLs under cursor
:!open <cfile>              " macOS
:!xdg-open <cfile>          " Linux
:!start <cfile>             " Windows

" Language-specific documentation
:!open https://docs.python.org/3/search.html?q=<cword>  " Python docs
```

```
:!firefox "https://developer.mozilla.org/en-US/search?q=<cword>"  " MDN docs

" Custom function for smart URL opening
function! OpenURL()
  let url = expand('<cfile>')
  if url =~ '^https\?://'
    execute '!open ' . shellescape(url)
  else
    echo "Not a valid URL"
  endif
endfunction
```

37.3 Build system integration

Category: Integration
Tags: build, make, cmake, gradle, maven, build

Integrate with various build systems and compilation tools.

Example

```
" Make integration
:make                     " run make with error parsing
:make clean               " clean build
:make install             " install build

" CMake integration
:!cmake -B build .        " configure build
:!cmake --build build     " compile
:!ctest --test-dir build  " run tests

" Gradle/Maven integration
:!./gradlew build         " gradle build
:!mvn compile test        " maven compile and test
```

37.4 Cloud platform integration

Category: Integration
Tags: cloud, aws, gcp, azure, kubectl, terraform

Integrate with cloud platforms and infrastructure tools.

Example

```
" Kubernetes integration
:!kubectl get pods
:!kubectl logs pod-name -f  " follow logs
:!kubectl describe pod pod-name
```

```
" AWS CLI integration
:!aws s3 ls s3://bucket-name
:!aws logs tail /aws/lambda/function-name --follow

" Terraform integration
:!terraform plan
:!terraform apply
```

37.5 Continuous Integration integration

Category: Integration
Tags: ci, cd, github, actions, jenkins, pipeline

Integrate with CI/CD pipelines and automation systems.

Example

```
" GitHub Actions
:e .github/workflows/main.yml
:!gh workflow list         " list workflows
:!gh run list              " list workflow runs

" Jenkins integration
:!curl -X POST http://jenkins.local/job/my-job/build
:r !curl -s http://jenkins.local/job/my-job/lastBuild/api/json

" GitLab CI integration
:e .gitlab-ci.yml
:!curl --header "PRIVATE-TOKEN: token" \
  "https://gitlab.com/api/v4/projects/ID/pipelines"
```

37.6 Database integration and querying

Category: Integration
Tags: database, sql, query, connection, dadbod

Connect to and query databases directly from Neovim.

Example

```
" Using vim-dadbod for database operations
:DB postgresql://user:pass@localhost/dbname
SELECT * FROM users;

" Execute SQL from buffer
:%DB                        " execute entire buffer
:'<,'>DB                   " execute visual selection

" Save connection for reuse
```

```
:let g:db = 'postgresql://localhost/mydb'
:DB g:db SELECT COUNT(*) FROM products;
```

37.7 Development server integration

Category: Integration
Tags: server, development, hot, reload, livereload

Integrate with development servers and hot reload systems.

Example

```
" Start development servers
:!npm start &             " start Node.js server in background
:!python -m http.server 8000 &  " start Python HTTP server
:!php -S localhost:8000 & " start PHP development server

" Auto-reload on save
autocmd BufWritePost *.js !kill -USR2 $(cat .pid)  " reload Node.js
autocmd BufWritePost *.py !touch /tmp/reload      " trigger reload

" LiveReload integration
:!livereload --wait 200 --extraExts 'vue,jsx'
```

37.8 Docker and container integration

Category: Integration
Tags: docker, container, dockerfile, build, run

Integrate Docker operations with Neovim development workflow.

Example

```
" Docker build and run
:!docker build -t myapp .
:!docker run -it myapp
:!docker ps                " list running containers

" Docker compose integration
:!docker-compose up -d      " start services
:!docker-compose logs -f web " follow logs

" Edit files in running container
:!docker exec -it container_name vi /app/config.yml
```

37.9 Documentation generation integration

Category: Integration
Tags: documentation, generate, doxygen, sphinx, javadoc

Integrate documentation generation tools with editing workflow.

Example

```
" Doxygen integration
:!doxygen Doxyfile        " generate documentation
:!open docs/html/index.html  " view generated docs

" Sphinx integration (Python)
:!make html               " build Sphinx docs
:!sphinx-autobuild . _build/html  " auto-rebuild on changes

" JSDoc integration
:!jsdoc -d docs/ src/     " generate JavaScript docs
:!npm run docs            " run documentation build script
```

37.10 Email and notification integration

Category: Integration
Tags: email, notification, alert, smtp, webhook

Send notifications and emails from Neovim for workflow automation.

Example

```
" Send email notifications
:!echo "Build completed" | mail -s "Build Status" user@example.com
:!curl -X POST -H 'Content-type: application/json' \
  --data '{"text":"Deployment finished"}' \
  https://hooks.slack.com/webhook-url

" Desktop notifications
:!notify-send "Neovim" "Operation completed"  " Linux
:!osascript -e 'display notification "Done" with title "Neovim"'  " macOS
```

37.11 External editor integration

Category: Integration
Tags: editor, external, gui, comparison, merge

Integrate with external editors for specific tasks and workflows.

Example

```
" Open in external editor
:!code %                    " VS Code
:!subl %                    " Sublime Text
:!atom %                    " Atom

" Merge tool integration
:!meld % file2.txt          " visual diff/merge
:!vimdiff % backup/%        " vim diff mode

" GUI text editor for complex formatting
:!libreoffice --writer %    " word processing
:!typora %                  " markdown editor
```

37.12 Git integration and workflow

Category: Integration
Tags: git, version, control, fugitive, workflow

Integrate Git operations seamlessly with Neovim editing workflow.

Example

```
" Git status and operations
:!git status                 " check git status
:!git add %                 " add current file
:!git commit -m "message"   " commit with message
:!git diff %                " diff current file

" Using terminal integration
:terminal git log --oneline " view git log in terminal
:terminal git commit        " interactive commit
```

37.13 Git integration with gitsigns plugin

Category: Integration
Tags: git, gitsigns, diff, blame, plugin

Use Gitsigns plugin commands for advanced git integration directly in Neovim.

Example

```
" Diff current buffer against previous version
:Gitsigns diffthis ~1

" Toggle git blame for current line
:Gitsigns toggle_current_line_blame

" Preview hunk under cursor
```

```
:Gitsigns preview_hunk

" Stage current hunk
:Gitsigns stage_hunk
```

Note: Requires gitsigns.nvim plugin to be installed.

37.14 Issue tracking integration

Category: Integration
Tags: issue, tracking, jira, github, gitlab, bug

Integrate with issue tracking systems for development workflow.

Example

```
" GitHub Issues integration
:!gh issue list           " list open issues
:!gh issue create --title "Bug report" --body "Description"
:!gh issue close 123      " close issue

" JIRA integration (via CLI)
:!jira list               " list issues
:!jira create --project=PROJ --type=Bug --summary="Title"

" Custom issue templates
:r ~/.config/nvim/templates/bug-report.md
:r ~/.config/nvim/templates/feature-request.md
```

37.15 Monitoring and logging integration

Category: Integration
Tags: monitoring, logging, logs, metrics, observability

Integrate with monitoring and logging systems for development insights.

Example

```
" Log file monitoring
:!tail -f /var/log/app.log  " follow log file
:terminal tail -f logs/production.log

" Grep through logs
:!grep ERROR /var/log/app.log | tail -20
:!journalctl -u service-name -f  " systemd logs

" Custom log analysis
function! AnalyzeLogs()
  :r !grep -c ERROR logs/*.log
```

```
    :r !grep -c WARN logs/*.log
  endfunction
```

37.16 Package manager integration

Category: Integration
Tags: package, manager, npm, pip, cargo, gem

Integrate with various package managers for dependency management.

Example

```
  " Node.js/npm integration
  :!npm install              " install dependencies
  :!npm run test             " run tests
  :!npm run build            " build project

  " Python/pip integration
  :!pip install -r requirements.txt
  :!python -m pytest        " run tests

  " Rust/cargo integration
  :!cargo build              " build project
  :!cargo test               " run tests
  :!cargo run                " run project
```

37.17 REST API testing integration

Category: Integration
Tags: rest, api, http, curl, testing, client

Test REST APIs directly from Neovim using HTTP client functionality.

Example

```
  " Using rest.nvim or similar plugins
  POST https://api.example.com/users
  Content-Type: application/json

  {
    "name": "John Doe",
    "email": "john@example.com"
  }

  " Curl integration
  :!curl -X POST -H "Content-Type: application/json" \
    -d '{"name":"test"}' https://api.example.com/users

  " Save API responses
```

```
:r !curl -s https://api.github.com/users/octocat
```

37.18 SSH and remote development integration

Category: Integration
Tags: ssh, remote, development, server, connection

Integrate SSH operations for remote development workflows.

Example

```
" SSH operations
:!ssh user@server 'ls -la /project'
:!scp % user@server:/path/to/destination/

" Remote editing with netrw
:e scp://user@server//path/to/file
:browse scp://user@server//home/user/

" SSH tunnel management
:!ssh -L 8080:localhost:80 user@server -N -f  " create tunnel
:!kill $(ps aux | grep 'ssh.*8080' | awk '{print $2}')  " close tunnel
```

37.19 System clipboard integration

Category: Integration
Tags: clipboard, system, copy, paste, register

Seamlessly integrate with system clipboard for cross-application workflows.

Example

```
-- Configure clipboard integration
vim.opt.clipboard = 'unnamedplus'  -- use system clipboard

-- Manual clipboard operations
vim.keymap.set({'n', 'v'}, '<leader>y', '"+y')  -- copy to system clipboard
vim.keymap.set({'n', 'v'}, '<leader>p', '"+p')  -- paste from system
 ↪  clipboard

-- Check clipboard availability
print(vim.fn.has('clipboard'))
```

37.20 Terminal multiplexer integration

Category: Integration
Tags: tmux, screen, multiplexer, pane, session

Integrate with terminal multiplexers for enhanced workflow management.

Example

```
" Tmux integration
:!tmux new-session -d -s work    " create tmux session
:!tmux send-keys -t work 'cd project' C-m  " send commands to tmux

" Screen integration
:!screen -S build -d -m make    " run make in screen session
:!screen -r build              " reattach to screen session

" Neovim terminal with tmux-like behavior
vim.keymap.set('t', '<C-\\><C-n>', '<C-\\><C-n>')  -- escape terminal mode
```

37.21 Testing framework integration

Category: Integration
Tags: testing, framework, jest, pytest, rspec, junit

Integrate with various testing frameworks for efficient testing workflow.

Example

```
" Jest integration (JavaScript)
:!npm test                " run all tests
:!npx jest %              " test current file
:!npx jest --watch        " watch mode

" pytest integration (Python)
:!python -m pytest %     " test current file
:!python -m pytest -v    " verbose test output
:!python -m pytest --cov " coverage report

" Go test integration
:!go test                 " run tests in current package
:!go test -v ./...        " verbose tests for all packages
```

37.22 Version control system integration

Category: Integration
Tags: vcs, svn, mercurial, bazaar, perforce

Integrate with various version control systems beyond Git.

Example

```
" Subversion integration
:!svn status              " check status
```

```
:!svn diff %              " diff current file
:!svn commit -m "message" " commit changes

" Mercurial integration
:!hg status               " check status
:!hg diff %               " diff current file
:!hg commit -m "message"  " commit changes

" Perforce integration
:!p4 edit %               " check out for edit
:!p4 diff %               " show differences
:!p4 submit               " submit changelist
```

CHAPTER 38

Lsp

38.1 LSP code actions

Category: LSP
Tags: lsp, actions, refactor, fix

Use `:lua vim.lsp.buf.code_action()` to show available code actions.

Example

```
:lua vim.lsp.buf.code_action()
```

38.2 LSP format document

Category: LSP
Tags: lsp, format, style, beautify

Use `:lua vim.lsp.buf.format()` to format current buffer using LSP.

Example

```
:lua vim.lsp.buf.format()
```

38.3 LSP implementation

Category: LSP
Tags: lsp, implementation, goto

Use `gi` to jump to implementation of symbol under cursor.

Example

```
gi  " jump to implementation
```

38.4 LSP incoming calls

Category: LSP
Tags: lsp, calls, incoming, hierarchy

Use `:lua vim.lsp.buf.incoming_calls()` to show incoming call hierarchy.

Example

```
:lua vim.lsp.buf.incoming_calls()
```

38.5 LSP list workspace folders

Category: LSP
Tags: lsp, workspace, folder, list

Use `:lua print(vim.inspect(vim.lsp.buf.list_workspace_folders()))` to list workspace folders.

Example

```
:lua print(vim.inspect(vim.lsp.buf.list_workspace_folders()))
```

38.6 LSP remove workspace folder

Category: LSP
Tags: lsp, workspace, folder, remove

Use `:lua vim.lsp.buf.remove_workspace_folder()` to remove folder from workspace.

Example

```
:lua vim.lsp.buf.remove_workspace_folder()
```

38.7 LSP rename

Category: LSP
Tags: lsp, rename, refactor

Use `:lua vim.lsp.buf.rename()` to rename symbol under cursor across the project.

Example

```
:lua vim.lsp.buf.rename()
```

38.8 LSP show signature help

Category: LSP
Tags: lsp, signature, parameters, help

Use `:lua vim.lsp.buf.signature_help()` to show function signature help.

Example

```
:lua vim.lsp.buf.signature_help()
```

38.9 Toggle LSP inlay hints

Category: LSP
Tags: lsp, inlay-hints, toggle, display, parameters

Toggle inlay hints to show/hide parameter names, type annotations, and other inline information provided by LSP.

Example

```
-- Toggle inlay hints on/off:
vim.keymap.set('n', '<leader>th', function()
  vim.lsp.inlay_hint.enable(not vim.lsp.inlay_hint.is_enabled())
end, { desc = 'Toggle inlay hints' })

-- Enable inlay hints by default for buffer with LSP:
vim.api.nvim_create_autocmd('LspAttach', {
  callback = function(args)
    local client = vim.lsp.get_client_by_id(args.data.client_id)
    if client.server_capabilities.inlayHintProvider then
      vim.lsp.inlay_hint.enable(true, { bufnr = args.buf })
    end
  end,
})
```

Example

```
" Check if inlay hints are available:
:lua print(vim.lsp.inlay_hint.is_enabled())
```

CHAPTER 39

Lua

39.1 Access Neovim API functions

Category: Lua
Tags: lua, api, vim.api

Use `vim.api` to access the full Neovim C API with functions prefixed by `nvim_`.

Example

```
-- Buffer operations
local buf = vim.api.nvim_create_buf(false, true)
vim.api.nvim_buf_set_lines(buf, 0, -1, false, {"Line 1", "Line 2"})

-- Window operations
local win = vim.api.nvim_get_current_win()
vim.api.nvim_win_set_height(win, 20)

-- Get/set options
local value = vim.api.nvim_get_option_value('number', {})
vim.api.nvim_set_option_value('number', true, {})
```

39.2 Access environment variables in Lua

Category: Lua
Tags: lua, environment, vim.env

Use `vim.env` to access and set environment variables.

Example

```
-- Read environment variable
local home = vim.env.HOME
local editor = vim.env.EDITOR

-- Set environment variable
vim.env.MY_VAR = "value"
```

39.3 Call Lua functions from Vimscript

Category: Lua
Tags: lua, vimscript, interop

Use `luaeval()` or `v:lua` to call Lua functions from Vimscript.

Example

```
" Using luaeval()
call luaeval('require("mymodule").my_function(_A)', [arg1, arg2])

" Using v:lua in mappings
nnoremap <leader>f <Cmd>lua require('mymodule').my_function()<CR>

" Using v:lua in expressions
let result = v:lua.vim.fn.has('nvim-0.9')
```

39.4 Call Vimscript functions from Lua

Category: Lua
Tags: lua, vimscript, vim.fn

Use `vim.fn` to call any Vimscript or user-defined function with automatic type conversion.

Example

```
-- Call built-in function
local line_count = vim.fn.line('$')

-- Call user function
local result = vim.fn.MyCustomFunction('arg1', 'arg2')
```

39.5 Check if list is empty

Category: Lua
Tags: lua, tables, utilities

Use `vim.tbl_isempty()` to check if a table is empty.

Example

```
local my_table = {}

if vim.tbl_isempty(my_table) then
  print("Table is empty")
end
```

39.6 Check if table contains key

Category: Lua
Tags: lua, tables, utilities

Use `vim.tbl_contains()` to check if a table contains a value.

Example

```
local modes = {'n', 'v', 'i'}

if vim.tbl_contains(modes, 'n') then
  print("Normal mode included")
end
```

39.7 Choose init.vim or init.lua

Category: Lua
Tags: lua, configuration, init

Use either `init.vim` or `init.lua` for configuration, but not both simultaneously. Neovim will prioritize one over the other.

Example

```
-- init.lua example
vim.opt.number = true
vim.opt.relativenumber = true
```

39.8 Create abstract base classes with error checking

Category: Lua
Tags: lua, oop, abstract-class, validation

Implement abstract base class pattern that requires derived classes to implement specific methods.

Example

```
---@class Shape (abstract)
local Shape = {}

function Shape:new(type)
  local instance = { type = type }
  self.__index = function(t, k)
    -- Check if method must be implemented by subclass
    if k == "area" or k == "perimeter" then
      error(string.format(
```

```
        "Abstract method '%s' must be implemented by subclass '%s'",
        k, t.type or "unknown"
      ))
    end
    return self[k]
  end
  return setmetatable(instance, self)
end

---@class Rectangle : Shape
local Rectangle = Shape:new("Rectangle")

function Rectangle:new(width, height)
  local instance = Shape.new(self, "Rectangle")
  instance.width = width
  instance.height = height
  return instance
end

-- Implement required abstract methods
function Rectangle:area()
  return self.width * self.height
end

function Rectangle:perimeter()
  return 2 * (self.width + self.height)
end

---@class Circle : Shape
local Circle = Shape:new("Circle")

function Circle:new(radius)
  local instance = Shape.new(self, "Circle")
  instance.radius = radius
  return instance
end

function Circle:area()
  return math.pi * self.radius * self.radius
end

function Circle:perimeter()
  return 2 * math.pi * self.radius
end

-- Usage
local rect = Rectangle:new(5, 3)
print("Rectangle area:", rect:area())      -- 15
print("Rectangle perimeter:", rect:perimeter())  -- 16

local circle = Circle:new(4)
print("Circle area:", circle:area())       -- ~50.27
print("Circle perimeter:", circle:perimeter())   -- ~25.13

-- This would error: local shape = Shape:new()
-- shape:area() -- Error: Abstract method 'area' must be implemented
```

39.9 Create autocommand groups

Category: Lua
Tags: lua, autocommands, groups, api

Use `vim.api.nvim_create_augroup()` to organize autocommands into groups.

Example

```
-- Create group
local mygroup = vim.api.nvim_create_augroup("MyGroup", { clear = true })

-- Add autocommands to group
vim.api.nvim_create_autocmd("BufWritePre", {
  group = mygroup,
  pattern = "*.lua",
  callback = function()
    vim.lsp.buf.format()
  end,
})
```

39.10 Create autocommands with nvim_create_autocmd

Category: Lua
Tags: lua, autocommands, api

Use `vim.api.nvim_create_autocmd()` to create autocommands with Lua callbacks.

Example

```
-- Simple autocommand
vim.api.nvim_create_autocmd("BufWritePre", {
  pattern = "*.lua",
  callback = function()
    vim.lsp.buf.format()
  end,
})

-- With multiple events
vim.api.nvim_create_autocmd({"BufEnter", "BufWinEnter"}, {
  pattern = {"*.c", "*.h"},
  callback = function()
    vim.bo.cindent = true
  end,
})
```

39.11 Create buffer-local keymaps

Category: Lua
Tags: lua, keymap, buffer-local

Use the `buffer` option to create buffer-local keymaps that only work in specific buffers.

Example

```
-- Buffer-local keymap for current buffer
vim.keymap.set('n', '<leader>r', ':!python %<CR>', {
  buffer = 0,
  desc = "Run Python file"
})

-- Buffer-local keymap for specific buffer
vim.keymap.set('n', 'K', vim.lsp.buf.hover, {
  buffer = bufnr,
  desc = "LSP Hover"
})
```

39.12 Create classes with metatables

Category: Lua
Tags: lua, oop, classes, metatables

Use metatables to create class-like structures in Lua for object-oriented programming. This pattern uses `__index` metamethod for method lookup and inheritance.

Example

```
-- Define a base class
---@class Animal
local Animal = {}

-- Constructor method
function Animal:new(name, type)
  local instance = {
    name = name,
    type = type or "unknown"
  }
  -- Set metatable to enable inheritance
  self.__index = self
  return setmetatable(instance, self)
end

-- Instance method
function Animal:speak()
  print(self.name .. " makes a sound")
end

function Animal:info()
  print("I am a " .. self.type .. " named " .. self.name)
end

-- Create instance
local generic = Animal:new("Rex", "animal")
generic:speak()  -- "Rex makes a sound"
```

```
generic:info()   -- "I am a animal named Rex"
```

39.13 Create keymaps with vim.keymap.set

Category: Lua
Tags: lua, keymap, mapping, vim.keymap

Use `vim.keymap.set()` to create keymaps for multiple modes with options.

Example

```
-- Basic mapping
vim.keymap.set('n', '<leader>w', ':write<CR>')

-- Mapping with Lua function
vim.keymap.set('n', '<leader>h', function()
  print("Hello!")
end, { desc = "Say hello" })

-- Multiple modes
vim.keymap.set({'n', 'v'}, '<leader>y', '"+y', { desc = "Yank to clipboard"
↪  })

-- Buffer-local mapping
vim.keymap.set('n', 'K', vim.lsp.buf.hover, { buffer = 0, desc = "Hover
↪  documentation" })
```

39.14 Create private state with closures

Category: Lua
Tags: lua, oop, encapsulation, closures

Use closures to create private state that cannot be accessed from outside the object, providing true encapsulation.

Example

```
-- Factory function that creates objects with private state
local function BankAccount(initial_balance)
  -- Private variables (not accessible outside)
  local balance = initial_balance or 0
  local transaction_history = {}

  -- Public interface
  local self = {}

  function self.deposit(amount)
    if amount > 0 then
      balance = balance + amount
```

```
      table.insert(transaction_history, {type = "deposit", amount = amount})
      return true
    end
    return false
  end

  function self.withdraw(amount)
    if amount > 0 and amount ⩽ balance then
      balance = balance - amount
      table.insert(transaction_history, {type = "withdraw", amount =
      ↪  amount})
      return true
    end
    return false
  end

  function self.get_balance()
    return balance
  end

  function self.get_history()
    -- Return a copy to prevent external modification
    local copy = {}
    for i, v in ipairs(transaction_history) do
      copy[i] = vim.deepcopy(v)
    end
    return copy
  end

  return self
end

-- Usage
local account = BankAccount(100)
account.deposit(50)
account.withdraw(30)
print(account.get_balance())  -- 120
-- Cannot access 'balance' directly - it's private!
```

39.15 Create singleton pattern with metatables

Category: Lua
Tags: lua, oop, singleton, design-patterns

Implement the singleton pattern to ensure only one instance of a class exists, useful for managing global state.

Example

```
-- Singleton implementation
local Singleton = {}
local instance = nil
```

```
function Singleton:new()
  -- Return existing instance if it exists
  if instance then
    return instance
  end

  -- Create new instance
  local obj = {
    data = {},
    created_at = os.time(),
  }

  self.__index = self
  instance = setmetatable(obj, self)
  return instance
end

function Singleton:set(key, value)
  self.data[key] = value
end

function Singleton:get(key)
  return self.data[key]
end

-- Usage
local s1 = Singleton:new()
s1:set("config", "value1")

local s2 = Singleton:new()
print(s2:get("config"))  -- "value1"

print(s1 == s2)  -- true (same instance)
```

39.16 Create user commands in Lua

Category: Lua
Tags: lua, commands, user-commands, api

Use `vim.api.nvim_create_user_command()` to create custom commands with Lua callbacks.

Example

```
-- Simple command
vim.api.nvim_create_user_command('Hello', function()
  print('Hello!')
end, {})

-- Command with arguments
vim.api.nvim_create_user_command('Greet', function(opts)
```

```
  print('Hello, ' .. opts.args)
end, { nargs = 1 })

-- Command with completion
vim.api.nvim_create_user_command('EditConfig', function(opts)
  vim.cmd.edit(opts.args)
end, {
  nargs = 1,
  complete = 'file',
})
```

39.17 Debug Lua values

Category: Lua
Tags: lua, debug, inspect

Use `vim.inspect()` to debug and pretty-print Lua values.

Example

```
:lua print(vim.inspect(vim.fn.getbufinfo()))
```

39.18 Deep merge tables with vim.tbl_deep_extend

Category: Lua
Tags: lua, tables, utilities

Use `vim.tbl_deep_extend()` to deeply merge tables, useful for configuration.

Example

```
local defaults = {
  ui = { border = "rounded" },
  lsp = { enabled = true },
}

local user_config = {
  ui = { width = 80 },
  lsp = { enabled = false },
}

local config = vim.tbl_deep_extend('force', defaults, user_config)
-- Result: { ui = { border = "rounded", width = 80 }, lsp = { enabled =
↪  false } }
```

39.19 Defer execution with vim.schedule

Category: Lua
Tags: lua, schedule, defer, async, callback

Use `vim.schedule()` to defer execution of code to be run on the main event loop, useful for async callbacks and avoiding "textlock" errors.

Example

```
-- Defer UI updates from async context:
vim.schedule(function()
  vim.notify("Operation completed", vim.log.levels.INFO)
end)

-- Safe buffer modification in callback:
local function async_operation()
  -- Simulate async work
  vim.schedule(function()
    local buf = vim.api.nvim_get_current_buf()
    vim.api.nvim_buf_set_lines(buf, 0, 1, false, {"New content"})
  end)
end

-- Schedule multiple operations:
for i = 1, 5 do
  vim.schedule(function()
    print("Deferred operation " .. i)
  end)
end

-- Useful in LSP callbacks:
vim.lsp.buf.format({
  async = true,
  callback = function()
    vim.schedule(function()
      vim.cmd('write')
    end)
  end
})
```

39.20 Defer function execution

Category: Lua
Tags: lua, defer, utilities

Use `vim.defer_fn()` to execute a function after a delay.

Example

```
-- Execute after 1000ms (1 second)
vim.defer_fn(function()
  print("Delayed message")
end, 1000)

-- Useful for delayed UI updates
vim.defer_fn(function()
  vim.notify("Setup complete!", vim.log.levels.INFO)
end, 500)
```

39.21 Execute Lua from command line

Category: Lua
Tags: lua, command-line, inline

Use `:lua` with `=` to evaluate and print expressions quickly.

Example

```
:lua =vim.version()
:lua =vim.fn.expand('%:p')
:lua =package.path
```

39.22 Execute Vim commands from Lua

Category: Lua
Tags: lua, vim-commands, vim.cmd

Use `vim.cmd()` to execute any Vim command from Lua code.

Example

```
-- Single command
vim.cmd("colorscheme habamax")

-- Multiple commands
vim.cmd([[
  set number
  set relativenumber
  highlight Normal guibg=NONE
]])
```

39.23 Execute normal mode commands from Lua

Category: Lua
Tags: lua, normal-mode, commands

Use `vim.cmd.normal()` or `vim.api.nvim_feedkeys()` to execute normal mode commands.

Example

```
-- Using vim.cmd
vim.cmd.normal('gg')  -- Go to first line
vim.cmd.normal('viw')  -- Visual select inner word

-- Using feedkeys for more control
vim.api.nvim_feedkeys(
  vim.api.nvim_replace_termcodes('<C-w>v', true, false, true),
  'n',
  false
)
```

39.24 Filter and map tables

Category: Lua
Tags: lua, tables, functional

Use `vim.tbl_filter()` and `vim.tbl_map()` for functional table operations.

Example

```
-- Filter table
local numbers = {1, 2, 3, 4, 5, 6}
local evens = vim.tbl_filter(function(v)
  return v % 2 == 0
end, numbers)
-- Result: {2, 4, 6}

-- Map table
local doubled = vim.tbl_map(function(v)
  return v * 2
end, numbers)
-- Result: {2, 4, 6, 8, 10, 12}
```

39.25 Format buffer with LSP

Category: Lua
Tags: lua, lsp, formatting

Use `vim.lsp.buf.format()` to format the current buffer using attached LSP servers.

Example

```
-- Format current buffer
vim.lsp.buf.format()

-- Format with options
```

```
vim.lsp.buf.format({
  async = true,
  timeout_ms = 2000,
  filter = function(client)
    return client.name ~= "tsserver"
  end,
})
```

39.26 Get LSP clients for buffer

Category: Lua
Tags: lua, lsp, buffers

Use `vim.lsp.get_clients()` to get LSP clients attached to a buffer.

Example

```
-- Get clients for current buffer
local clients = vim.lsp.get_clients({ bufnr = 0 })

for _, client in ipairs(clients) do
  print("Client: " .. client.name)
end
```

39.27 Get current mode in Lua

Category: Lua
Tags: lua, mode, api

Use `vim.api.nvim_get_mode()` to get the current editor mode.

Example

```
local mode = vim.api.nvim_get_mode()
print("Current mode: " .. mode.mode)

-- Check specific mode
if mode.mode == 'n' then
  print("In normal mode")
end
```

39.28 Implement __tostring metamethod

Category: Lua
Tags: lua, oop, metatables, metamethods

Use the `__tostring` metamethod to customize how objects are converted to strings, use-

ful for debugging and logging.

Example

```
---@class Person
local Person = {}

function Person:new(name, age)
  local instance = {
    name = name,
    age = age,
  }
  self.__index = self

  -- Define how this object should be converted to string
  self.__tostring = function(obj)
    return string.format("Person(name='%s', age=%d)", obj.name, obj.age)
  end

  return setmetatable(instance, self)
end

-- Usage
local person = Person:new("Alice", 30)

-- Automatically calls __tostring when converted to string
print(person)              -- "Person(name='Alice', age=30)"
print("Hello " .. tostring(person))  -- "Hello Person(name='Alice', age=30)"

-- Useful in logging
vim.notify("Created: " .. tostring(person))
```

39.29 Implement class inheritance with metatables

Category: Lua
Tags: lua, oop, inheritance, metatables

Create derived classes by calling the parent's constructor and extending functionality. The `__index` metamethod enables method lookup in the parent class.

Example

```
-- Base class
---@class Animal
local Animal = {}

function Animal:new(name, type)
  local instance = { name = name, type = type }
  self.__index = self
  return setmetatable(instance, self)
end

function Animal:info()
```

```
  print("I am a " .. self.type)
end

-- Derived class (inherits from Animal)
---@class Dog : Animal
local Dog = Animal:new(nil, "dog")

-- Override parent method
function Dog:info()
  print("Woof! I am a " .. self.type .. " and my name is " .. self.name)
end

-- Add new method specific to Dog
function Dog:fetch()
  print(self.name .. " is fetching the ball!")
end

-- Create Dog instance
local buddy = Dog:new("Buddy", "dog")
buddy:info()   -- "Woof! I am a dog and my name is Buddy"
buddy:fetch()  -- "Buddy is fetching the ball!"

-- Another derived class
---@class Cat : Animal
local Cat = Animal:new(nil, "cat")

function Cat:info()
  print("Meow! I am a " .. self.type)
end

function Cat:scratch()
  print(self.name .. " scratches the furniture")
end

local whiskers = Cat:new("Whiskers", "cat")
whiskers:info()     -- "Meow! I am a cat"
whiskers:scratch()  -- "Whiskers scratches the furniture"
```

39.30 Implement operator overloading with metamethods

Category: Lua
Tags: lua, oop, metatables, metamethods, operators

Use metamethods to overload operators like +, -, *, ==, etc., making custom objects behave like built-in types.

Example

```
---@class Vector
local Vector = {}

function Vector:new(x, y)
```

```
  local instance = { x = x or 0, y = y or 0 }
  self.__index = self

  -- Overload addition operator
  self.__add = function(v1, v2)
    return Vector:new(v1.x + v2.x, v1.y + v2.y)
  end

  -- Overload subtraction operator
  self.__sub = function(v1, v2)
    return Vector:new(v1.x - v2.x, v1.y - v2.y)
  end

  -- Overload multiplication (scalar)
  self.__mul = function(v, scalar)
    if type(v) == "number" then
      v, scalar = scalar, v
    end
    return Vector:new(v.x * scalar, v.y * scalar)
  end

  -- Overload equality operator
  self.__eq = function(v1, v2)
    return v1.x == v2.x and v1.y == v2.y
  end

  -- Custom string representation
  self.__tostring = function(v)
    return string.format("Vector(%d, %d)", v.x, v.y)
  end

  return setmetatable(instance, self)
end

function Vector:length()
  return math.sqrt(self.x * self.x + self.y * self.y)
end

-- Usage
local v1 = Vector:new(3, 4)
local v2 = Vector:new(1, 2)

local v3 = v1 + v2        -- Vector(4, 6)
local v4 = v1 - v2        -- Vector(2, 2)
local v5 = v1 * 2         -- Vector(6, 8)
local v6 = 3 * v2         -- Vector(3, 6)

print(v3)                 -- "Vector(4, 6)"
print(v1 == v2)           -- false
print(v1 == Vector:new(3, 4))  -- true
print("Length:", v1:length())  -- "Length: 5.0"
```

39.31 Load Lua modules with require

Category: Lua
Tags: lua, modules, require

Use `require()` to load Lua modules from the `lua/` directory in your runtime path. Modules are automatically cached after first load.

Example

```
-- Load lua/mymodule.lua
local mymodule = require('mymodule')

-- Load lua/other_modules/anothermodule.lua
local another = require('other_modules.anothermodule')
```

39.32 Lua keymaps

Category: Lua
Tags: lua, keymap, mapping

Use `vim.keymap.set()` to create keymaps with inline Lua functions.

Example

```
:lua vim.keymap.set("n", "<leader>hi", function() print("Hello!") end)
```

39.33 Method chaining with metatables

Category: Lua
Tags: lua, oop, metatables, fluent-interface

Implement method chaining (fluent interface) by returning `self` from methods, allowing multiple method calls in sequence.

Example

```
-- Query builder with method chaining
---@class QueryBuilder
local QueryBuilder = {}

function QueryBuilder:new()
  local instance = {
    _table = nil,
    _where = {},
    _limit = nil,
    _order = nil,
  }
```

```
  self.__index = self
  return setmetatable(instance, self)
end

function QueryBuilder:from(table_name)
  self._table = table_name
  return self  -- Return self for chaining
end

function QueryBuilder:where(condition)
  table.insert(self._where, condition)
  return self
end

function QueryBuilder:order_by(field)
  self._order = field
  return self
end

function QueryBuilder:limit(n)
  self._limit = n
  return self
end

function QueryBuilder:build()
  local query = "SELECT * FROM " .. self._table

  if #self._where > 0 then
    query = query .. " WHERE " .. table.concat(self._where, " AND ")
  end

  if self._order then
    query = query .. " ORDER BY " .. self._order
  end

  if self._limit then
    query = query .. " LIMIT " .. self._limit
  end

  return query
end

-- Usage with method chaining
local query = QueryBuilder:new()
  :from("users")
  :where("age > 18")
  :where("active = true")
  :order_by("name")
  :limit(10)
  :build()

print(query)
-- "SELECT * FROM users WHERE age > 18 AND active = true ORDER BY name LIMIT
 ↪  10"
```

39.34 Module structure best practices

Category: Lua
Tags: lua, modules, best-practices

Structure Lua modules using the standard pattern with local tables and explicit returns.

Example

```
-- lua/mymodule.lua
local M = {}

M.config = {
  enabled = true,
}

function M.setup(opts)
  M.config = vim.tbl_extend('force', M.config, opts or {})
end

function M.do_something()
  if M.config.enabled then
    print("Doing something!")
  end
end

return M
```

39.35 Plugin configuration with metatables

Category: Lua
Tags: lua, oop, metatables, configuration, plugin

Use metatables to create flexible configuration systems that provide default values while allowing user customization.

Example

```
-- Create a configuration system with defaults
local function make_config_mt(defaults)
  return {
    -- Provide default values when key doesn't exist
    __index = function(t, k)
      local default_value = defaults[k]
      if type(default_value) ~= 'table' then
        return default_value
      end
      -- For nested tables, create a new table with metatable
      rawset(t, k, {})
      setmetatable(t[k], make_config_mt(default_value))
      return t[k]
```

```
    end,

    -- Handle setting new values
    __newindex = function(t, k, v)
      rawset(t, k, v)
      -- If value is a table, give it the same metatable behavior
      if type(v) == 'table' and defaults[k] then
        setmetatable(v, make_config_mt(defaults[k]))
      end
    end
  }
end

-- Default configuration
local defaults = {
  ui = {
    border = "rounded",
    width = 80,
    height = 20,
  },
  lsp = {
    enabled = true,
    timeout = 1000,
  },
  debug = false,
}

-- Create user config with defaults
local config = {}
setmetatable(config, make_config_mt(defaults))

-- Access default values without setting them
print(config.ui.border)  -- "rounded"
print(config.debug)      -- false

-- Override specific values
config.ui.width = 100
config.debug = true

-- Nested tables work automatically
config.lsp.timeout = 2000

print(vim.inspect(config))
```

39.36 Reload Lua modules during development

Category: Lua
Tags: lua, modules, development, reload

Clear `package.loaded` to force reload of a module during development.

Example

```
-- Reload a module
package.loaded['mymodule'] = nil
local mymodule = require('mymodule')

-- Helper function to reload
local function reload_module(name)
  package.loaded[name] = nil
  return require(name)
end

local mymod = reload_module('mymodule')
```

39.37 Run current Lua file

Category: Lua
Tags: lua, file, execute

Use `:luafile %` to execute the current Lua file inside Neovim.

Example

```
:luafile %  " run current Lua file
```

39.38 Run inline Lua code

Category: Lua
Tags: lua, inline, execute

Use `:lua` to run Lua code directly in Neovim.

Example

```
:lua print("Hello from Lua!")
```

39.39 Safely load modules with pcall

Category: Lua
Tags: lua, modules, error-handling

Use `pcall()` to safely load modules and handle potential errors without crashing.

Example

```
local ok, module = pcall(require, 'optional_module')
if ok then
  module.setup()
else
  print("Module not found: " .. module)
end
```

39.40 Schedule callbacks with vim.schedule

Category: Lua
Tags: lua, async, callbacks, vim.schedule

Use `vim.schedule()` to schedule callbacks in the main event loop safely.

Example

```
-- Schedule from async context
vim.loop.fs_stat('file.txt', function(err, stat)
  vim.schedule(function()
    -- Safe to call Neovim API here
    print(vim.inspect(stat))
  end)
end)
```

39.41 Set buffer-local options

Category: Lua
Tags: lua, options, buffer, vim.bo

Use `vim.bo` to set buffer-local options for the current or specific buffer.

Example

```
-- Current buffer
vim.bo.filetype = 'lua'
vim.bo.expandtab = true

-- Specific buffer
vim.bo[5].tabstop = 2
```

39.42 Set buffer-local variables

Category: Lua
Tags: lua, variables, buffer, vim.b

Use `vim.b` to set buffer-local variables for the current or specific buffer.

Example

```
-- Current buffer
vim.b.current_syntax = 'lua'

-- Specific buffer
vim.b[5].custom_data = { foo = "bar" }
```

39.43 Set cursor position in Lua

Category: Lua
Tags: lua, cursor, navigation

Use `vim.api.nvim_win_set_cursor()` to set cursor position programmatically.

Example

```
-- Set cursor to line 10, column 5 (1-indexed line, 0-indexed column)
vim.api.nvim_win_set_cursor(0, {10, 5})

-- Get current cursor position
local cursor = vim.api.nvim_win_get_cursor(0)
local row, col = cursor[1], cursor[2]
```

39.44 Set global variables in Lua

Category: Lua
Tags: lua, variables, vim.g

Use `vim.g` to set global variables with native Lua types.

Example

```
-- Set global variable
vim.g.mapleader = ' '
vim.g.my_config = { key = "value", enabled = true }

-- Read global variable
local leader = vim.g.mapleader
```

39.45 Set options with vim.o

Category: Lua
Tags: lua, options, vim.o

Use `vim.o` for direct variable-like access to global options.

Example

```
-- Set global option
vim.o.ignorecase = true
vim.o.smartcase = true

-- More concise than vim.opt for simple assignments
vim.o.updatetime = 250
```

39.46 Set options with vim.opt

Category: Lua
Tags: lua, options, vim.opt

Use `vim.opt` to set options with a convenient Lua interface that supports advanced operations.

Example

```
-- Set single option
vim.opt.number = true
vim.opt.tabstop = 4

-- Append to list option
vim.opt.wildignore:append('*.pyc')

-- Remove from list option
vim.opt.wildignore:remove('*.o')
```

39.47 Set tabpage-local variables

Category: Lua
Tags: lua, variables, tabpage, vim.t

Use `vim.t` to set tabpage-local variables.

Example

```
-- Current tabpage
vim.t.tab_name = "Main"

-- Specific tabpage
vim.t[2].custom_data = {}
```

39.48 Set window-local options

Category: Lua
Tags: lua, options, window, vim.wo

Use `vim.wo` to set window-local options.

Example

```
-- Current window
vim.wo.number = true
vim.wo.wrap = false

-- Specific window
vim.wo[1001].cursorline = true
```

39.49 Set window-local variables

Category: Lua
Tags: lua, variables, window, vim.w

Use `vim.w` to set window-local variables.

Example

```
-- Current window
vim.w.quickfix_title = "My Results"

-- Specific window
vim.w[1001].custom_setting = true
```

39.50 Split and join strings

Category: Lua
Tags: lua, strings, utilities

Use `vim.split()` to split strings into tables.

Example

```
-- Split by delimiter
local parts = vim.split("foo,bar,baz", ",")
-- Result: {"foo", "bar", "baz"}

-- Split by pattern
local lines = vim.split(text, "\n", { plain = true })
```

39.51 Trim whitespace from strings

Category: Lua
Tags: lua, strings, utilities

Use `vim.trim()` to remove leading and trailing whitespace.

Example

```
local cleaned = vim.trim("  hello world  ")
-- Result: "hello world"
```

39.52 Use Lua heredoc in Vimscript

Category: Lua
Tags: lua, vimscript, heredoc

Use Lua heredoc syntax to write Lua code blocks in `init.vim`.

Example

```
lua << EOF
  local function hello()
    print("Hello from Lua in Vimscript!")
  end
  hello()
EOF
```

39.53 Use vim.loop for async operations

Category: Lua
Tags: lua, async, libuv, vim.loop

Use `vim.loop` (luv) to access libuv for asynchronous operations.

Example

```
-- File system operations
vim.loop.fs_stat('myfile.txt', function(err, stat)
  if stat then
    print('File size: ' .. stat.size)
  end
end)

-- Timers
local timer = vim.loop.new_timer()
timer:start(1000, 0, function()
  print('Timer fired!')
  timer:close()
end)
```

39.54 Use vim.notify for notifications

Category: Lua
Tags: lua, notifications, ui

Use `vim.notify()` to show notifications with different log levels.

Example

```
-- Basic notification
vim.notify("Operation completed")

-- With log level
vim.notify("Warning message", vim.log.levels.WARN)
vim.notify("Error occurred", vim.log.levels.ERROR)
vim.notify("Info message", vim.log.levels.INFO)
```

39.55 Use vim.ui.input for user input

Category: Lua
Tags: lua, ui, input

Use `vim.ui.input()` to prompt user for text input.

Example

```
vim.ui.input({
  prompt = 'Enter your name: ',
  default = 'Anonymous',
}, function(input)
  if input then
    print('Hello, ' .. input)
  end
end)
```

39.56 Use vim.ui.select for user selection

Category: Lua
Tags: lua, ui, selection

Use `vim.ui.select()` to prompt user to select from a list of options.

Example

```
vim.ui.select(
  {'Option 1', 'Option 2', 'Option 3'},
  {
    prompt = 'Choose an option:',
```

```
    format_item = function(item)
      return "→ " .. item
    end,
  },
  function(choice)
    if choice then
      print('You selected: ' .. choice)
    end
  end
)
```

39.57 Validate function arguments

Category: Lua
Tags: lua, validation, utilities

Use `vim.validate()` to validate function arguments with clear error messages.

Example

```
function my_function(opts)
  vim.validate({
    name = {opts.name, 'string'},
    age = {opts.age, 'number', true},  -- true means optional
    callback = {opts.callback, 'function'},
  })

  -- Function implementation
end
```

39.58 View loaded Lua modules

Category: Lua
Tags: modules, loaded, debug

Use `package.loaded` to view all loaded Lua modules.

Example

```
:lua print(vim.inspect(package.loaded))
```

39.59 Work with buffer text in Lua

Category: Lua
Tags: lua, buffer, text, api

Use buffer API functions to read and modify buffer content.

Example

```
-- Get lines from buffer
local lines = vim.api.nvim_buf_get_lines(0, 0, -1, false)

-- Set lines in buffer
vim.api.nvim_buf_set_lines(0, 0, -1, false, {"New line 1", "New line 2"})

-- Get specific line
local current_line = vim.api.nvim_get_current_line()

-- Set specific line
vim.api.nvim_set_current_line("Modified line")
```

39.60 Work with quickfix list in Lua

Category: Lua
Tags: lua, quickfix, diagnostics

Use `vim.fn.setqflist()` and `vim.fn.getqflist()` to work with quickfix list.

Example

```
-- Set quickfix list
vim.fn.setqflist({
  {filename = 'file1.lua', lnum = 10, text = 'Error message'},
  {filename = 'file2.lua', lnum = 20, text = 'Warning'},
})

-- Open quickfix window
vim.cmd.copen()

-- Get quickfix items
local qf_items = vim.fn.getqflist()
```

CHAPTER 40

Macros

40.1 Edit macro in command line

Category: Macros
Tags: macro, edit, modify, command

Use `:let @a='` then `Ctrl+R Ctrl+R a` to paste macro contents for editing, then close with `'`.

Example

```
:let @a='<Ctrl+R><Ctrl+R>a'  " edit macro 'a' inline
```

40.2 Execute macro

Category: Macros
Tags: macro, execute, replay

Use `@{letter}` to execute macro stored in register {letter}, or `@@` to repeat the last executed macro.

Example

```
@a  " execute macro 'a'
@@  " repeat last macro
```

40.3 Macro for data transformation

Category: Macros
Tags: macro, transform, data, format

Use macros to transform structured data formats efficiently.

Example

```
" Transform tab-separated to Python dict format
qa              " start recording macro 'a'
```

```
I"<Esc>         " insert quote at beginning
f<Tab>          " find tab character
r:              " replace with colon
a": "<Esc>      " append quote-colon-quote
A",<Esc>        " append comma at end
j0              " move to next line, beginning
q               " stop recording

" Apply to multiple lines
10@a            " run macro 'a' 10 times
```

40.4 Make existing macro recursive

Category: Macros
Tags: macro, recursive, modify, qQ

Convert an existing macro to recursive by appending the macro call to itself using `qQ@qq`.

Example

```
" After recording macro @q normally:
qQ@qq   " q=start recording to Q, Q=append to q, @q=call q, q=stop
" Now @q is recursive and will loop until end of file
```

40.5 Quick macro shortcuts

Category: Macros
Tags: macro, shortcut, mapping, space

Set up convenient mappings for macro execution and recording.

Example

```
" Map space to execute last macro
nnoremap <Space> @@

" Map specific keys for common macro registers
nnoremap <leader>1 @q
nnoremap <leader>2 @w
nnoremap <leader>3 @e

" Map for visual selection macro execution
vnoremap <leader>m :normal @q<CR>
```

40.6 Record recursive macro by including the self-reference

Category: Macros
Tags: macro, recursive, loop, automation

Create recursive macros by including `@q` (self-reference) within the macro recording to process entire file automatically.

Example

```
qqq          " clear register q
qq           " start recording macro q
" ... your editing commands ...
@q           " recursive call to self
q            " stop recording
@q           " execute recursive macro
```

40.7 Record recursive macro that calls itself until a condition is met

Category: Macros
Tags: macro, recursive, loop, repeat

Create a recursive macro that calls itself for repeated operations until a condition is met.

Example

```
" Record recursive macro in register 'a'
qa              " start recording
/pattern<CR>    " search for pattern (will fail when no more matches)
n               " go to next match
@a              " call itself recursively
q               " stop recording

" Execute to process all matches
@a
```

40.8 Run macro on multiple files

Category: Macros
Tags: macro, files, multiple, batch

Use `:argdo normal @q` to run macro q on all files in argument list, or `:bufdo normal @q` for all buffers.

Example

```
:args *.txt         " load all txt files
:argdo normal @q    " run macro q on all files
:argdo update       " save all changed files
```

40.9 Run macro over visual selection

Category: Macros
Tags: macro, visual, selection

Use `:'<,'>normal @q` to run macro q over visual selection.

Example

```
:'<,'>normal @q  " run macro q on selection
```

40.10 Save macro in vimrc

Category: Macros
Tags: macro, save, persistent, vimrc

Use `let @a='macro_contents'` in vimrc to make macros persistent across Vim sessions.

Example

```
let @a='ddp'  " save line swap macro permanently
```

40.11 View macro contents

Category: Macros
Tags: macro, view, register, debug

Use `:reg` to view all registers including macros, or `:reg a` to view specific macro in register 'a'.

Example

```
:reg     " view all registers
:reg a   " view macro in register 'a'
```

CHAPTER 41

Marks

41.1 Jump to marks

Category: Marks
Tags: marks, jump, navigation

Use `'{letter}` to jump to beginning of line with mark, or `` `{letter} `` to jump to exact mark position.

Example

```
'a   " jump to line with mark 'a'
`a   " jump to exact position of mark 'a'
```

41.2 Set marks

Category: Marks
Tags: marks, position, bookmark

Use `m{letter}` to set a mark at current position. Use lowercase letters for file-specific marks and uppercase for global marks.

Example

```
ma  " set mark 'a'
mB  " set global mark 'B'
```

CHAPTER 42

Modern neovim api

42.1 Advanced autocommand patterns and groups

Category: Autocommands
Tags: autocmd, pattern, group, multiple, events

Use advanced patterns and groups with `vim.api.nvim_create_autocmd()` for sophisticated event handling.

Example

```
-- Create autocommand group for organization
local group = vim.api.nvim_create_augroup('MyCustomGroup', { clear = true })

-- Multiple events and patterns
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  group = group,
  pattern = {'*.md', '*.txt', '*.rst'},
  callback = function(event)
    vim.opt_local.spell = true
    vim.opt_local.wrap = true
    print('Text file opened: ' .. event.file)
  end
})

-- Conditional logic in callback
vim.api.nvim_create_autocmd('BufWritePre', {
  group = group,
  pattern = '*',
  callback = function()
    -- Only format if LSP client is attached
    if next(vim.lsp.get_active_clients({bufnr = 0})) then
      vim.lsp.buf.format({ timeout_ms = 2000 })
    end
  end
})
```

42.2 Buffer-local configurations and mappings

Category: Configuration
Tags: buffer, local, mapping, option, scope

Use `vim.opt_local` and buffer-specific keymaps to create settings that only apply to specific buffers.

Example

```
-- Buffer-local options (reset when switching buffers)
vim.opt_local.tabstop = 2
vim.opt_local.shiftwidth = 2
vim.opt_local.expandtab = true

-- Buffer-local keymaps (only work in this buffer)
vim.keymap.set('n', '<leader>r', ':!node %<CR>', {
  buffer = 0,
  desc = 'Run current JS file'
})

-- Autocommand for filetype-specific buffer settings
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.opt_local.textwidth = 79
    vim.keymap.set('n', '<F5>', ':!python %<CR>', { buffer = true })
  end
})
```

42.3 Create floating windows with Neovim API

Category: Advanced Neovim
Tags: floating, window, api, modern, popup

Use `vim.api.nvim_open_win()` to create floating windows programmatically for custom interfaces and popups.

Example

```
-- Create a floating window
local buf = vim.api.nvim_create_buf(false, true)
local win = vim.api.nvim_open_win(buf, true, {
  relative = 'cursor',
  width = 50,
  height = 10,
  row = 1,
  col = 0,
  style = 'minimal',
  border = 'rounded'
})

-- Add content
vim.api.nvim_buf_set_lines(buf, 0, -1, false, {
  'Hello from floating window!',
  'Press q to close'
})
```

```
-- Close on q
vim.keymap.set('n', 'q', '<cmd>close<cr>', { buffer = buf })
```

42.4 Custom diagnostic configuration

Category: Diagnostics
Tags: diagnostic, lsp, configuration, signs, virtual

Use `vim.diagnostic.config()` to customize how diagnostics are displayed and behave.

Example

```
-- Configure diagnostic display
vim.diagnostic.config({
  virtual_text = {
    prefix = '*',
    spacing = 2,
    severity = { min = vim.diagnostic.severity.WARN }
  },
  signs = {
    severity = { min = vim.diagnostic.severity.INFO }
  },
  underline = {
    severity = { min = vim.diagnostic.severity.ERROR }
  },
  float = {
    border = 'rounded',
    source = 'always',
    header = '',
    prefix = '',
  },
  update_in_insert = false,
  severity_sort = true,
})

-- Custom diagnostic signs
local signs = { Error = " ", Warn = " ", Hint = " ", Info = " " }
for type, icon in pairs(signs) do
  local hl = "DiagnosticSign" .. type
  vim.fn.sign_define(hl, { text = icon, texthl = hl, numhl = hl })
end
```

42.5 Custom user commands with completion

Category: Advanced Neovim
Tags: command, completion, api, custom, user

Use `vim.api.nvim_create_user_command()` to create custom commands with intelligent completion and argument handling.

Example

```
-- Command with file completion
vim.api.nvim_create_user_command('EditConfig', function(opts)
  vim.cmd('edit ' .. vim.fn.stdpath('config') .. '/' .. opts.args)
end, {
  nargs = 1,
  complete = 'file',
  desc = 'Edit config file'
})

-- Command with custom completion
vim.api.nvim_create_user_command('LogLevel', function(opts)
  vim.log.level = vim.log.levels[opts.args:upper()]
end, {
  nargs = 1,
  complete = function() return {'debug', 'info', 'warn', 'error'} end
})

-- Usage: :EditConfig init.lua  or  :LogLevel debug
```

CHAPTER 43

Movement

43.1 Alternative movement keys

Category: Movement
Tags: alternative, movement, keys

Use `Ctrl+h` (same as `h`), `Ctrl+j` (same as `j`), `Ctrl+k` (same as `k`), `Ctrl+n` (same as `j`), `Ctrl+p` (same as `k`) as alternative movement keys.

Example

```
Ctrl+h  " same as h (left)
Ctrl+j  " same as j (down)
Ctrl+k  " same as k (up)
Ctrl+n  " same as j (down)
Ctrl+p  " same as k (up)
```

43.2 Basic cursor movement

Category: Movement
Tags: cursor, navigation, movement

Use `h`, `j`, `k`, `l` to move the cursor left, down, up, and right respectively.

Example

```
h  " move left
j  " move down
k  " move up
l  " move right
```

43.3 Center cursor on screen

Category: Movement
Tags: center, screen, cursor

Use `zz` to center the current line on screen, `zt` to move it to the top, and `zb` to move it to the bottom.

Example

```
zz  " center line
zt  " line to top
zb  " line to bottom
```

43.4 Change list navigation

Category: Movement
Tags: change, list, edit

Use `g;` to go to previous change location and `g,` to go to next change location. Use `:changes` to see the change list.

Example

```
g;       " previous change
g,       " next change
:changes " show change list
```

43.5 Character search on line

Category: Movement
Tags: character, find, line

Use `f{char}` to find next occurrence of character, `F{char}` to find previous occurrence, `t{char}` to move to before next occurrence, and `T{char}` to move to after previous occurrence.

Example

```
fa  " find next 'a'
Fa  " find previous 'a'
ta  " move to before next 'a'
Ta  " move to after previous 'a'
```

43.6 Document navigation

Category: Movement
Tags: document, navigation, movement

Use `gg` to go to the first line of the document and `G` to go to the last line.

Example

```
gg  " first line
G   " last line
```

43.7 Jump list navigation

Category: Movement
Tags: jump, list, navigation

Use `Ctrl+o` to go back to previous location and `Ctrl+i` to go forward in the jump list. Use `:jumps` to see the jump list.

Example

```
Ctrl+o  " previous location
Ctrl+i  " next location
:jumps  " show jump list
```

43.8 Jump multiple lines with arrow keys

Category: Movement
Tags: jump, lines, arrows, count

Use number + arrow keys to jump multiple lines quickly. More intuitive than `j`/`k` for some users.

Example

```
5↑    " jump 5 lines up
5↓    " jump 5 lines down
10↑   " jump 10 lines up
3→    " move 3 characters right
```

43.9 Jump to definition

Category: Movement
Tags: definition, jump, lsp

Use `gd` to jump to the definition of the symbol under the cursor (requires LSP). Use `gD` to go to declaration instead of definition.

Example

```
gd  " go to definition
gD  " go to declaration
```

43.10 Jump to specific line

Category: Movement
Tags: line, jump, navigation

Use `{number}G` to jump to a specific line number, or `:{number}` as an alternative.

Example

```
42G   " jump to line 42
:42   " jump to line 42
```

43.11 Last non-blank character motion (g_)

Category: Movement
Tags: motion, line, end, g_, non-blank

Use `g_` to move to the last non-blank character of the line, unlike `$` which goes to the very end including whitespace.

Example

```
g_     " go to last non-blank character
yg_    " yank to last non-blank character (no trailing spaces)
dg_    " delete to last non-blank character
```

43.12 Line navigation

Category: Movement
Tags: line, navigation, movement

Use `0` to jump to the beginning of the line, `^` to jump to the first non-blank character, and `$` to jump to the end of the line.

Example

```
0  " line start
^  " first non-blank
$  " line end
```

43.13 Line number movement

Category: Movement
Tags: line, number, goto, absolute

Use `{number}gg` or `{number}G` to go to absolute line number, where `{number}` is the line

you want to jump to.

Example

```
42gg  " go to line 42
100G  " go to line 100
1G    " go to first line (same as gg)
```

43.14 Matching brackets

Category: Movement
Tags: brackets, matching, navigation

Use `%` to jump to the matching bracket, parenthesis, or brace.

Example

```
%  " jump to matching bracket
```

43.15 Middle of screen

Category: Movement
Tags: middle, screen, position

Use `M` to move cursor to the middle line of the screen.

Example

```
M  " move to middle of screen
```

43.16 Page movement

Category: Movement
Tags: page, scroll, movement

Use `Ctrl+f` to move forward one full page, `Ctrl+b` to move backward one full page, `Ctrl+d` to move down half page, and `Ctrl+u` to move up half page.

Example

```
Ctrl+f  " forward full page
Ctrl+b  " backward full page
Ctrl+d  " down half page
Ctrl+u  " up half page
```

43.17 Page scrolling with cursor positioning

Category: Movement
Tags: scroll, page, cursor, position

Use `Ctrl+f` to scroll forward full page, `Ctrl+b` backward full page, `Ctrl+d` down half page, `Ctrl+u` up half page, `Ctrl+e` scroll up (cursor stays), `Ctrl+y` scroll down (cursor stays).

Example

```
Ctrl+f  " page forward
Ctrl+b  " page backward
Ctrl+d  " half page down
Ctrl+u  " half page up
Ctrl+e  " scroll up (cursor stays)
Ctrl+y  " scroll down (cursor stays)
```

43.18 Paragraph movement

Category: Movement
Tags: paragraph, navigation, text

Use `{` to move to the beginning of current paragraph and `}` to move to the beginning of next paragraph.

Example

```
{  " previous paragraph
}  " next paragraph
```

43.19 Repeat character search

Category: Movement
Tags: repeat, character, search

Use `;` to repeat last character search in the same direction and `,` to repeat in the opposite direction.

Example

```
;  " repeat search forward
,  " repeat search backward
```

43.20 Screen position navigation

Category: Movement
Tags: screen, position, navigation

Use `H` to move cursor to top of screen and `L` to move cursor to bottom of screen.

Example

```
H  " move to top of screen
L  " move to bottom of screen
```

43.21 Screen scrolling

Category: Movement
Tags: scroll, screen, navigation

Use `Ctrl+e` to scroll down and `Ctrl+y` to scroll up without moving the cursor position.

Example

```
Ctrl+e  " scroll down
Ctrl+y  " scroll up
```

43.22 Sentence movement

Category: Movement
Tags: sentence, navigation, text

Use `(` to move to the beginning of current sentence and `)` to move to the beginning of next sentence.

Example

```
(  " previous sentence
)  " next sentence
```

43.23 Suspend and background

Category: Movement
Tags: suspend, background, shell

Use `Ctrl+z` to suspend Neovim and return to shell (terminal). You can do whatever you want in the hsell. Use `jobs` in shell to list suspended jobs. The list contains rows formatted like the following one:

Example

```
[5]  + suspended  nvim playground
```

The number in square brackets is the number of the job that can be resumed. Just use `fg #` (in the shell) followed by job number to resume the job:

Example

```
Ctrl+z  " suspend to shell
jobs    " list suspended jobs
fg #5   " resume job #5
```

43.24 Word movement

Category: Movement
Tags: word, navigation, movement

Use `w` to jump to the start of the next word, `e` to jump to the end of the current word, and `b` to jump backwards to the start of the previous word.

Example

```
w  " next word start
e  " end of word
b  " previous word start
```

43.25 Word movement alternatives

Category: Movement
Tags: word, WORD, movement, whitespace

Use `W` to jump to start of next WORD, `E` to jump to end of current WORD, and `B` to jump to start of previous WORD (WORD means whitespace-separated).

Example

```
W  " next WORD (whitespace-separated)
E  " end of WORD
B  " previous WORD
```

43.26 Z-commands - horizontal scrolling

Category: Movement
Tags: scroll, horizontal, wrap, screen

Use `zh`/`zl` to scroll left/right by character, `zH`/`zL` for half-screen, `zs`/`ze` to position cursor at start/end.

Example

```
zh  " scroll right (when wrap is off)
zl  " scroll left (when wrap is off)
zH  " scroll right half-screenwidth
zL  " scroll left half-screenwidth
zs  " scroll cursor to start of screen
ze  " scroll cursor to end of screen
```

43.27 Z-commands - redraw with cursor positioning

Category: Movement
Tags: redraw, cursor, position, screen

Use `z<Enter>` to redraw with cursor at top (first non-blank), `z.` for center, `z-` for bottom.

Example

```
z<Enter>  " redraw, cursor line at top (first non-blank)
z.        " redraw, cursor line at center (first non-blank)
z-        " redraw, cursor line at bottom (first non-blank)
```

43.28 Z-commands - window height adjustment

Category: Movement
Tags: window, height, resize, redraw

Use `z{height}<Enter>` to set window height and redraw, `z+` for line below window, `z^` for line above.

Example

```
z20<Enter>  " make window 20 lines high
z+          " cursor to line below window
z^          " cursor to line above window
```

CHAPTER 44

Navigation

44.1 Buffer switching shortcuts

Category: Navigation
Tags: buffer, switching, shortcuts, quick

Use `:ls` to list buffers, `:b#` for previous buffer, or create mappings for quick buffer navigation.

Example

```
:ls         " list all buffers
:b#         " switch to previous buffer
Ctrl+^      " alternate between current and previous buffer
:b partial  " switch to buffer matching partial name
```

44.2 Enhanced navigation with Flash.nvim

Category: Navigation
Tags: plugin, flash, leap, jump, motion, search

Use Flash.nvim plugin to enhance native Neovim motions like `f`, `t`, `w` with multi-line, labeled jumping.

Example

```
-- Install with lazy.nvim:
{
  "folke/flash.nvim",
  event = "VeryLazy",
  opts = {
    modes = {
      char = {
        enabled = true,  -- Enable for f, F, t, T motions
        jump_labels = true
      }
    }
  },
  keys = {
```

```
    { "s", mode = { "n", "x", "o" }, function() require("flash").jump() end,
    ↪  desc = "Flash" },
    { "S", mode = { "n", "x", "o" }, function()
    ↪  require("flash").treesitter() end, desc = "Flash Treesitter" },
    { "r", mode = "o", function() require("flash").remote() end, desc =
    ↪  "Remote Flash" },
    { "R", mode = { "o", "x" }, function()
    ↪  require("flash").treesitter_search() end, desc = "Treesitter Search"
    ↪  },
  },
}

-- Usage:
-- Press 's' followed by characters to search
-- Press 'S' for treesitter-aware selection
-- Use enhanced f/t/w motions with labels
```

44.3 Fast buffer access

Category: Navigation
Tags: buffer, fast, access, number

Create mappings to quickly access first nine buffers using leader key combinations.

Example

```
nnoremap <leader>1 :1b<CR>
nnoremap <leader>2 :2b<CR>
nnoremap <leader>3 :3b<CR>
nnoremap <leader>4 :4b<CR>
nnoremap <leader>5 :5b<CR>
" Continue for buffers 6-9
```

44.4 Go to declaration

Category: Navigation
Tags: lsp, declaration, goto

Use `gD` to go to declaration of symbol under cursor.

Example

```
gD  " go to declaration
```

44.5 Go to file and open URL under cursor

Category: Navigation
Tags: file, cursor, goto, url, gf, gx

Use `gf` to open the file whose name is under the cursor, or `gx` to open URLs/links in external browser.

Example

```
gf  " go to file under cursor (path/to/file.txt)
gx  " open URL under cursor in browser (https://example.com)
```

44.6 Jump between functions

Category: Navigation
Tags: function, jump, treesitter

Use `]m` to jump to next function start and `[m` to jump to previous function start.

Example

```
]m  " next function start
[m  " previous function start
```

44.7 Jump between matching pair of parenthesis `([{...}])`

Category: Navigation
Tags: parenthesis

Position your cursor on `(`, `)`, `[`, `]`, `{`, `}`. Use `%` to jump between corresponding opening and closing symbols.

Example

```
%   "jumps between corresponding parenthesis
```

44.8 Jump to block boundaries

Category: Navigation
Tags: block, boundaries, jump

Use `[{` to jump to start of current block and `]}` to jump to end of current block.

Example

```
[{  " jump to block start
]}  " jump to block end
```

44.9 Jump to definition with split

Category: Navigation
Tags: definition, split, window, tags

Use `Ctrl+W ]` to open tag definition in new split window.

Example

```
Ctrl+W ]  " open tag in split
```

44.10 Jump to last edit location

Category: Navigation
Tags: edit, location, jump

Use `` `. `` to jump to the exact location of the last edit.

Example

```
`.  " jump to last edit location
```

44.11 Jump to matching brace

Category: Navigation
Tags: brace, bracket, matching, jump

Use `%` to jump to matching brace/bracket/parenthesis, works with (), [], {}, and more.

Example

```
%   " jump to matching brace/bracket/parenthesis
[%  " jump to previous unmatched (
]%  " jump to next unmatched )
```

44.12 Jump to random line

Category: Navigation
Tags: random, line, jump, goto

Use `:{number}G` or `:{number}` to jump to specific line, or `:echo line('$')` to see total lines.

Example

```
:42G        " jump to line 42
:42         " jump to line 42 (alternative)
G           " jump to last line
:echo line('$')  " show total number of lines
```

44.13 Jump to tag under cursor

Category: Navigation
Tags: tags, jump, definition, ctags

Use `Ctrl+]` to jump to tag under cursor, or `Ctrl+T` to jump back. Requires tags file.

Example

```
Ctrl+]  " jump to tag
Ctrl+T  " jump back
```

44.14 LSP go to references

Category: Navigation
Tags: lsp, references, goto

Use `gr` to go to references of symbol under cursor (requires LSP server).

Example

```
gr  " go to references
```

44.15 List jump locations

Category: Navigation
Tags: jump, list, history

Use `:ju` to list all jump locations in the jump list.

Example

```
:ju  " list jump locations
```

44.16 Navigate quickfix list

Category: Navigation
Tags: quickfix, navigation, errors

Use `:cnext` to go to next item in quickfix list and `:cprev` to go to previous item.

Example

```
:cnext  " next quickfix item
:cprev  " previous quickfix item
```

Title: Quickfix navigation with bracket commands # Category: Navigation # Tags: quickfix, navigation, bracket, [q,]q, [l,]l — Use `[q` and `]q` to navigate quickfix items, `[l` and `]l` for location list items.

Example

```
[q    " go to previous quickfix item
]q    " go to next quickfix item
[l    " go to previous location list item
]l    " go to next location list item
```

44.17 Navigate to alternate file

Category: Navigation
Tags: alternate, file, header, source

Use `:A` to switch to alternate file (e.g., .h to .c), or `Ctrl+^` to switch to previous buffer.

Example

```
:A      " alternate file
Ctrl+^  " previous buffer
```

44.18 Square bracket navigation - C comments

Category: Navigation
Tags: comment, C, navigation

Use `[/` and `]/` to jump to start/end of C-style comments. Use `[*` as alternative to `[/`.

Example

```
[/  " jump to previous start of C comment
]/  " jump to next end of C comment
[*  " same as [/ (alternative)
]*  " same as ]/ (alternative)
```

44.19 Square bracket navigation - changes and diffs

Category: Navigation
Tags: change, diff, navigation

Use `[c` and `]c` to jump between changes in diff mode.

Example

```
[c  " jump to previous change
]c  " jump to next change
```

44.20 Square bracket navigation - definitions and includes

Category: Navigation
Tags: definition, include, search

Use `[Ctrl+d/]Ctrl+d` to jump to #define, `[Ctrl+i/]Ctrl+i` to jump to lines containing word under cursor.

Example

```
[Ctrl+d  " jump to previous #define matching word
]Ctrl+d  " jump to next #define matching word
[Ctrl+i  " jump to previous line containing word
]Ctrl+i  " jump to next line containing word
```

44.21 Square bracket navigation - folds

Category: Navigation
Tags: fold, navigation, code

Use `[z` and `]z` to jump to start/end of open fold.

Example

```
[z  " jump to start of open fold
]z  " jump to end of open fold
```

44.22 Square bracket navigation - list definitions

Category: Navigation
Tags: list, definition, search, include

Use `[D/]D` to list all #defines, `[I/]I` to list all lines containing word under cursor.

Example

```
[D  " list all #defines matching word under cursor
]D  " list all #defines matching word under cursor
[I  " list all lines containing word under cursor
]I  " list all lines containing word under cursor
```

44.23 Square bracket navigation - marks

Category: Navigation
Tags: mark, navigation, position

Use `['` and `]'` to jump to previous/next lowercase mark (first non-blank), `[`` and `]`` to jump to exact mark position.

Example

```
['  " jump to previous mark (first non-blank)
]'  " jump to next mark (first non-blank)
[`  " jump to previous mark (exact position)
]`  " jump to next mark (exact position)
```

44.24 Square bracket navigation - member functions

Category: Navigation
Tags: function, member, class, navigation

Use `[m` and `]m` to jump between member function starts.

Example

```
[m  " jump to previous start of member function
]m  " jump to next start of member function
```

44.25 Square bracket navigation - preprocessing

Category: Navigation
Tags: preprocessing, define, include

Use `[#` and `]#` to jump between #if/#else/#endif blocks.

Example

```
[#  " jump to previous #if, #else, or #ifdef
]#  " jump to next #endif or #else
```

44.26 Square bracket navigation - sections

Category: Navigation
Tags: section, navigation, document

Use [[and]] to jump between sections, [] and][to jump between SECTIONS (different formatting).

Example

```
[[  " jump to previous section
]]  " jump to next section
[]  " jump to previous SECTION
][  " jump to next SECTION
```

44.27 Square bracket navigation - show definitions

Category: Navigation
Tags: show, definition, preview

Use [d/]d to show first #define, [i/]i to show first line containing word under cursor.

Example

```
[d  " show first #define matching word
]d  " show first #define matching word
[i  " show first line containing word
]i  " show first line containing word
```

44.28 Square bracket navigation - spelling

Category: Navigation
Tags: spelling, error, navigation

Use [s and]s to jump between misspelled words.

Example

```
[s  " jump to previous misspelled word
]s  " jump to next misspelled word
```

44.29 Square bracket navigation - unmatched brackets

Category: Navigation
Tags: bracket, unmatched, navigation

Use `[(` and `])` to jump to unmatched parentheses, `[{` and `]}` to jump to unmatched braces.

Example

```
[(  " jump to previous unmatched (
])  " jump to next unmatched )
[{  " jump to previous unmatched {
]}  " jump to next unmatched }
```

44.30 Toggle netrw file explorer

Category: Navigation
Tags: netrw, explorer, toggle, file, browser

Use `:Lexplore` to toggle the netrw file explorer in a vertical split on the left side.

Example

```
:Lexplore    " toggle left explorer
:Vexplore    " open explorer in vertical split
:Sexplore    " open explorer in horizontal split
:Explore     " open explorer in current window
```

44.31 View jump list

Category: Navigation
Tags: jump, list, view

Use `:jumps` to show the jump list with all stored positions.

Example

```
:jumps  " show jump list
```

CHAPTER 45

Neovim features

45.1 Auto commands with Lua

Category: Neovim Features
Tags: autocmd, lua, events, modern

Create auto commands using Lua API for better organization and type safety.

Example

```
vim.api.nvim_create_autocmd('BufWritePre', {
  pattern = '*.lua',
  callback = function()
    vim.lsp.buf.format()
  end,
  desc = 'Format Lua files on save'
})
```

45.2 Built-in snippet support

Category: Neovim Features
Tags: snippets, completion, modern

Neovim 0.10+ has built-in snippet support for LSP and completion engines.

Example

```
-- Expand snippet
vim.snippet.expand("for i in range(10):\n\tpass")

-- Jump to next placeholder
vim.snippet.jump(1)

-- Jump to previous placeholder
vim.snippet.jump(-1)
```

45.3 Built-in terminal

Category: Neovim Features
Tags: terminal, integrated, modern

Use `:term` to open terminal, `Ctrl+\` followed by `Ctrl+n` to exit terminal mode to normal mode.

Example

```
:term           " open terminal
Ctrl+\ Ctrl+n   " exit terminal mode
```

45.4 Diagnostic API

Category: Neovim Features
Tags: diagnostics, api, lsp, modern

Use Neovim's built-in diagnostic API for showing errors, warnings, and info messages.

Example

```
-- Set diagnostics
vim.diagnostic.set(ns_id, buf, {
  {
    lnum = 0,
    col = 0,
    message = "Error message",
    severity = vim.diagnostic.severity.ERROR
  }
})

-- Show diagnostics in floating window
vim.diagnostic.open_float()
```

45.5 Extended marks

Category: Neovim Features
Tags: marks, extmarks, api, highlighting

Use extmarks for advanced text annotations and virtual text that persists across edits.

Example

```
local ns = vim.api.nvim_create_namespace('my_namespace')
vim.api.nvim_buf_set_extmark(0, ns, 0, 0, {
  virt_text = {{'Virtual text', 'Comment'}},
  virt_text_pos = 'eol'
```

```
})
```

45.6 Floating windows API

Category: Neovim Features
Tags: floating, windows, api, modern

Create floating windows for custom UI elements using Neovim's floating window API.

Example

```
local buf = vim.api.nvim_create_buf(false, true)
local win = vim.api.nvim_open_win(buf, true, {
  relative = 'cursor',
  width = 50,
  height = 10,
  row = 1,
  col = 0,
  style = 'minimal',
  border = 'rounded'
})
```

45.7 Health checks

Category: Neovim Features
Tags: health, check, diagnostics, system

Use `:checkhealth` to diagnose Neovim installation and plugin issues.

Example

```
:checkhealth           " check all health
:checkhealth nvim      " check Neovim core
:checkhealth telescope " check specific plugin
```

45.8 Lua configuration

Category: Neovim Features
Tags: lua, configuration, modern, scripting

Use Lua for configuration instead of Vimscript for better performance and modern syntax.

Example

```
-- ~/.config/nvim/init.lua
vim.opt.number = true
vim.opt.relativenumber = true
vim.keymap.set('n', '<leader>ff', '<cmd>Telescope find_files<cr>')
```

45.9 Multiple cursors simulation

Category: Neovim Features
Tags: cursor, multiple, editing

Use `cgn` after searching to change next match, then press `.` to repeat on subsequent matches.

Example

```
/word   " search for 'word'
cgn     " change next match
.       " repeat change on next match
```

45.10 Quick fix navigation

Category: Neovim Features
Tags: quickfix, navigation, errors

Use `:cn` to go to next error/item in quickfix list, `:cp` for previous, `:copen` to open quickfix window.

Example

```
:cn     " next quickfix item
:cp     " previous quickfix item
:copen  " open quickfix window
```

45.11 RPC and job control (jobstart)

Category: Neovim Features
Tags: rpc, jobs, async, communication

Use Neovim's job control and RPC capabilities for asynchronous operations.

Example

```
local job_id = vim.fn.jobstart({'ls', '-la'}, {
  on_stdout = function(_, data)
    for _, line in ipairs(data) do
```

```
      if line ~= '' then
        print(line)
      end
    end
  end
})
```

45.12 Statusline and tabline API

Category: Neovim Features
Tags: statusline, tabline, ui, customization

Customize statusline and tabline using Lua functions for dynamic content.

Example

```
function _G.custom_statusline()
  return '%f %m %r%=%l,%c %p%%'
end

vim.opt.statusline = '%!v:lua.custom_statusline()'
```

45.13 Tree-sitter text objects

Category: Neovim Features
Tags: treesitter, textobject, modern

Use `vaf` to select around function, `vif` for inside function, `vac` for around class (requires treesitter text objects).

Example

```
vaf  " select around function
vif  " select inside function
vac  " select around class
```

45.14 User commands

Category: Neovim Features
Tags: commands, user, custom, lua

Create custom user commands with Lua for better functionality and completion.

Example

```
vim.api.nvim_create_user_command('Hello', function(opts)
  print('Hello ' .. (opts.args or 'World'))
end, {
  nargs = '?',
  desc = 'Say hello to someone'
})
```

45.15 Virtual text

Category: Neovim Features
Tags: virtual, text, inline, diagnostics

Display virtual text inline for diagnostics, git blame, or other contextual information.

Example

```
local ns = vim.api.nvim_create_namespace('virtual_text')
vim.api.nvim_buf_set_extmark(0, ns, 0, -1, {
  virt_text = {{'  → This is virtual text', 'Comment'}},
  virt_text_pos = 'eol'
})
```

CHAPTER 46

Neovim terminal

46.1 Hidden terminal processes

Category: Terminal
Tags: terminal, hidden, background, process

Use hidden terminals to run background processes while maintaining editor workflow.

Example

```
:lua local buf = vim.api.nvim_create_buf(false, true)
:lua local job = vim.fn.termopen('tail -f logfile.log', {
  stdout_buffered = true,
  on_stdout = function(id, data)
    -- Process log data
  end
})
```

46.2 Split terminal workflows

Category: Terminal
Tags: terminal, split, workflow, development

Use terminal splits for integrated development workflows without leaving Neovim.

Example

```
:split | terminal         " horizontal split terminal
:vsplit | terminal        " vertical split terminal
:tabnew | terminal        " terminal in new tab
" Create persistent terminal splits for common tasks
```

46.3 Terminal REPL workflows

Category: Terminal
Tags: terminal, repl, workflow, interactive

Use terminal for REPL-driven development with language-specific interactive environ-

ments.

Example

```
:terminal python3         " Python REPL
:terminal node            " Node.js REPL
:terminal irb             " Ruby IRB
:terminal ghci            " Haskell GHCi
" Send code from buffer to REPL using mappings
```

46.4 Terminal and quickfix integration

Category: Terminal
Tags: terminal, quickfix, integration, errors

Use terminal output parsing to populate quickfix list with build errors and navigation.

Example

```
:set errorformat=%f:%l:%m  " set error format
:terminal make 2>&1 | tee build.log
" Then: :cfile build.log to load errors into quickfix
:lua vim.api.nvim_create_autocmd('TermClose', {
  callback = function() vim.cmd('cfile build.log') end
})
```

46.5 Terminal autocmd events

Category: Terminal
Tags: terminal, autocmd, events, TermOpen

Use terminal-specific autocommand events to customize terminal behavior and appearance.

Example

```
:autocmd TermOpen * setlocal nonumber norelativenumber
:autocmd TermOpen * nnoremap <buffer> <C-c> i<C-c>
:autocmd TermClose * echo "Terminal closed"
:autocmd TermEnter * startinsert  " enter insert mode
```

46.6 Terminal buffer job control

Category: Terminal
Tags: terminal, job, control, process

Use `jobstart()` and `jobstop()` to manage background processes and communicate with

terminal jobs.

Example

```
:lua local job_id = vim.fn.jobstart({'python', 'script.py'}, {
  on_stdout = function(id, data) print(table.concat(data, '\n')) end
})
:lua vim.fn.jobstop(job_id)
```

46.7 Terminal buffer naming

Category: Terminal
Tags: terminal, buffer, naming, identification

Use buffer naming to identify and switch between multiple terminal instances easily.

Example

```
:terminal ++title=server " named terminal buffer
:terminal ++title=build
:ls                      " shows named terminal buffers
:buffer server           " switch to named terminal
```

46.8 Terminal color and appearance

Category: Terminal
Tags: terminal, color, appearance, highlight

Use terminal-specific highlighting and color configuration for better visual integration.

Example

```
:hi Terminal ctermfg=white ctermbg=black
:hi TermCursor ctermfg=red ctermbg=red
:hi TermCursorNC ctermfg=white ctermbg=darkgray
:set termguicolors  " enable 24-bit colors in terminal
```

46.9 Terminal debugging integration

Category: Terminal
Tags: terminal, debugging, gdb, integration

Use terminal for integrated debugging sessions with GDB, Python debugger, or other CLI debuggers.

Example

```
:terminal gdb ./program   " GDB in terminal
:terminal python -m pdb script.py  " Python debugger
" Use terminal splits to debug while viewing source
:split | edit source.c | split | terminal gdb ./program
```

46.10 Terminal environment variables

Category: Terminal
Tags: terminal, environment, variables, env

Use environment variable control for terminal processes launched from Neovim.

Example

```
:let $EDITOR = 'nvim'
:let $TERM = 'xterm-256color'
:terminal env              " show environment
:terminal ENV_VAR=value command  " set env var for command
```

46.11 Terminal mode key mappings

Category: Terminal
Tags: terminal, mode, mappings, tnoremap

Use terminal mode mappings to customize key behavior inside built-in terminal emulator.

Example

```
:tnoremap <Esc> <C-\><C-n>                    " easier normal mode
:tnoremap <C-w>h <C-\><C-n><C-w>h            " window navigation
:tnoremap <A-h> <C-\><C-n><C-w>h            " alt+h navigation
:tnoremap <C-]> <C-\><C-n>:q<CR>            " quick close
```

46.12 Terminal output processing

Category: Terminal
Tags: terminal, output, processing, callback

Use terminal output callbacks to process terminal output and integrate with editor workflows.

Example

```
:lua vim.fn.termopen('make', {
  on_exit = function(job_id, exit_code, event_type)
    if exit_code == 0 then
      vim.cmd('echo "Build successful!"')
    else
      vim.cmd('copen')
    end
  end
})
```

46.13 Terminal plugin integration

Category: Terminal
Tags: terminal, plugin, integration, compatibility

Use terminal integration patterns that work well with common Neovim plugins and workflows.

Example

```
" Terminal-friendly settings
:autocmd TermOpen * setlocal statusline=%{b:term_title}
:autocmd TermOpen * lua vim.wo.winhighlight = "Normal:TermNormal"
:autocmd BufEnter term://* startinsert  " auto enter insert mode
```

46.14 Terminal process communication

Category: Terminal
Tags: terminal, process, communication, stdin

Use `chansend()` to send input to terminal processes programmatically.

Example

```
:lua local term_id = vim.fn.bufnr()
:lua vim.fn.chansend(term_id, "ls -la\n")
:lua vim.fn.chansend(term_id, {"python", "-c", "print('hello')", "\n"})
" Send commands to terminal buffer programmatically
```

46.15 Terminal scrollback and history

Category: Terminal
Tags: terminal, scrollback, history, buffer

Use `scrollback` option to control terminal history and access previous output in termi-

nal buffers.

Example

```
:set termguicolors        " enable 24-bit colors
:let g:terminal_scrollback_buffer_size = 10000
:terminal                 " open terminal
" In terminal: <C-\><C-n> then /pattern to search history
```

46.16 Terminal session persistence

Category: Terminal
Tags: terminal, session, persistence, restore

Use terminal session restoration to maintain terminal state across Neovim sessions.

Example

```
" In session file, terminals are saved as:
:terminal ++restore
" Or create custom session saving:
:lua function save_terminals()
  -- Custom logic to save terminal commands/state
end
```

46.17 Terminal size and dimensions

Category: Terminal
Tags: terminal, size, dimensions, rows, cols

Use terminal size options to create terminals with specific dimensions for different tasks.

Example

```
:terminal ++rows=20       " terminal with 20 rows
:terminal ++cols=80       " terminal with 80 columns
:20split | terminal       " split with specific height
:vertical 80split | terminal  " split with specific width
```

46.18 Terminal window management

Category: Terminal
Tags: terminal, window, management, layout

Use advanced window management for terminal-focused layouts and workflows.

Example

```
:tabnew | terminal        " dedicated terminal tab
:only | split | terminal  " editor above, terminal below
:vsplit | terminal | vertical resize 80  " side terminal
" Create terminal-focused layout commands
```

46.19 Terminal with specific shell

Category: Terminal
Tags: terminal, shell, specific, custom

Use `:terminal` with specific shell or command for customized terminal environments.

Example

```
:terminal bash            " specific shell
:terminal python3         " Python REPL
:terminal node            " Node.js REPL
:terminal zsh -c 'cd ~/project && zsh'  " custom environment
```

46.20 Terminal with working directory

Category: Terminal
Tags: terminal, working, directory, cwd

Use `++cwd` to start terminals in specific working directories for project-based workflows.

Example

```
:terminal ++cwd=~/project " start in specific directory
:split | terminal ++cwd=%:h " terminal in current file's directory
:lua vim.cmd('terminal ++cwd=' .. vim.fn.expand('%:h'))
```

CHAPTER 47

Normal mode (advanced)

47.1 Buffer navigation shortcuts

Category: Normal Mode
Tags: buffer, navigate, switch, file

Use `Ctrl+^` or `Ctrl+6` to switch to alternate buffer (previously edited file).

Example

```
Ctrl+^    " switch to alternate buffer
Ctrl+6    " same as Ctrl+^ (switch to alternate)
```

47.2 Case conversion commands

Category: Normal Mode
Tags: case, upper, lower, toggle, conversion

Use `~` to toggle case of character, `g~` with motion for range case toggle, `gu` for lowercase, `gU` for uppercase.

Example

```
~         " toggle case of character under cursor
g~w       " toggle case of word
guw       " lowercase word
gUw       " uppercase word
g~~       " toggle case of entire line
```

47.3 Change case of text

Category: Normal Mode
Tags: case, change, text, range

Use `g~` followed by motion to toggle case, `gu` for lowercase, `gU` for uppercase.

Example

```
g~$       " toggle case from cursor to end of line
guw       " lowercase word under cursor
gUiw      " uppercase inner word
g~ap      " toggle case of paragraph
```

47.4 Change operations

Category: Normal Mode
Tags: change, replace, word, line, text

Use `c` with motion to change (delete and enter insert mode), `cc` to change line, `C` to change to end.

Example

```
cw        " change word
cc        " change entire line
C         " change from cursor to end of line
ciw       " change inner word
cip       " change inner paragraph
```

47.5 Completion in insert mode trigger

Category: Normal Mode
Tags: completion, insert, keyword, file

Use `Ctrl+n` and `Ctrl+p` in insert mode for word completion, `Ctrl+x Ctrl+f` for filename completion.

Example

```
" In insert mode:
Ctrl+n    " next completion
Ctrl+p    " previous completion
Ctrl+x Ctrl+f  " filename completion
Ctrl+x Ctrl+l  " line completion
```

47.6 Delete characters and words

Category: Normal Mode
Tags: delete, character, word, backspace

Use `x` to delete character under cursor, `X` to delete before cursor, `dw` to delete word, `dd` to delete line.

Example

```
x         " delete character under cursor
X         " delete character before cursor
dw        " delete word
dd        " delete entire line
D         " delete from cursor to end of line
```

47.7 Digraph insertion

Category: Normal Mode
Tags: digraph, special, character, unicode

Use `Ctrl+k` in insert mode followed by two characters to insert special characters.

Example

```
" In insert mode:
Ctrl+k a'  " insert á (a with acute accent)
Ctrl+k e`  " insert è (e with grave accent)
Ctrl+k c,  " insert ç (c with cedilla)
Ctrl+k >>  " insert » (right guillemet)
```

47.8 Ex mode and command execution

Category: Normal Mode
Tags: ex, command, colon, execute

Use `:` to enter command-line mode, `Q` to enter Ex mode (rarely used).

Example

```
:         " enter command-line mode
Q         " enter Ex mode (exit with :vi)
```

47.9 File under cursor operations

Category: Normal Mode
Tags: file, cursor, edit, goto

Use `gf` to open file under cursor, `gF` to open file with line number.

Example

```
gf        " open file under cursor
gF        " open file under cursor with line number
Ctrl+w f  " open file under cursor in new window
```

```
Ctrl+w gf " open file under cursor in new tab
```

47.10 Filter through external command

Category: Normal Mode
Tags: filter, external, command, process

Use `!` with motion to filter text through external command, `!!` to filter current line.

Example

```
!}sort    " sort from cursor to end of paragraph
!!date    " replace current line with date
!5jsort   " sort next 5 lines
!Gsort    " sort from cursor to end of file
```

47.11 Fold operations

Category: Normal Mode
Tags: fold, unfold, toggle, code

Use `za` to toggle fold, `zo` to open fold, `zc` to close fold, `zR` to open all folds.

Example

```
za        " toggle fold at cursor
zo        " open fold at cursor
zc        " close fold at cursor
zR        " open all folds in buffer
zM        " close all folds in buffer
```

47.12 Format text

Category: Normal Mode
Tags: format, indent, text, alignment

Use `=` with motion to format/indent text, `==` to format current line.

Example

```
==        " format current line
=ap       " format around paragraph
=G        " format from cursor to end of file
gg=G      " format entire file
```

47.13 Go to column

Category: Normal Mode
Tags: column, position, horizontal, goto

Use `{number}|` to go to specific column number on current line.

Example

```
10|       " go to column 10
1|        " go to column 1 (beginning of line)
$         " go to end of line (last column)
```

47.14 Increment and decrement numbers

Category: Normal Mode
Tags: number, increment, decrement, arithmetic

Use `Ctrl+a` to increment number under cursor, `Ctrl+x` to decrement. Works with decimal, hex, octal, and binary.

Example

```
Ctrl+a    " increment number under cursor
Ctrl+x    " decrement number under cursor
5Ctrl+a   " increment by 5
```

Works on hex (0x1F), octal (017), binary (0b1010), and decimals.

47.15 Indent and outdent

Category: Normal Mode
Tags: indent, outdent, shift, tab

Use `>` to indent, `<` to outdent. Works with motions and counts.

Example

```
>>        " indent current line
<<        " outdent current line
>ap       " indent around paragraph
3>>       " indent next 3 lines
>G        " indent from cursor to end
```

47.16 Insert at line ends/beginnings

Category: Normal Mode
Tags: insert, line, beginning, end, multiple

Use `I` to insert at beginning of line, `A` to append at end of line.

Example

```
I         " insert at beginning of line (first non-blank)
A         " append at end of line
gI        " insert at column 1 (absolute beginning)
```

47.17 Join lines with space control

Category: Normal Mode
Tags: join, lines, space, merge

Use `J` to join current line with next (adds space), `gJ` to join without adding space.

Example

```
J         " join lines with space
gJ        " join lines without space
3J        " join current line with next 2 lines
```

47.18 Line completion and duplication

Category: Normal Mode
Tags: line, duplicate, copy, complete

Use `yyp` to duplicate current line, `Yp` for same effect, `yy` then `p` anywhere to paste line.

Example

```
yyp       " duplicate current line below
yyP       " duplicate current line above
Yp        " same as yyp (Y yanks line, p pastes)
```

47.19 Mark commands

Category: Normal Mode
Tags: mark, position, jump, navigate

Use `m{letter}` to set mark, `'{letter}` to jump to mark's line, `` `{letter}`` to jump to exact position.

Example

```
ma        " set mark 'a' at current position
'a        " jump to line of mark 'a' (first non-blank)
`a        " jump to exact position of mark 'a'
``        " jump to position before last jump
''        " jump to line before last jump
```

47.20 Open new lines

Category: Normal Mode
Tags: open, line, above, below, insert

Use `o` to open line below cursor, `O` to open line above cursor (both enter insert mode).

Example

```
o         " open new line below and enter insert mode
O         " open new line above and enter insert mode
3o        " open 3 new lines below
```

47.21 Put operations

Category: Normal Mode
Tags: put, paste, register, before, after

Use `p` to put (paste) after cursor, `P` to put before cursor. Works with any register content.

Example

```
p         " put after cursor/line
P         " put before cursor/line
"ap       " put from register 'a' after cursor
"0p       " put from yank register (register 0)
```

47.22 Record and replay macros

Category: Normal Mode
Tags: macro, record, replay, automation

Use `q{letter}` to start recording macro, `q` to stop, `@{letter}` to replay, `@@` to replay last macro.

Example

```
qa        " start recording macro to register 'a'
... commands ...
q         " stop recording
@a        " replay macro 'a'
@@        " replay last macro
5@a       " replay macro 'a' 5 times
```

47.23 Repeat last command

Category: Normal Mode
Tags: repeat, command, dot, redo

Use `.` to repeat the last change command. One of Vim's most powerful features for efficient editing.

Example

```
.         " repeat last change
dd.       " delete line, then repeat delete
cw foo<Esc>.  " change word to foo, then repeat on next word
```

47.24 Replace single character

Category: Normal Mode
Tags: replace, character, single, substitute

Use `r{char}` to replace character under cursor with {char}, `R` to enter replace mode.

Example

```
ra        " replace character under cursor with 'a'
r<Space>  " replace with space
R         " enter replace mode
```

47.25 Search under cursor

Category: Normal Mode
Tags: search, word, cursor, find, highlight

Use `*` to search forward for word under cursor, `#` to search backward.

Example

```
*         " search forward for word under cursor
#         " search backward for word under cursor
g*        " search forward for partial word match
g#        " search backward for partial word match
```

47.26 Spelling navigation

Category: Normal Mode
Tags: spell, navigation, error, correction

Use `]s` to go to next misspelled word, `[s` for previous, `z=` for suggestions.

Example

```
]s        " next misspelled word
[s        " previous misspelled word
z=        " show spelling suggestions
zg        " add word to good word list
zw        " add word as misspelled
```

47.27 Tag navigation

Category: Normal Mode
Tags: tag, definition, ctags, jump

Use `Ctrl+]` to jump to tag under cursor, `Ctrl+t` to return from tag jump.

Example

```
Ctrl+]    " jump to tag definition
Ctrl+t    " return from tag jump
g Ctrl+]  " show list of matching tags
```

47.28 Undo and redo

Category: Normal Mode
Tags: undo, redo, history, changes

Use `u` to undo last change, `Ctrl+r` to redo, `U` to undo all changes on current line.

Example

```
u         " undo last change
Ctrl+r    " redo last undone change
U         " undo all changes on current line
```

47.29 Visual selection commands

Category: Normal Mode
Tags: visual, select, line, block, character

Use `v` for character-wise visual, `V` for line-wise visual, `Ctrl+v` for block-wise visual.

Example

```
v         " start character-wise visual selection
V         " start line-wise visual selection
Ctrl+v    " start block-wise visual selection
gv        " reselect last visual selection
```

47.30 Window navigation

Category: Normal Mode
Tags: window, switch, navigate, split

Use `Ctrl+w` followed by direction to move between windows.

Example

```
Ctrl+w h  " move to left window
Ctrl+w j  " move to window below
Ctrl+w k  " move to window above
Ctrl+w l  " move to right window
Ctrl+w w  " cycle through windows
```

47.31 Yank operations

Category: Normal Mode
Tags: yank, copy, line, word, clipboard

Use `y` with motion to yank (copy), `yy` to yank line, `Y` to yank to end of line.

Example

```
yy        " yank entire line
yw        " yank word
y$        " yank from cursor to end of line
Y         " yank from cursor to end of line (same as y$)
yap       " yank around paragraph
```

CHAPTER 48
Performance

48.1 Disable unused features

Category: Performance
Tags: disable, features, optimization, settings

Disable unused built-in features to improve performance and reduce memory usage.

Example

```
vim.g.loaded_gzip = 1
vim.g.loaded_tar = 1
vim.g.loaded_tarPlugin = 1
vim.g.loaded_zip = 1
vim.g.loaded_zipPlugin = 1
vim.g.loaded_netrw = 1
vim.g.loaded_netrwPlugin = 1
```

48.2 Lazy load plugins

Category: Performance
Tags: lazy, loading, plugins, optimization

Use lazy loading for plugins that aren't needed immediately to improve startup time.

Example

```
-- lazy.nvim example
{
  "telescope.nvim",
  cmd = "Telescope",  -- load only when command is used
  keys = "<leader>ff" -- load only when key is pressed
}
```

48.3 Lazy-load plugins for faster startup

Category: Performance
Tags: lazy, plugins, startup, optimization, lazy.nvim

Use lazy-loading strategies with lazy.nvim to defer plugin loading until needed, significantly improving startup time.

Example

```
-- Lazy-load on file type:
{
  "fatih/vim-go",
  ft = "go",  -- Only load for Go files
}

-- Lazy-load on command:
{
  "mbbill/undotree",
  cmd = "UndotreeToggle",  -- Only load when command is used
}

-- Lazy-load on keymap:
{
  "folke/trouble.nvim",
  keys = {
    { "<leader>xx", "<cmd>TroubleToggle<cr>", desc = "Toggle Trouble" },
  },
}

-- Lazy-load on event:
{
  "nvim-lualine/lualine.nvim",
  event = "VeryLazy",  -- Load after startup
}

-- Conditional loading:
{
  "tpope/vim-fugitive",
  cond = function()
    return vim.fn.isdirectory(".git") == 1
  end,
}

-- Check startup time:
-- nvim --startuptime startup.log
```

48.4 Memory usage monitoring

Category: Performance
Tags: memory, monitoring, usage, debug

Monitor Neovim memory usage to identify memory leaks or excessive usage.

Example

```
:lua print(collectgarbage("count") .. " KB")  " current memory usage
:lua collectgarbage()  " force garbage collection
```

48.5 Optimize file type detection

Category: Performance
Tags: filetype, detection, performance

Use efficient filetype detection and disable unnecessary patterns.

Example

```
vim.g.do_filetype_lua = 1  -- use Lua for filetype detection
vim.g.did_load_filetypes = 0  -- don't use Vim script detection
```

48.6 Optimize line numbers

Category: Performance
Tags: numbers, relative, performance, display

Use relative line numbers only when needed, as they can impact performance on large files.

Example

```
vim.opt.number = true
vim.opt.relativenumber = false  -- disable for better performance
-- Or enable only in normal mode
vim.api.nvim_create_autocmd({'BufEnter', 'FocusGained', 'InsertLeave'}, {
  pattern = '*',
  command = 'set relativenumber'
})
```

48.7 Optimize updatetime

Category: Performance
Tags: updatetime, performance, responsiveness

Set appropriate updatetime for better responsiveness (default 4000ms is often too slow).

Example

```
vim.opt.updatetime = 250  -- faster completion and diagnostics
```

48.8 Profile Lua code

Category: Performance
Tags: profile, lua, performance, debug

Use built-in Lua profiler to identify performance bottlenecks in your config.

Example

```
-- Start profiler
vim.loop.fs_open('/tmp/profile.log', 'w', 438, function(err, fd)
  if not err then
    vim.loop.fs_close(fd)
  end
end)

-- Profile code
local start = vim.loop.hrtime()
-- your code here
local elapsed = vim.loop.hrtime() - start
print(string.format("Elapsed: %.2fms", elapsed / 1e6))
```

48.9 Profile startup time

Category: Performance
Tags: profile, startup, performance

Use `nvim --startuptime profile.log` to profile Neovim startup time.

Example

```
nvim --startuptime profile.log
```

48.10 Reduce redraw frequency

Category: Performance
Tags: redraw, display, performance, optimization

Use lazyredraw to improve performance during macros and complex operations.

Example

```
vim.opt.lazyredraw = true  -- don't redraw during macros
```

48.11 Syntax highlighting limits

Category: Performance
Tags: syntax, highlighting, limits, large files

Set limits for syntax highlighting to maintain performance on large files.

Example

```
vim.opt.synmaxcol = 200  -- don't highlight lines longer than 200 chars
vim.g.syntax_timeout = 1000  -- timeout after 1 second
```

48.12 Use swap files efficiently

Category: Performance
Tags: swap, files, memory, performance

Configure swap files for better performance and crash recovery.

Example

```
vim.opt.swapfile = true
vim.opt.directory = vim.fn.expand('~/.local/share/nvim/swap//')
vim.opt.updatecount = 100  -- write swap after 100 keystrokes
```

CHAPTER 49

Performance (advanced)

49.1 Autocommand optimization

Category: Performance Optimization Advanced
Tags: autocommand, event, performance, grouping

Optimize autocommand usage to reduce event processing overhead.

Example

```
-- Group related autocommands for better performance
local group = vim.api.nvim_create_augroup('PerformanceOptimization', { clear
↪  = true })

-- Use specific patterns instead of wildcards
vim.api.nvim_create_autocmd('BufReadPost', {
  group = group,
  pattern = {'*.py', '*.js', '*.lua'},  -- specific patterns
  callback = function()
    -- optimized callback
  end
})

-- Debounce frequent events
local timer = nil
vim.api.nvim_create_autocmd('CursorMoved', {
  group = group,
  callback = function()
    if timer then
      timer:stop()
    end
    timer = vim.loop.new_timer()
    timer:start(100, 0, vim.schedule_wrap(function()
      -- debounced action
    end))
  end
})
```

49.2 Buffer and window optimization

Category: Performance Optimization Advanced
Tags: buffer, window, memory, cleanup, optimization

Optimize buffer and window management for better memory usage.

Example

```
-- Auto-cleanup hidden buffers
vim.api.nvim_create_autocmd('BufHidden', {
  callback = function(args)
    if vim.bo[args.buf].buftype == 'nofile' then
      vim.api.nvim_buf_delete(args.buf, {})
    end
  end
})

-- Limit number of open buffers
vim.opt.hidden = true
vim.opt.maxmem = 2000000  -- 2GB memory limit
vim.opt.maxmemtot = 4000000  -- 4GB total memory limit
```

49.3 Completion system optimization

Category: Performance Optimization Advanced
Tags: completion, cmp, performance, async, cache

Optimize completion systems for faster and more responsive completions.

Example

```
-- Optimize nvim-cmp performance
require('cmp').setup({
  performance = {
    debounce = 150,
    throttle = 30,
    fetching_timeout = 200,
    confirm_resolve_timeout = 80,
    async_budget = 1,
    max_view_entries = 50,
  },

  -- Limit completion sources for performance
  sources = {
    { name = 'nvim_lsp', max_item_count = 20 },
    { name = 'buffer', max_item_count = 10, keyword_length = 3 },
    { name = 'path', max_item_count = 10 },
  },
})
```

49.4 Concurrent operations optimization

Category: Performance Optimization Advanced
Tags: concurrent, async, parallel, threading, performance

Implement concurrent operations for better performance and responsiveness.

Example

```
-- Parallel file processing
local function process_files_concurrent(files, processor, callback)
  local results = {}
  local completed = 0

  for i, file in ipairs(files) do
    vim.loop.fs_open(file, 'r', 438, function(err, fd)
      if not err then
        vim.loop.fs_read(fd, 4096, 0, function(err2, data)
          vim.loop.fs_close(fd)
          if not err2 then
            results[i] = processor(data)
          end
          completed = completed + 1

          if completed == #files and callback then
            callback(results)
          end
        end)
      end
    end)
  end
end

-- Debounced operations
local debounce_timers = {}
local function debounce(key, fn, delay)
  if debounce_timers[key] then
    debounce_timers[key]:stop()
  end

  debounce_timers[key] = vim.defer_fn(function()
    fn()
    debounce_timers[key] = nil
  end, delay)
end
```

49.5 Diff and merge performance optimization

Category: Performance Optimization Advanced
Tags: diff, merge, algorithm, performance, comparison

Optimize diff operations and merge performance for large files.

Example

```
-- Configure diff algorithm for better performance
vim.opt.diffopt = {
  'internal',          -- use internal diff algorithm
  'filler',           -- show filler lines
  'closeoff',         -- close diff when one window closes
  'hiddenoff',        -- turn off diff when buffer becomes hidden
  'algorithm:patience' -- use patience algorithm for better diffs
}

-- Optimize for large diffs
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    if vim.wo.diff then
      -- Disable expensive features during diff
      vim.wo.cursorline = false
      vim.wo.relativenumber = false
      vim.opt_local.syntax = 'off'
    end
  end
})
```

49.6 Display and rendering optimization

Category: Performance Optimization Advanced
Tags: display, render, redraw, terminal, optimization

Optimize display rendering and terminal performance.

Example

```
-- Terminal-specific optimizations
if os.getenv('TERM') == 'xterm-256color' then
  vim.opt.ttyfast = true
  vim.opt.lazyredraw = true
end

-- Reduce redraw frequency
vim.opt.scrolljump = 8       -- scroll 8 lines at a time
vim.opt.sidescroll = 15      -- horizontal scroll 15 chars
vim.opt.sidescrolloff = 5    -- horizontal scroll offset

-- Optimize cursor movement
vim.opt.cursorline = false   -- disable cursor line highlighting
vim.opt.cursorcolumn = false -- disable cursor column

-- Optimize sign column
vim.opt.signcolumn = 'yes:1' -- always show 1 sign column
```

49.7 File I/O optimization

Category: Performance Optimization Advanced
Tags: file, io, read, write, performance, async

Optimize file reading and writing operations for better performance.

Example

```
-- Optimize file reading
vim.opt.fsync = false        -- disable fsync for faster writes
vim.opt.swapsync = ""        -- disable swap sync

-- Async file operations
local function async_read_file(path, callback)
  vim.loop.fs_open(path, 'r', 438, function(err, fd)
    if not err then
      vim.loop.fs_fstat(fd, function(err2, stat)
        if not err2 then
          vim.loop.fs_read(fd, stat.size, 0, function(err3, data)
            vim.loop.fs_close(fd)
            if callback then callback(err3, data) end
          end)
        end
      end)
    end
  end)
end
```

49.8 LSP performance optimization

Category: Performance Optimization Advanced
Tags: lsp, language, server, performance, debounce

Optimize Language Server Protocol interactions for better responsiveness.

Example

```
-- Debounce LSP diagnostics
vim.lsp.handlers['textDocument/publishDiagnostics'] = vim.lsp.with(
  vim.lsp.diagnostic.on_publish_diagnostics, {
    update_in_insert = false,  -- don't update diagnostics in insert mode
    severity_sort = true,
    virtual_text = false,      -- disable virtual text for performance
  }
)

-- Optimize LSP client settings
local clients = vim.lsp.get_active_clients()
for _, client in ipairs(clients) do
  client.server_capabilities.semanticTokensProvider = nil  -- disable
  ↪  semantic tokens
```

```
end

-- Limit concurrent LSP requests
vim.lsp.buf.format({ timeout_ms = 2000, async = true })
```

49.9 Large file handling optimization

Category: Performance Optimization Advanced
Tags: large, file, handling, performance, memory

Implement specialized handling for large files to maintain performance.

Example

```
-- Large file detection and optimization
local function optimize_for_large_file(bufnr)
  local file_size = vim.fn.getfsize(vim.api.nvim_buf_get_name(bufnr))

  if file_size > 1024 * 1024 then  -- > 1MB
    -- Disable expensive features
    vim.bo[bufnr].syntax = 'off'
    vim.bo[bufnr].filetype = ''
    vim.bo[bufnr].swapfile = false
    vim.bo[bufnr].undolevels = -1
    vim.wo.foldmethod = 'manual'
    vim.wo.list = false

    -- Show warning
    vim.notify('Large file detected - some features disabled for
    ↪  performance')
  end
end

vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function(args)
    optimize_for_large_file(args.buf)
  end
})
```

49.10 Memory management and garbage collection

Category: Performance Optimization Advanced
Tags: memory, garbage, collection, lua, cleanup

Implement efficient memory management and garbage collection strategies.

Example

```
-- Monitor memory usage
local function check_memory()
  local mem_kb = collectgarbage('count')
  if mem_kb > 100000 then  -- 100MB
    collectgarbage('collect')
    print(string.format('Memory cleaned: %.2f MB', mem_kb/1024))
  end
end

-- Periodic garbage collection
local gc_timer = vim.loop.new_timer()
gc_timer:start(60000, 60000, vim.schedule_wrap(check_memory))  -- every
↪  minute

-- Clean up on buffer delete
vim.api.nvim_create_autocmd('BufDelete', {
  callback = function()
    collectgarbage('collect')
  end
})
```

49.11 Network and remote file optimization

Category: Performance Optimization Advanced
Tags: network, remote, file, ssh, ftp, optimization

Optimize network operations and remote file editing performance.

Example

```
-- Configure network timeouts
vim.g.netrw_timeout = 10     -- 10 second timeout
vim.g.netrw_retry = 3        -- retry 3 times

-- Optimize remote file editing
vim.api.nvim_create_autocmd('BufReadPre', {
  pattern = {'sftp://*', 'scp://*', 'ftp://*'},
  callback = function()
    -- Disable expensive features for remote files
    vim.opt_local.backup = false
    vim.opt_local.writebackup = false
    vim.opt_local.swapfile = false
    vim.opt_local.undofile = false
  end
})

-- Async remote operations
local function async_remote_read(url, callback)
  local job = vim.fn.jobstart({'curl', '-s', url}, {
    on_stdout = function(_, data, _)
      if callback then callback(data) end
```

```
    end
  })
end
```

49.12 Optimize plugin loading strategy

Category: Performance Optimization Advanced
Tags: plugin, loading, lazy, startup, optimization

Implement sophisticated plugin loading strategies for minimal startup time.

Example

```
-- Conditional loading based on file size
local function should_load_heavy_plugins()
  local file_size = vim.fn.getfsize(vim.fn.expand('%'))
  return file_size < 1024 * 1024  -- Load only for files < 1MB
end

-- Load plugins conditionally
if should_load_heavy_plugins() then
  require('expensive-plugin').setup()
end
```

49.13 Plugin configuration caching

Category: Performance Optimization Advanced
Tags: cache, config, plugin, startup, optimization

Implement configuration caching for faster plugin loading.

Example

```
-- Cache heavy computations
local cache = {}
local function get_cached_config(key, compute_fn)
  if not cache[key] then
    cache[key] = compute_fn()
  end
  return cache[key]
end

-- Example usage
local function expensive_config()
  -- expensive computation
  return { complex = 'configuration' }
end

local config = get_cached_config('my_plugin', expensive_config)
```

```
-- Persistent caching across sessions
local cache_file = vim.fn.stdpath('cache') .. '/my_config.json'
local function load_cache()
  if vim.fn.filereadable(cache_file) == 1 then
    local content = vim.fn.readfile(cache_file)
    return vim.fn.json_decode(table.concat(content))
  end
  return {}
end
```

49.14 Search and regex performance tuning

Category: Performance Optimization Advanced
Tags: search, regex, performance, timeout, optimization

Optimize search operations and regex performance for better responsiveness.

Example

```
-- Set search timeouts
vim.opt.redrawtime = 5000    -- 5 seconds max for syntax highlighting
vim.opt.maxmempattern = 2000 -- memory limit for pattern matching

-- Optimize search behavior
vim.opt.ignorecase = true
vim.opt.smartcase = true
vim.opt.hlsearch = false     -- disable search highlighting for performance

-- Use faster search methods
vim.keymap.set('n', '*', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.setreg('/', '\\<' .. word .. '\\>')
  vim.cmd('normal! n')
end)
```

49.15 Startup time profiling and analysis

Category: Performance Optimization Advanced
Tags: profile, startup, analysis, benchmark, timing

Implement comprehensive startup profiling and performance analysis.

Example

```
-- Startup timing measurement
local start_time = vim.loop.hrtime()

vim.api.nvim_create_autocmd('VimEnter', {
```

```
  callback = function()
    local end_time = vim.loop.hrtime()
    local startup_time = (end_time - start_time) / 1e6  -- convert to
    ↪  milliseconds
    print(string.format('Startup time: %.2f ms', startup_time))
  end
})

-- Profile plugin loading times
local plugin_times = {}
local original_require = require

require = function(module)
  local start = vim.loop.hrtime()
  local result = original_require(module)
  local elapsed = (vim.loop.hrtime() - start) / 1e6

  plugin_times[module] = (plugin_times[module] or 0) + elapsed
  return result
end

-- Show plugin timings
vim.api.nvim_create_user_command('ProfileReport', function()
  local sorted = {}
  for module, time in pairs(plugin_times) do
    table.insert(sorted, {module, time})
  end
  table.sort(sorted, function(a, b) return a[2] > b[2] end)

  for _, entry in ipairs(sorted) do
    print(string.format('%-30s: %.2f ms', entry[1], entry[2]))
  end
end, {})
```

49.16 Syntax and highlighting optimization

Category: Performance Optimization Advanced
Tags: syntax, highlight, treesitter, performance

Optimize syntax highlighting for better performance on large files.

Example

```
-- Disable syntax for large files
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    local file_size = vim.fn.getfsize(vim.fn.expand('%'))
    if file_size > 1024 * 1024 then  -- 1MB
      vim.opt_local.syntax = 'off'
      vim.opt_local.filetype = ''
      vim.opt_local.undolevels = -1
    end
```

```
  end
})

-- Optimize treesitter for performance
require('nvim-treesitter.configs').setup({
  highlight = {
    enable = true,
    disable = function(lang, buf)
      local max_filesize = 100 * 1024 -- 100 KB
      local ok, stats = pcall(vim.loop.fs_stat,
      ↪  vim.api.nvim_buf_get_name(buf))
      if ok and stats and stats.size > max_filesize then
        return true
      end
    end,
  },
})
```

CHAPTER 50

Registers

50.1 Append to register

Category: Registers
Tags: register, append, uppercase

Use uppercase letter to append to a register instead of replacing its contents.

Example

```
"ayy   " yank line into register a
"Ayy   " append line to register a (note uppercase A)
"ap    " paste both lines from register a
```

50.2 Clear specific register

Category: Registers
Tags: register, clear, empty, macro

Use `q{register}q` to clear/empty a specific register by recording an empty macro.

Example

```
qAq    " clear register 'A'
qaq    " clear register 'a'
q1q    " clear register '1'
q:q    " clear command register
```

50.3 Delete without affecting register

Category: Registers
Tags: delete, register, blackhole

Use `"_d` to delete text without affecting the default register (sends to blackhole register).

Example

```
"_d  " delete to blackhole register
```

50.4 Get current buffer path in register

Category: Registers
Tags: buffer, path, filename, register, clipboard

Use `"%p` to paste current filename, `:let @+=@%` to copy buffer name to system clipboard.

Example

```
"%p            " paste current filename
":let @+=@%     " copy current buffer name to system clipboard
":let @"=@%     " copy current buffer name to default register
```

50.5 Paste without overwriting register

Category: Registers
Tags: paste, register, overwrite, visual, multiple

Use `P` (capital) in visual mode to paste without overwriting the register, allowing multiple pastes.

Example

```
" Select text, then:
P     " paste without overwriting register (can repeat)
p     " paste and overwrite register with selected text
```

50.6 Set register manually

Category: Registers
Tags: register, set, manual

Use `:let @a='text'` to manually set the contents of register a.

Example

```
:let @a='hello world'  " set register a to 'hello world'
```

50.7 System clipboard

Category: Registers
Tags: clipboard, system, yank

Use `"+y` to yank to the system clipboard and `"+p` to paste from the system clipboard.

Example

```
"+y  " yank to system clipboard
"+p  " paste from system clipboard
```

50.8 Use specific register

Category: Registers
Tags: registers, yank, specific

Use `"xy` to yank into specific register x. Replace x with any letter or number.

Example

```
"ay  " yank into register a
"bp  " paste from register b
```

50.9 View registers

Category: Registers
Tags: registers, clipboard, view

Use `:registers` to show the contents of all registers.

Example

```
:registers
```

CHAPTER 51

Search

51.1 Advanced search and replace with regex

Category: Search
Tags: replace, regex, advanced

Use `:%s/\v(foo|bar)/baz/g` to replace either 'foo' or 'bar' with 'baz' using very magic mode.

Example

```
:%s/\v(foo|bar)/baz/g  " replace foo or bar with baz
```

51.2 Case insensitive search

Category: Search
Tags: search, case, insensitive

Use `/pattern\c` for case insensitive search, or `/pattern\C` for case sensitive search.

Example

```
/hello\c  " case insensitive
/hello\C  " case sensitive
```

51.3 Delete lines containing pattern

Category: Search
Tags: delete, pattern, global, lines

Use `:g/pattern/d` to delete all lines containing a pattern, or `:g!/pattern/d` to delete lines NOT containing pattern.

Example

```
:g/pattern/d        " delete all lines containing 'pattern'
:g/^\s*$/d          " delete empty or whitespace-only lines
:g!/error/d         " delete lines NOT containing 'error'
:g!/error\|warn/d   " delete lines NOT containing 'error' or 'warn'
```

51.4 Global command with pattern

Category: Search
Tags: global, command, execute, pattern

Use `:g/pattern/command` to execute a command on all lines matching pattern.

Example

```
:g/TODO/d        " delete all lines containing TODO
:g/function/p    " print all lines containing 'function'
:g/error/s/old/new/g " replace 'old' with 'new' on lines with 'error'
```

51.5 Global search and replace

Category: Search
Tags: replace, global, substitute

Use `:%s/old/new/g` to replace all occurrences of 'old' with 'new' in the entire file.

Example

```
:%s/foo/bar/g  " replace all 'foo' with 'bar'
```

51.6 Multi-line search pattern

Category: Search
Tags: search, multiline, pattern, regex

Use `\_s` for whitespace including newlines, `\_.*` to match across lines in search patterns.

Example

```
/function\_s*name    " function followed by whitespace/newlines
/start\_.*end        " match start to end across lines
```

51.7 Negative search (inverse)

Category: Search
Tags: search, negative, inverse, exclude

Use `:v/pattern/command` or `:g!/pattern/command` to execute command on lines NOT matching pattern.

Example

```
:v/pattern/d     " delete lines NOT containing pattern
:g!/TODO/p       " print lines NOT containing TODO
```

51.8 Perform change on lines returned by vimgrep regex search

Category: Search
Tags: replace, regex, search, vimgrep, cdo

Suppose that you have a set of .html documents and you want to find all <a> tags that have some attribute in it, for example: `text-red`. You want to replace that attribute with `text-blue`. Do the following:

Example

```
:vimgrep /<a [^>]*text-red[^>]*>/gj **/*.html
```

This will create a quickfix list made of lines that match the regular expression and open the file with the first matching line highlighted. After that you can execute the substitution:

Example

```
:cdo s/text-red/text-blue/gc
```

Thanks to `c` flag you'll have a cnahce to approve every change. Note that `cfdo` would perform changes on matched FILES, while `cdo` works on matched lines. Also in substitution command use `s/`, not `%s/` because the first one is executed on the current line and the second one would process the whole doucment.

51.9 Recursive file search

Category: Search
Tags: vimgrep, recursive, files

Use `:vimgrep /pattern/ **/*.ext` to search for pattern recursively in files with specific extension.

Example

```
:vimgrep /pattern/ **/*.lua  " search in all .lua files
```

51.10 Remove search highlighting

Category: Search
Tags: search, highlight, remove

Use `:nohl` to remove search highlighting after performing a search.

Example

```
:nohl
```

51.11 Repeat last search in substitution

Category: Search
Tags: substitute, repeat, search

Use `:%s//replacement/g` to use the last search pattern in substitution command.

Example

```
:%s//new_text/g  " replace last searched pattern with new_text
```

51.12 Replace only within visual selection

Category: Search
Tags: replace, visual, selection, visual-pattern

Use `\%V` in search pattern to restrict replacement to only the visual selection area.

Example

```
" After making visual selection:
:'<,'>s/\%Vold/new/g   " replace only within selection
" \%V ensures replacement only happens in selected text
```

51.13 Search and execute command

Category: Search
Tags: search, execute, global, command

Use `:g/pattern/command` to execute command on all lines matching pattern.

Example

```
:g/TODO/d        " delete all lines containing TODO
:g/^$/d          " delete all empty lines
:g/pattern/p     " print all lines matching pattern
```

51.14 Search backward

Category: Search
Tags: search, backward, reverse

Use `?pattern` to search backward for a pattern. Press `n` to go to next match and `N` for previous.

Example

```
?hello  " search backward for 'hello'
n       " next match (backward)
N       " previous match (forward)
```

51.15 Search in selection

Category: Search
Tags: replace, selection, range

Use `:'<,'>s/old/new/g` to replace only in visual selection.

Example

```
:'<,'>s/foo/bar/g  " replace in selection
```

51.16 Search with offset

Category: Search
Tags: search, offset, cursor, position

Use `/pattern/+n` to position cursor n lines after match, or `/pattern/-n` for n lines before.

Example

```
/function/+2     " position cursor 2 lines after 'function'
/end/-1          " position cursor 1 line before 'end'
```

51.17 Search word boundaries with very magic

Category: Search
Tags: search, regex, word, boundary, magic

Use \v for very magic mode to make regex more intuitive, or \<word\> for exact word boundaries.

Example

```
/\v(hello|world)  " search for 'hello' or 'world' (very magic)
/\<function\>     " search for exact word 'function'
/\vd+             " search for one or more digits
```

51.18 Very magic search mode

Category: Search
Tags: search, regex, magic

Use \v at start of search pattern for ”very magic” mode, making regex more intuitive (similar to other languages).

Example

```
/\v(hello|world)  " search for 'hello' or 'world'
/\vd+             " search for one or more digits
```

CHAPTER 52

Session

52.1 Ex commands - arglist and project files

Category: Session
Tags: ex, arglist, args, project, files

Use `:args` to set argument list, `:argadd` to add files, `:next`/`:prev` to navigate, `:argdo` for commands on all.

Example

```
:args *.py       " set arglist to all Python files
:argadd test.py  " add file to arglist
:next            " go to next file in arglist
:prev            " go to previous file
:argdo %s/old/new/g  " run command on all files
```

52.2 Ex commands - session options

Category: Session
Tags: ex, session, options, save, restore

Use `:set sessionoptions` to control what gets saved, `:mksession {file}` for custom filename, `:source` to restore.

Example

```
:set sessionoptions=buffers,curdir,folds,help,tabpages,winsize
:mksession mysession.vim  " save to custom file (fails if the file already
↪  exists)
:mksession mysession.vim! " save with custom name (overwrites possibly
↪  existing file)
:source mysession.vim     " restore specific session
```

52.3 Ex commands - viminfo and shada

Category: Session
Tags: ex, viminfo, shada, history, persistent

Use `:wviminfo` to write viminfo, `:rviminfo` to read, `:wshada` and `:rshada` for Neovim's shada file.

Example

```
:wshada          " write shada file
:rshada          " read shada file
:wshada backup.shada  " save to specific file
:rshada backup.shada  " read from specific file
```

52.4 Ex commands - working with multiple files

Category: Session
Tags: ex, multiple, files, bufdo, windo, tabdo

Use `:bufdo` for all buffers, `:windo` for all windows, `:tabdo` for all tabs to execute commands across multiple contexts.

Example

```
:bufdo %s/old/new/ge  " substitute in all buffers
:windo set number     " set line numbers in all windows
:tabdo close          " close all tabs
:argdo write          " save all files in arglist
```

52.5 Session management

Category: Session
Tags: session, save, restore

Use `:mksession!` to save session and `:source Session.vim` to restore it.

Example

```
:mksession!        " save session
:source Session.vim " restore session
```

CHAPTER 53

System

53.1 Async shell commands

Category: System
Tags: async, shell, lua

Use `vim.loop.spawn()` to run shell commands asynchronously without blocking Neovim.

Example

```
:lua vim.loop.spawn("ls", {args={"-la"}}, function() print("Done!") end)
```

53.2 Confirm dangerous operations

Category: System
Tags: confirm, dialog, save, quit, dangerous

Use `:confirm {command}` to show confirmation dialog for potentially dangerous operations.

Example

```
:confirm quit     " show dialog if unsaved changes exist
:confirm qall     " confirm before quitting all windows
:confirm write    " confirm before writing file
:confirm !rm %    " confirm before executing external command
```

53.3 Ex commands - external command execution

Category: System
Tags: ex, external, command, shell, bang

Use `:!command` to run external commands, `:!!` to repeat last command, `:silent !` to run without output.

Example

```
:!ls              " run ls command
:!make            " run make command
:!!               " repeat last external command
:silent !make     " run make without showing output
```

53.4 Ex commands - file system operations

Category: System
Tags: ex, file, system, mkdir, delete, rename

Use `:!mkdir`, `:!rm`, `:!mv` for file operations, or use Neovim's built-in file functions.

Example

```
:!mkdir newdir        " create directory
:!rm file.txt         " delete file
:!mv old.txt new.txt  " rename file
:!cp file.txt backup.txt  " copy file
```

53.5 Ex commands - make and quickfix

Category: System
Tags: ex, make, quickfix, error, jump

Use `:make` to run make command, `:copen` for quickfix window, `:cnext`/`:cprev` to navigate errors.

Example

```
:make           " run make command
:make clean     " run make with target
:copen          " open quickfix window
:cnext          " jump to next error
:cprev          " jump to previous error
:cfirst         " jump to first error
:clast          " jump to last error
```

53.6 Ex commands - shell and environment

Category: System
Tags: ex, shell, environment, cd, pwd

Use `:shell` to start shell, `:cd` to change directory, `:pwd` to show current directory, `:lcd` for local directory.

Example

```
:shell          " start interactive shell
:cd /home/user " change to directory
:pwd            " show current directory
:lcd ~/project " change local directory for current window
```

53.7 Execute line as command

Category: System
Tags: execute, line, command, shell

Use `!!` to replace current line with output of line executed as shell command.

Example

```
!!date              " replace line with current date
!!ls                " replace line with directory listing
!!pwd               " replace line with current directory
" Or use visual selection:
V!!sort             " sort selected lines in place
```

53.8 Read command output into buffer

Category: System
Tags: command, output, read, external

Use `:r !command` to read external command output into current buffer at cursor position.

Example

```
:r !ls              " insert file listing
:r !date            " insert current date
:0r !whoami         " insert username at top of file
:r !curl -s url     " insert web content
```

53.9 Redirect command output

Category: System
Tags: redirect, output, capture, redir

Use `:redir` to redirect command output to variables, registers, or files for later use.

Example

```
:redir @a           " redirect to register 'a'
:set all           " run commands
:redir END         " stop redirecting
"ap                " paste captured output

:redir > output.txt " redirect to file
:echo "hello"      " commands get redirected
:redir END         " stop redirecting
```

53.10 Write buffer to command

Category: System
Tags: write, command, pipe, external

Use `:w !command` to pipe buffer contents to external command without saving file.

Example

```
:w !wc              " count words without saving
:w !python          " execute buffer as Python script
:%w !sh             " execute entire buffer as shell script
:'<,'>w !sort       " sort selected lines
```

CHAPTER 54

Tabs

54.1 Close tab

Category: Tabs
Tags: tab, close, exit

Use `:tabclose` to close current tab.

Example

```
:tabclose
```

54.2 Navigate tabs

Category: Tabs
Tags: tab, navigate, switch

Use `gt` to go to next tab, `gT` to go to previous tab, or `{number}gt` to go to specific tab.

Example

```
gt   " next tab
gT   " previous tab
2gt  " go to tab 2
```

54.3 Open commands in new tabs

Category: Tabs
Tags: tab, command, open, prefix

Use `:tab {command}` to open any command in a new tab instead of current window.

Example

```
:tab split        " open split in new tab
:tab help motion  " open help in new tab
:tab edit file.txt " open file in new tab
```

```
:tab ball         " open all buffers in tabs
```

54.4 Open new tab

Category: Tabs
Tags: tab, new, open

Use `:tabnew` or `:tabedit {file}` to open a new tab, optionally with a file.

Example

```
:tabnew
:tabedit file.txt
```

CHAPTER 55

Terminal

55.1 Open terminal in current window

Category: Terminal
Tags: terminal, open, current, window

Use `:terminal` or `:term` to open terminal in current window.

Example

```
:terminal     " open terminal in current window
:term         " shorthand for :terminal
:term bash    " open specific shell
```

55.2 Open terminal in new window

Category: Terminal
Tags: terminal, window, split, tab

Use `:sp | terminal` for horizontal split, `:vsp | terminal` for vertical split, `:tabe | terminal` for new tab.

Example

```
:sp | terminal   " horizontal split terminal
:vsp | terminal  " vertical split terminal
:tabe | terminal " terminal in new tab
```

55.3 Send commands to terminal

Category: Terminal
Tags: terminal, command, send

Use `vim.api.nvim_chan_send()` to send commands to terminal buffer from Lua.

Example

```
:lua vim.api.nvim_chan_send(terminal_job_id, "ls\n")
```

55.4 Terminal insert mode

Category: Terminal
Tags: terminal, insert, mode, interaction

Use `i`, `a`, or `A` to return to terminal mode from normal mode for terminal interaction.

Example

```
" From terminal normal mode:
i   " enter terminal mode at cursor
a   " enter terminal mode after cursor
A   " enter terminal mode at end of line
```

55.5 Terminal mode - execute one command

Category: Terminal
Tags: terminal, mode, execute, command

Use `Ctrl+\ Ctrl+o` to execute one normal mode command and return to terminal mode.

Example

```
" In terminal mode:
Ctrl+\ Ctrl+o  " execute one normal mode command
```

55.6 Terminal mode - exit to normal mode

Category: Terminal
Tags: terminal, mode, exit, normal

Use `Ctrl+\ Ctrl+n` to exit terminal mode and go to normal mode.

Example

```
" In terminal mode:
Ctrl+\ Ctrl+n  " exit to normal mode
```

55.7 Terminal mode - key forwarding

Category: Terminal
Tags: terminal, keys, forwarding, passthrough

All keys except `Ctrl+\` are forwarded directly to the terminal job. Use `Ctrl+\` as escape prefix for Neovim commands.

Example

```
" In terminal mode:
ls<Enter>           " sent to terminal
Ctrl+c              " sent to terminal (interrupt)
Ctrl+\ Ctrl+n       " Neovim command (exit to normal)
```

55.8 Terminal scrollback buffer

Category: Terminal
Tags: terminal, scrollback, buffer, history

In normal mode, you can navigate terminal scrollback buffer like any other buffer using standard movement commands.

Example

```
" In terminal normal mode (after Ctrl+\ Ctrl+n):
gg    " go to top of scrollback
G     " go to bottom
/text " search in terminal output
```

CHAPTER 56

Text manipulation

56.1 Align numbers at decimal point

Category: Text Manipulation
Tags: align, numbers, decimal, format

Use visual selection and substitute to align decimal numbers at their decimal points.

Example

```
" Select lines with numbers, then:
:'<,'>s/\(\d\+\)\.\(\d\+\)/\=printf("%6.2f", submatch(0))/
" Or use Align plugin:
:'<,'>Align \.
```

56.2 Binary number operations

Category: Text Manipulation
Tags: binary, numbers, conversion, base

Convert and manipulate binary numbers using expressions and external tools.

Example

```
" In insert mode, convert decimal to binary:
Ctrl+r =printf("%b", 42)<Enter>   " inserts 101010

" Convert binary to decimal:
Ctrl+r =str2nr("101010", 2)<Enter>  " inserts 42

" Format as hex:
Ctrl+r =printf("0x%x", 42)<Enter>   " inserts 0x2a
```

56.3 Comment and uncomment blocks

Category: Text Manipulation
Tags: comment, uncomment, code, blocks

Add or remove comment markers from blocks of code.

Example

```
" For line comments (e.g., //):
:'<,'>s/^/\/\/ /    " add comment
:'<,'>s/^\/\/ //    " remove comment

" For block comments:
:'<,'>s/^/\/* /     " add start comment
:'<,'>s/$/ *\//     " add end comment

" Using substitute with confirmation:
:%s/^/# /gc         " add # comments with confirmation
```

56.4 Convert tabs to spaces

Category: Text Manipulation
Tags: tabs, spaces, convert, whitespace

Use `:retab` to convert tabs to spaces using current tabstop setting, or `:set expandtab | retab` to convert and set future tabs as spaces.

Example

```
:retab              " convert tabs to spaces
:set expandtab | retab  " convert and set expandtab
```

56.5 Create incremental sequence with g Ctrl+a

Category: Text Manipulation
Tags: increment, sequence, numbers, visual, ctrl-a

Use `g Ctrl+a` in visual block mode to create incremental number sequences instead of incrementing all numbers by the same amount.

Example

```
" Select multiple lines with numbers, then:
g<C-a>    " creates 1,2,3,4... sequence
<C-a>     " would increment all by 1
```

56.6 Delete blank lines

Category: Text Manipulation
Tags: delete, blank, lines

Use `:g/^$/d` to delete all blank/empty lines in the buffer.

Example

```
:g/^$/d  " delete blank lines
```

56.7 Delete character operations

Category: Text Manipulation
Tags: delete, character, cursor

Use `x` to delete character under cursor and `X` to delete character before cursor.

Example

```
x   " delete character under cursor
X   " delete character before cursor
5x  " delete 5 characters forward
```

56.8 Delete non-matching lines

Category: Text Manipulation
Tags: delete, pattern, inverse

Use `:v/pattern/d` to delete all lines that do NOT match the pattern.

Example

```
:v/TODO/d  " delete lines that don't contain 'TODO'
```

56.9 Duplicate lines or selections

Category: Text Manipulation
Tags: duplicate, copy, lines, repeat

Duplicate current line or selected text efficiently.

Example

```
yyp                " duplicate current line (yank and paste)
"ayy"ap            " duplicate line using register a
:'<,'>co'<-1       " duplicate selected lines above
:'<,'>co'>         " duplicate selected lines below
:.,.+5co$          " copy lines to end of file
```

56.10 Filter text through external commands

Category: Text Manipulation
Tags: filter, external, command, !, pipe, processing

Use `!{motion}{command}` to filter text through external commands for text processing.

Example

```
!}sort              " sort paragraph (motion: })
:'<,'>!sort         " sort visual selection
!Gtidy -iq -xml     " format XML until end of file
:%!python3 -m json.tool  " format entire file as JSON
:10,20!nl           " add line numbers to lines 10-20
```

56.11 Format paragraph

Category: Text Manipulation
Tags: format, paragraph, wrap

Use `gqap` to format/wrap a paragraph according to textwidth.

Example

```
gqap  " format around paragraph
```

56.12 Generate increasing numbers column

Category: Text Manipulation
Tags: numbers, increment, column, sequence, generate

Generate a column of increasing numbers using visual block mode and increment commands.

Example

```
" Method 1: Visual Incrementing script
Ctrl+V          " select column in visual block
:I              " replace selection with incremental numbers
:I 5            " increment by 5 instead of 1
:II             " increment with left padding

" Method 2: Manual approach
qa              " start recording macro
I1<Esc>         " insert 1 at beginning
j               " move down
<C-a>           " increment number
q               " stop recording
```

```
@a              " execute macro to continue sequence
```

56.13 Handle common typos

Category: Text Manipulation
Tags: typos, abbreviations, correction, auto

Create abbreviations to automatically fix common typing mistakes.

Example

```
iab teh the
iab adn and
iab recieve receive
iab seperate separate
iab definately definitely
```

56.14 Increment/decrement numbers

Category: Text Manipulation
Tags: increment, decrement, numbers, math

Modify numbers in text using increment and decrement operations.

Example

```
Ctrl+a              " increment number under cursor
Ctrl+x              " decrement number under cursor
10<C-a>             " increment by 10
" In visual block mode:
g<C-a>              " increment each selected number progressively
g<C-x>              " decrement each selected number progressively
```

56.15 Insert column of text

Category: Text Manipulation
Tags: column, insert, visual, block

Use visual block mode (`Ctrl+V`), select column, press `I` to insert, type text, then `Esc` to apply to all lines.

Example

```
Ctrl+V  " start visual block
I       " insert at beginning of block
```

```
text    " type text to insert
Esc     " apply to all selected lines
```

56.16 Insert line numbers

Category: Text Manipulation
Tags: numbers, lines, automatic, sequence

Use `:put =range(1,10)` to insert numbers 1-10, or use visual block with `g<C-a>` to create sequences.

Example

```
:put =range(1,10)        " insert numbers 1 through 10
" Or select column with Ctrl+V, then:
g<C-a>                   " increment each line by 1 more than previous
```

56.17 Insert numbering

Category: Text Manipulation
Tags: numbering, sequence, insert, auto

Use `:put =range(1,10)` to insert numbers 1-10, or select lines and use `:s/^/\=line('.')-line("'<"` for relative numbering.

Example

```
:put =range(1,10)  " insert numbers 1-10
" Or for selected lines:
:'<,'>s/^/\=line('.')-line("'<")+1.'. '/
```

56.18 Join lines with custom separator

Category: Text Manipulation
Tags: join, separator, custom, lines

Use `:'<,'>s/\n/, /g` to join selected lines with custom separator (comma-space in example).

Example

```
:'<,'>s/\n/, /g     " join lines with ", "
:'<,'>s/\n/ | /g    " join lines with " | "
```

56.19 Lowercase/uppercase current line

Category: Text Manipulation
Tags: case, line, transform

Use `guu` to lowercase current line or `gUU` to uppercase current line.

Example

```
guu  " lowercase current line
gUU  " uppercase current line
```

56.20 Put text from register

Category: Text Manipulation
Tags: put, paste, register

Use `p` to put (paste) text after cursor and `P` to put text before cursor.

Example

```
p  " put text after cursor
P  " put text before cursor
"ap " put from register 'a' after cursor
```

56.21 ROT13 encoding

Category: Text Manipulation
Tags: rot13, encoding, cipher, transform

Apply ROT13 cipher to selected text using `g?` operator or external command.

Example

```
g??          " ROT13 current line
g?ap         " ROT13 around paragraph
:'<,'>!tr 'A-Za-z' 'N-ZA-Mn-za-m'  " ROT13 using external tr
```

56.22 Remove duplicate lines

Category: Text Manipulation
Tags: duplicate, unique, lines, clean

Use sort with unique flag or visual block operations to remove duplicate lines.

Example

```
:%!sort -u          " sort and remove duplicates
:sort u             " vim's built-in sort unique
" Or manually:
:g/^\(.*\)$\n\1$/d  " remove consecutive duplicates
```

56.23 Remove trailing whitespace

Category: Text Manipulation
Tags: whitespace, trailing, clean

Use `:%s/\s\+$//` to remove trailing whitespace from all lines.

Example

```
:%s/\s\+$//  " remove trailing whitespace
```

56.24 Replace mode operations

Category: Text Manipulation
Tags: replace, mode, overwrite

Use `R` to enter Replace mode where typed characters overwrite existing text. Use `r{char}` to replace single character.

Example

```
R    " enter Replace mode
ra   " replace character under cursor with 'a'
3rx  " replace 3 characters with 'x'
```

56.25 Reverse lines

Category: Text Manipulation
Tags: reverse, lines, order, flip

Use `:g/^/m0` to reverse all lines in buffer, or select lines and use `:'<,'>g/^/m '<-1` for selection.

Example

```
:g/^/m0             " reverse all lines
:'<,'>g/^/m'<-1     " reverse selected lines
```

56.26 Text alignment and padding

Category: Text Manipulation
Tags: align, pad, format, columns, spacing

Align text in columns and add padding for better formatting.

Example

```
" Align on specific character (e.g., =)
:'<,'>s/=/\t=/g         " add tab before =
:'<,'>!column -t -s $'\t'  " align columns

" Manual alignment
:%s/^/    /             " add 4 spaces to start of each line
:%s/$/    /             " add 4 spaces to end of each line
```

56.27 Text statistics

Category: Text Manipulation
Tags: statistics, analysis, count, metrics

Get detailed text statistics including character, word, and line counts.

Example

```
g<C-g>              " detailed stats for selection/buffer
:%s/word//gn        " count occurrences of 'word'
:%s/\w\+//gn        " count total words
:%s/.//gn           " count total characters
:%s/\n//gn          " count total lines
```

56.28 Transpose characters

Category: Text Manipulation
Tags: transpose, swap, characters, exchange

Swap adjacent characters or transpose text elements efficiently.

Example

```
xp              " transpose characters (delete and paste)

" For words: dawbP (delete word, back, paste)
daw             " delete a word
b               " go back one word
P               " paste before cursor
```

56.29 Undo and redo operations

Category: Text Manipulation
Tags: undo, redo, history

Use `u` to undo changes, `Ctrl+r` to redo undone changes, and `U` to undo all changes on current line.

Example

```
u      " undo last change
Ctrl+r " redo (undo the undo)
U      " undo all changes on current line
```

56.30 Unique line removal

Category: Text Manipulation
Tags: unique, duplicate, remove, lines

Remove duplicate lines while keeping unique entries using sort and uniq operations.

Example

```
:%!sort | uniq       " sort and remove duplicates (external)
:sort u              " sort and remove duplicates (internal)
:%!uniq              " remove consecutive duplicates only
```

56.31 Uppercase current word

Category: Text Manipulation
Tags: case, word, uppercase

Use `gUw` to uppercase current word.

Example

```
gUw  " uppercase current word
```

56.32 Word count methods

Category: Text Manipulation
Tags: count, words, statistics, analyze

Use `g Ctrl+g` for word count, or external commands for detailed statistics.

Example

```
g<C-g>              " show word count for buffer/selection
:w !wc              " count using external wc command
:%s/pattern//gn     " count pattern occurrences
" For live word count in statusline:
function! WordCount()
  return wordcount().words
endfunction
```

56.33 Work with CSV files

Category: Text Manipulation
Tags: csv, columns, data, tabular

Use CSV plugin commands to navigate and manipulate comma-separated data.

Example

```
" With CSV plugin:
:Csv 5              " highlight column 5
H, J, K, L          " navigate between cells
:Sort               " sort by column
:CC                 " copy column
:DC                 " delete column
" Convert to columns for viewing:
:%s/,/\t/g          " replace commas with tabs
```

CHAPTER 57

Text objects

57.1 Select around parentheses

Category: Text Objects
Tags: select, parentheses, around

Use `va(` to select text around parentheses (including the parentheses).

Example

```
va(  " select around parentheses
```

57.2 Select inside quotes

Category: Text Objects
Tags: select, quotes, inside

Use `vi"` to select text inside double quotes or `vi'` for single quotes.

Example

```
vi"  " select inside double quotes
vi'  " select inside single quotes
```

57.3 Text objects - HTML/XML tags

Category: Text Objects
Tags: textobject, html, xml, tags

Use `it` for inside HTML/XML tags and `at` for around tags including the tag markup.

Example

```
cit  " change inside HTML tag
dat  " delete around HTML tag
yit  " yank inside tag content
vat  " select around tag including markup
```

57.4 Text objects - alternative bracket notation

Category: Text Objects
Tags: textobject, brackets, alternatives

Use `ib` or `ab` as alternatives for `i(` or `a(`, and `iB` or `aB` as alternatives for `i{` or `a{`.

Example

```
cib  " change inside parentheses  - same as ci(, enters insert mode
dab  " delete around parentheses  - same as da(
yiB  " yank inside curly braces   - same as yi{
vib  " select inside parentheses  - same as vi(
vab  " select around parentheses  - sane as va(
vaB  " select around curly braces - same as va{
viB  " select inside curly braces - same as vi{
```

57.5 Text objects - angle brackets

Category: Text Objects
Tags: textobject, angle, brackets

Use `i<` and `a<` (or `i>` and `a>`) to operate inside/around angle brackets.

Example

```
ci<  " change inside angle brackets
da>  " delete around angle brackets
vi<  " select inside angle brackets
```

57.6 Text objects - around brackets

Category: Text Objects
Tags: textobject, brackets, around

Use `ca(`, `ca[`, `ca{` to change around parentheses, square brackets, or curly braces including the brackets.

Example

```
ca(  " change around parentheses
da[  " delete around square brackets
ya{  " yank around curly braces
```

57.7 Text objects - inside brackets

Category: Text Objects
Tags: textobject, brackets, inside

Use `ci(`, `ci[`, `ci{` to change inside parentheses, square brackets, or curly braces. Works with `d`, `y`, `v` too.

Example

```
ci(  " change inside parentheses
di[  " delete inside square brackets
yi{  " yank inside curly braces
```

57.8 Text objects - quoted strings

Category: Text Objects
Tags: textobject, quotes, strings

Use `i"` and `a"` for double-quoted strings, `i'` and `a'` for single-quoted strings, `i`` and `a`` for backtick strings.

Example

```
ci"  " change inside double quotes
da'  " delete around single quotes
yi`  " yank inside backticks
```

57.9 Text objects - sentences and paragraphs

Category: Text Objects
Tags: textobject, sentence, paragraph

Use `is`/`as` for inside/around sentence and `ip`/`ap` for inside/around paragraph.

Example

```
cis  " change inside sentence
das  " delete around sentence
vip  " select inside paragraph
yap  " yank around paragraph
```

57.10 Text objects - square brackets

Category: Text Objects
Tags: textobject, square, brackets

Use `i[` and `a[` (or `i]` and `a]`) to operate inside/around square brackets.

Example

```
ci[  " change inside square brackets
da]  " delete around square brackets
yi[  " yank inside square brackets
```

57.11 Text objects - word variations

Category: Text Objects
Tags: textobject, word, inner

Use `iw` for inside word, `aw` for around word (includes space), `iW` for inside WORD, `aW` for around WORD.

Example

```
ciw  " change inside word
daw  " delete around word (includes space)
yiW  " yank inside WORD (space-separated)
```

57.12 Text objects with operators

Category: Text Objects
Tags: textobject, operators, combinations

Text objects work with all operators: `c` (change), `d` (delete), `y` (yank), `v` (visual select), `=` (format), `>` (indent right), `<` (indent left).

Example

```
=ap  " format around paragraph
>i{  " indent inside curly braces
<as  " unindent around sentence
```

CHAPTER 58

Treesitter

58.1 Build a treesitter-based code outline

Category: Treesitter
Tags: treesitter, outline, navigation, structure

Create a code outline viewer using treesitter to extract document structure.

Example

```
local function get_code_outline(bufnr)
  bufnr = bufnr or 0
  local parser = vim.treesitter.get_parser(bufnr)
  local tree = parser:parse()[1]
  local root = tree:root()
  local lang = parser:lang()

  -- Different queries for different languages
  local queries = {
    lua = [[
      (function_declaration name: (identifier) @name) @definition
      (assignment_statement
        (variable_list name: (identifier) @name)
        (expression_list value: (function_definition))) @definition
    ]],
    python = [[
      (function_definition name: (identifier) @name) @definition
      (class_definition name: (identifier) @name) @definition
    ]],
    javascript = [[
      (function_declaration name: (identifier) @name) @definition
      (class_declaration name: (identifier) @name) @definition
      (method_definition name: (property_identifier) @name) @definition
    ]],
  }

  local query_str = queries[lang]
  if not query_str then return {} end

  local query = vim.treesitter.query.parse(lang, query_str)
  local outline = {}

  for id, node in query:iter_captures(root, bufnr, 0, -1) do
    if query.captures[id] == "name" then
```

```
      local name = vim.treesitter.get_node_text(node, bufnr)
      local row = node:start()
      local parent = node:parent()
      local kind = parent and parent:type() or "unknown"

      table.insert(outline, {
        name = name,
        line = row + 1,
        kind = kind,
      })
    end
  end

  return outline
end

-- Display outline
local function show_outline()
  local outline = get_code_outline()
  local lines = {}

  for _, item in ipairs(outline) do
    table.insert(lines, string.format("%4d: %s (%s)",
      item.line, item.name, item.kind))
  end

  -- Show in floating window or quickfix
  vim.fn.setqflist({}, "r", {
    title = "Code Outline",
    lines = lines,
  })
  vim.cmd("copen")
end

vim.api.nvim_create_user_command("Outline", show_outline, {})
```

58.2 Check if treesitter is available for filetype

Category: Treesitter
Tags: treesitter, check, available, parser

Check if treesitter parser is installed and available for current buffer.

Example

```
-- Check if parser exists for language
local function has_parser(lang)
  local ok = pcall(vim.treesitter.get_parser, 0, lang)
  return ok
end

-- Check current buffer
```

```
local function check_treesitter()
  local ft = vim.bo.filetype

  if has_parser(ft) then
    print("Treesitter available for " .. ft)

    -- Get parser info
    local parser = vim.treesitter.get_parser(0, ft)
    print("Language:", parser:lang())

    return true
  else
    print("No treesitter parser for " .. ft)
    print("Install with: :TSInstall " .. ft)

    return false
  end
end

vim.api.nvim_create_user_command("TSCheck", check_treesitter, {})
```

58.3 Create custom treesitter text objects

Category: Treesitter
Tags: treesitter, text-objects, custom, selection

Use treesitter queries to define custom text objects for inner/outer selections.

Example

```
-- Define custom text object for function body
local query = vim.treesitter.query.parse("lua", [[
  (function_declaration
    body: (block) @function.inner) @function.outer

  (function_definition
    body: (block) @function.inner) @function.outer
]])

-- Select function inner (body only)
local function select_function_inner()
  local node = vim.treesitter.get_node()

  -- Find parent function
  while node do
    if node:type() == "function_declaration" or
       node:type() == "function_definition" then
      -- Find block child
      for child in node:iter_children() do
        if child:type() == "block" then
          local start_row, start_col, end_row, end_col = child:range()
          -- Select range
```

```
          vim.api.nvim_win_set_cursor(0, {start_row + 1, start_col})
          vim.cmd("normal! v")
          vim.api.nvim_win_set_cursor(0, {end_row + 1, end_col - 1})
          return
        end
      end
    end
    node = node:parent()
  end
end

-- Map to 'if' (inner function)
vim.keymap.set({"o", "x"}, "if", select_function_inner,
  {desc = "Select inner function"})
```

58.4 Enhanced text objects with treesitter

Category: Treesitter
Tags: text-objects, selection, nvim-treesitter-textobjects, editing

Use nvim-treesitter-textobjects plugin to add treesitter-aware text objects for functions, classes, parameters, and more.

Example

```
-- Install with lazy.nvim:
{
  "nvim-treesitter/nvim-treesitter-textobjects",
  dependencies = "nvim-treesitter/nvim-treesitter",
  config = function()
    require('nvim-treesitter.configs').setup({
      textobjects = {
        select = {
          enable = true,
          lookahead = true,  -- Automatically jump forward to textobj
          keymaps = {
            ["af"] = "@function.outer",
            ["if"] = "@function.inner",
            ["ac"] = "@class.outer",
            ["ic"] = "@class.inner",
            ["aa"] = "@parameter.outer",
            ["ia"] = "@parameter.inner",
          },
        },
        move = {
          enable = true,
          set_jumps = true,  -- Add to jumplist
          goto_next_start = {
            ["]f"] = "@function.outer",
            ["]c"] = "@class.outer",
          },
          goto_previous_start = {
```

```
            ["[f"] = "@function.outer",
            ["[c"] = "@class.outer",
          },
        },
      },
    })
  end,
}

-- Usage:
-- vaf - select around function
-- dif - delete inside function
-- ]f - jump to next function
```

58.5 Find all nodes of specific type

Category: Treesitter
Tags: treesitter, queries, search, nodes

Use queries to find all nodes matching a specific type in the buffer.

Example

```
-- Find all function definitions in Lua
local function find_functions(bufnr)
  bufnr = bufnr or 0
  local parser = vim.treesitter.get_parser(bufnr, "lua")
  local tree = parser:parse()[1]
  local root = tree:root()

  local query = vim.treesitter.query.parse("lua", [[
    (function_declaration
      name: (identifier) @name) @function

    (assignment_statement
      (variable_list
        name: (identifier) @name)
      (expression_list
        value: (function_definition) @function))
  ]])

  local functions = {}
  for id, node in query:iter_captures(root, bufnr, 0, -1) do
    if query.captures[id] == "name" then
      local func_name = vim.treesitter.get_node_text(node, bufnr)
      local row, col = node:start()
      table.insert(functions, {
        name = func_name,
        line = row + 1,
        col = col
      })
    end
```

```
  end

  return functions
end

-- Usage
local funcs = find_functions()
for _, f in ipairs(funcs) do
  print(string.format("%s at line %d", f.name, f.line))
end
```

58.6 Get treesitter node under cursor

Category: Treesitter
Tags: treesitter, api, node, cursor

Use `vim.treesitter.get_node()` to get the syntax node under the cursor. Essential for building treesitter-based features.

Example

```
-- Get node under cursor
local node = vim.treesitter.get_node()

if node then
  print("Node type:", node:type())
  print("Node text:", vim.treesitter.get_node_text(node, 0))

  -- Get node range
  local start_row, start_col, end_row, end_col = node:range()
  print(string.format("Range: [%d,%d] to [%d,%d]",
    start_row, start_col, end_row, end_col))
end
```

58.7 Get treesitter parser for buffer

Category: Treesitter
Tags: treesitter, api, parser, buffer

Use `vim.treesitter.get_parser()` to access the treesitter parser and syntax tree for any buffer.

Example

```
-- Get parser for current buffer
local parser = vim.treesitter.get_parser(0, "lua")

-- Parse and get syntax tree
local trees = parser:parse()
```

```
local tree = trees[1]
local root = tree:root()

-- Get language
print("Language:", parser:lang())

-- Get tree information
local start_row, start_col, end_row, end_col = root:range()
print(string.format("Tree spans lines %d to %d", start_row, end_row))

-- Parser for specific language (with fallback)
local ok, py_parser = pcall(vim.treesitter.get_parser, 0, "python")
if ok then
  print("Python parser available")
end
```

58.8 Handle treesitter injection languages

Category: Treesitter
Tags: treesitter, injection, embedded, languages

Work with embedded languages like Lua in Vimscript, SQL in strings, or markdown code blocks.

Example

```
-- Get all parsers including injected languages
local parser = vim.treesitter.get_parser(0)

parser:for_each_tree(function(tree, language_tree)
  local lang = language_tree:lang()
  local root = tree:root()

  print("Found language:", lang)

  -- Process each language separately
  if lang == "lua" then
    -- Query Lua nodes
  elseif lang == "vim" then
    -- Query Vim nodes
  end
end)

-- Example: Find Lua code in Vim heredoc
local query = vim.treesitter.query.parse("vim", [[
  (let_statement
    (identifier)
    (heredoc
      (heredoc_body) @lua.code))
]])
```

58.9 Highlight custom patterns with treesitter

Category: Treesitter
Tags: treesitter, highlight, queries, custom

Use treesitter queries to add custom syntax highlighting beyond default parsers.

Example

```
-- Highlight TODO comments specially
local ns_id = vim.api.nvim_create_namespace("custom_hl")

local function highlight_todos()
  local parser = vim.treesitter.get_parser(0)
  local tree = parser:parse()[1]
  local root = tree:root()

  -- Query for comments
  local query = vim.treesitter.query.parse(parser:lang(), [[
    (comment) @comment
  ]])

  vim.api.nvim_buf_clear_namespace(0, ns_id, 0, -1)

  for id, node in query:iter_captures(root, 0, 0, -1) do
    local text = vim.treesitter.get_node_text(node, 0)
    if text:match("TODO") or text:match("FIXME") or text:match("HACK") then
      local start_row, start_col, end_row, end_col = node:range()
      vim.api.nvim_buf_set_extmark(0, ns_id, start_row, start_col, {
        end_row = end_row,
        end_col = end_col,
        hl_group = "WarningMsg",
        priority = 150,
      })
    end
  end
end

-- Run on buffer enter and changes
vim.api.nvim_create_autocmd({"BufEnter", "TextChanged", "TextChangedI"}, {
  callback = highlight_todos,
})
```

58.10 Navigate treesitter tree programmatically

Category: Treesitter
Tags: treesitter, api, tree, navigation

Use treesitter node methods to traverse the syntax tree and find related nodes.

Example

```
local node = vim.treesitter.get_node()

if node then
  -- Get parent node
  local parent = node:parent()

  -- Get next/previous sibling
  local next_sibling = node:next_sibling()
  local prev_sibling = node:prev_sibling()

  -- Get children
  for child in node:iter_children() do
    print("Child type:", child:type())
  end

  -- Get named children only (skip punctuation, etc.)
  for child in node:iter_children() do
    if child:named() then
      print("Named child:", child:type())
    end
  end

  -- Find parent of specific type
  while node do
    if node:type() == "function_declaration" then
      print("Found function!")
      break
    end
    node = node:parent()
  end
end
```

58.11 Treesitter folding

Category: Treesitter
Tags: treesitter, folding, code, structure

Set `foldmethod=expr` and `foldexpr=nvim_treesitter#foldexpr()` to use treesitter-based folding.

Example

```
:set foldmethod=expr
:set foldexpr=nvim_treesitter#foldexpr()
```

58.12 Treesitter incremental selection

Category: Treesitter
Tags: treesitter, selection, incremental, expand

Use `Ctrl-space` to start incremental selection, then repeat to expand selection based on syntax tree.

Example

```
Ctrl-space  " start/expand selection
Ctrl-x      " shrink selection (if configured)
```

58.13 Treesitter install parser

Category: Treesitter
Tags: treesitter, install, parser, language

Use `:TSInstall <language>` to install treesitter parser for a specific language.

Example

```
:TSInstall lua        " install Lua parser
:TSInstall javascript " install JavaScript parser
:TSInstall all        " install all maintained parsers
```

58.14 Treesitter node navigation

Category: Treesitter
Tags: treesitter, navigation, nodes, movement

Use `]f` and `[f` to navigate between function nodes, or `]c` and `[c` for class nodes (if configured).

Example

```
]f  " next function
[f  " previous function
]c  " next class
[c  " previous class
```

58.15 Treesitter playground

Category: Treesitter
Tags: treesitter, playground, debug, explore

Use `:TSPlaygroundToggle` to open treesitter playground for exploring syntax tree interactively.

Example

```
:TSPlaygroundToggle  " toggle treesitter playground
```

58.16 Treesitter query predicates and directives

Category: Treesitter
Tags: treesitter, queries, predicates, advanced

Use query predicates to add conditions to treesitter pattern matching.

Example

```
-- Query with predicates
local query = vim.treesitter.query.parse("lua", [[
  ; Find function calls named "require"
  (function_call
    name: (identifier) @func
    (#eq? @func "require"))

  ; Find strings longer than 10 characters
  (string
    content: (string_content) @str
    (#lua-match? @str "^.{10,}$"))

  ; Match only in comments
  (comment) @comment
    (#match? @comment "TODO")
]])

-- Available predicates:
-- #eq? - exact match
-- #match? - Lua pattern match
-- #lua-match? - Lua string.match
-- #contains? - contains substring
-- #any-of? - matches any of given values

-- Example: Find long variable names
local long_var_query = vim.treesitter.query.parse("lua", [[
  (identifier) @var
    (#lua-match? @var "^.{15,}$")
]])

local parser = vim.treesitter.get_parser(0, "lua")
local tree = parser:parse()[1]
local root = tree:root()

for id, node in long_var_query:iter_captures(root, 0, 0, -1) do
  local name = vim.treesitter.get_node_text(node, 0)
  local row = node:start()
  print(string.format("Long variable '%s' at line %d", name, row + 1))
end
```

58.17 Treesitter swap nodes

Category: Treesitter
Tags: treesitter, swap, parameters, arguments

Use treesitter to swap function parameters or other syntax nodes using configured keymaps.

Example

```
gs  " swap with next parameter/node
gS  " swap with previous parameter/node
```

58.18 Use treesitter for smart text editing

Category: Treesitter
Tags: treesitter, editing, smart, refactoring

Leverage treesitter for intelligent code refactoring and editing operations.

Example

```
-- Rename variable in current scope
local function rename_variable()
  local node = vim.treesitter.get_node()

  -- Find identifier under cursor
  while node and node:type() ~= "identifier" do
    node = node:parent()
  end

  if not node then
    print("Not on an identifier")
    return
  end

  local old_name = vim.treesitter.get_node_text(node, 0)

  -- Get new name from user
  local new_name = vim.fn.input("Rename to: ", old_name)
  if new_name == "" or new_name == old_name then return end

  -- Find scope (function or file)
  local scope = node
  while scope do
    if scope:type() == "function_declaration" or
       scope:type() == "function_definition" then
      break
    end
    scope = scope:parent()
  end

  if not scope then
```

```
    -- Use root as scope
    local parser = vim.treesitter.get_parser(0)
    scope = parser:parse()[1]:root()
  end

  -- Find all identifiers with same name in scope
  local query = vim.treesitter.query.parse("lua", [[
    (identifier) @id
  ]])

  local changes = {}
  for id, found_node in query:iter_captures(scope, 0) do
    if vim.treesitter.get_node_text(found_node, 0) == old_name then
      local start_row, start_col, end_row, end_col = found_node:range()
      table.insert(changes, {
        start_row = start_row,
        start_col = start_col,
        end_row = end_row,
        end_col = end_col,
      })
    end
  end

  -- Apply changes in reverse order to maintain positions
  table.sort(changes, function(a, b)
    return a.start_row > b.start_row or
           (a.start_row == b.start_row and a.start_col > b.start_col)
  end)

  for _, change in ipairs(changes) do
    vim.api.nvim_buf_set_text(0,
      change.start_row, change.start_col,
      change.end_row, change.end_col,
      {new_name})
  end

  print(string.format("Renamed %d occurrences", #changes))
end

vim.keymap.set("n", "<leader>rn", rename_variable,
  {desc = "Rename variable in scope"})
```

58.19 Write custom treesitter queries

Category: Treesitter
Tags: treesitter, queries, pattern-matching

Use treesitter query language to find patterns in code. Queries use S-expressions to match syntax nodes.

Example

```
-- Query to find all function calls
local query = vim.treesitter.query.parse("lua", [[
  (function_call
    name: (identifier) @function.name
    arguments: (arguments) @function.args)
]])

local parser = vim.treesitter.get_parser(0, "lua")
local tree = parser:parse()[1]
local root = tree:root()

-- Execute query
for id, node, metadata in query:iter_captures(root, 0, 0, -1) do
  local name = query.captures[id]
  local text = vim.treesitter.get_node_text(node, 0)
  print(name .. ":", text)
end
```

CHAPTER 59

Ui

59.1 Change highlight group on the fly

Category: UI
Tags: highlight, groups, fun

You can change the highlight group on the fly. For example, the next command changes all comments to red italic:

Example

```
:hi Comment guifg=#ffaa00 gui=italic
```

59.2 Check highlight groups

Category: UI
Tags: highlight, groups, colors

Use `:hi` to view all highlight groups and their current settings.

Example

```
:hi  " show all highlight groups
```

59.3 Custom statusline

Category: UI
Tags: statusline, custom, display

Use `vim.opt.statusline` to set a custom statusline format.

Example

```
:lua vim.opt.statusline = "%f %y %m %= %l:%c"
```

59.4 Flesh yanked text

Category: UI
Tags: highlight, group, yank, flash

Use the following command to flash yanked text using the `IncSearch` highlight for `200` milliseconds.

Example

```
:au TextYankPost *
↪  lua=vim.highlight.on_yank{higroup='IncSearch',timeout=200}
```

59.5 Highlight goroups

Category: UI
Tags: highlight, groups, fun

Use the following code to create command `HLList`.

Example

```
command! HLList lua local b=vim.api.nvim_create_buf(false,true)
↪  vim.api.nvim_set_current_buf(b) local
↪  g=vim.fn.getcompletion("","highlight")
↪  vim.api.nvim_buf_set_lines(b,0,-1,false,g) for i,n in ipairs(g) do
↪  pcall(vim.api.nvim_buf_add_highlight,b,-1,n,i-1,0,-1) end
```

When run, the command creates a scratch buffer with one line per highlight group, with each line styled with its own group.

59.6 Print treesitter highlight group info

Category: UI
Tags: highlight, group, treesitter

Use the following command to check the highlight info for the text under the cursor:

Example

```
:lua print(vim.treesitter.get_captures_at_cursor()[1] or "NONE")
```

CHAPTER 60

Vimscript fundamentals

60.1 Autocommand creation in script

Category: Vim Script
Tags: autocmd, autocommand, event, group, script

Create autocommands programmatically with proper grouping and cleanup.

Example

```
" Create autocommand group
:augroup MyGroup
  :autocmd!  " clear existing autocommands in group
  :autocmd BufRead *.py setlocal tabstop=4
  :autocmd BufWritePre * call TrimWhitespace()
:augroup END

" Function called by autocommand
function! TrimWhitespace()
  %s/\s\+$//e
endfunction
```

60.2 Basic variable assignment and types

Category: Vim Script
Tags: variable, let, type, assignment, scope

Use `:let` to assign variables with different scopes: `g:` (global), `l:` (local), `s:` (script), `a:` (argument), `v:` (vim).

Example

```
:let g:my_var = 42           " global variable
:let l:local_var = "hello"   " local variable
:let s:script_var = []       " script-local variable
:let b:buffer_var = {}       " buffer-local variable
:let w:window_var = 3.14     " window-local variable
```

60.3 Built-in function usage

Category: Vim Script
Tags: builtin, function, vim, utility, helper

Utilize Vim's extensive built-in function library for common tasks.

Example

```
" String functions
:echo strwidth("text")          " display width
:echo stridx("hello", "ll")     " find substring index
:echo repeat("*", 10)           " repeat string

" List functions
:echo max([1, 5, 3, 2])         " maximum value
:echo sort(['c', 'a', 'b'])     " sort list
:echo uniq(sort([1, 2, 2, 3]))  " unique sorted values

" File functions
:echo expand('%:p')             " full path of current file
:echo fnamemodify('file.txt', ':r')  " remove extension
:echo glob('*.vim')             " glob pattern matching
```

60.4 Conditional statements and logic

Category: Vim Script
Tags: if, else, condition, logic, comparison

Use `if`, `elseif`, `else`, `endif` for conditional logic with comparison operators.

Example

```
:let score = 85
:if score >= 90
  echo "Excellent"
:elseif score >= 70
  echo "Good"
:else
  echo "Needs improvement"
:endif

" Comparison operators: == != > < >= <= =~ !~
```

60.5 Debugging Vim scripts

Category: Vim Script
Tags: debug, echo, echom, verbose, profile

Debug Vim scripts using various techniques and built-in tools.

Example

```
" Basic debugging
:echo "Debug: variable = " . variable
:echom "Message saved to :messages"     " persistent message

" Conditional debugging
if exists('g:debug_mode') && g:debug_mode
  echom "Debug info: " . string(data)
endif

" Verbose execution
:verbose source script.vim           " show sourcing info
:set verbose=9                       " detailed execution info

" Profile script execution
:profile start profile.log
:profile func *                      " profile all functions
:source slow_script.vim
:profile pause
```

60.6 Error handling with try-catch

Category: Vim Script
Tags: try, catch, finally, error, exception

Handle errors gracefully using try-catch-finally blocks.

Example

```
:try
  " Potentially failing code
  let result = some_risky_function()
:catch /^Error:/
  " Handle specific error pattern
  echo "Caught error: " . v:exception
:catch /.*/
  " Handle any other error
  echo "Unknown error occurred"
:finally
  " Always execute this
  echo "Cleanup code"
:endtry
```

60.7 Event handling and callbacks

Category: Vim Script
Tags: event, callback, timer, async, handler

Handle events and create callbacks using timers and autocommands.

Example

```
" Timer callback
function! TimerCallback(timer_id)
  echo "Timer " . a:timer_id . " fired!"
endfunction

:let timer_id = timer_start(1000, 'TimerCallback')      " 1 second
:let repeat_timer = timer_start(500, 'TimerCallback', {'repeat': 5})

" Autocommand with function callback
function! OnBufEnter()
  echo "Entered buffer: " . bufname('%')
endfunction

:autocmd BufEnter * call OnBufEnter()
```

60.8 File and buffer operations

Category: Vim Script
Tags: file, buffer, read, write, operation

Read/write files and manipulate buffers programmatically.

Example

```
" File operations
:let lines = readfile('config.txt')    " read file to list
:call writefile(['line1', 'line2'], 'output.txt')  " write list to file

" Buffer operations
:let bufnr = bufnr('%')                " current buffer number
:let bufname = bufname(bufnr)          " buffer name
:let lines = getbufline(bufnr, 1, '$') " get all lines
:call setbufline(bufnr, 1, 'new first line')  " set line
```

60.9 Function definition and calling

Category: Vim Script
Tags: function, define, call, return, parameter

Define functions with `function` keyword, call with `:call` or directly in expressions.

Example

```
function! MyFunction(param1, param2)
  let result = a:param1 + a:param2
```

```
  return result
endfunction

:call MyFunction(5, 3)           " call function
:echo MyFunction(10, 20)         " use in expression
:let value = MyFunction(1, 2)    " assign result
```

60.10 List and dictionary operations

Category: Vim Script
Tags: list, dictionary, array, hash, data-structure

Work with lists `[]` and dictionaries `{}` using built-in functions and operators.

Example

```
" Lists
:let mylist = ['apple', 'banana', 'cherry']
:let mylist += ['date']          " add item
:let first = mylist[0]           " access by index
:call add(mylist, 'elderberry')  " append item
:call remove(mylist, 1)          " remove by index

" Dictionaries
:let mydict = {'name': 'vim', 'version': 8}
:let mydict.type = 'editor'      " add key
:let name = mydict['name']       " access value
:call remove(mydict, 'version')  " remove key
```

60.11 Loops and iteration

Category: Vim Script
Tags: for, while, loop, iteration, range

Use `for` loops for iteration over lists/ranges, `while` for conditional loops.

Example

```
" For loop over range
:for i in range(1, 5)
  echo i
:endfor

" For loop over list
:for item in ['a', 'b', 'c']
  echo item
:endfor

" While loop
```

```
:let i = 0
:while i < 3
  echo i
  let i += 1
:endwhile
```

60.12 Lua integration in Vim script

Category: Vim Script
Tags: lua, integration, execute, call, hybrid

Execute Lua code from Vim script and pass data between them.

Example

```
" Execute Lua code
:lua print("Hello from Lua")
:lua vim.opt.number = true

" Call Lua from Vim script
:let result = luaeval('2 + 3')
:let data = luaeval('vim.fn.getbufinfo()')

" Execute Lua file
:luafile ~/.config/nvim/lua/config.lua

" Vim script function called from Lua
function! VimFunction(arg)
  return "Vim received: " . a:arg
endfunction
```

60.13 Mappings in Vim script

Category: Vim Script
Tags: mapping, keymap, bind, shortcut, script

Create key mappings programmatically with different modes and options.

Example

```
" Normal mode mapping
:nnoremap <leader>w :write<CR>

" Insert mode mapping
:inoremap jk <Esc>

" Visual mode mapping
:vnoremap <leader>c "+y
```

```
" Mapping with script-local function
:nnoremap <leader>f :call <SID>MyFunction()<CR>

function! s:MyFunction()
  echo "Script-local function called"
endfunction
```

60.14 Option manipulation

Category: Vim Script
Tags: option, set, setlocal, global, buffer

Get and set Vim options programmatically using & syntax.

Example

```
" Get option value
:let current_ts = &tabstop
:let buf_ft = &l:filetype        " buffer-local option

" Set option value
:let &tabstop = 4
:let &l:number = 1               " buffer-local
:let &g:background = 'dark'      " global

" Toggle boolean option
:let &wrap = !&wrap
```

60.15 Regular expressions in Vim script

Category: Vim Script
Tags: regex, pattern, match, substitute, search

Use `=~` and `!~` operators for pattern matching, `substitute()` for replacements.

Example

```
:let text = "Hello World 123"
:if text =~ '\d\+'              " contains digits
  echo "Has numbers"
:endif

:let clean = substitute(text, '\d\+', 'XXX', 'g')  " replace digits
:let matches = matchlist(text, '\(\w\+\) \(\w\+\)')  " capture groups
:echo match(text, 'World')      " find position
```

60.16 Script sourcing and modules

Category: Vim Script
Tags: source, module, include, script, organization

Source other scripts and create modular code organization.

Example

```
" Source another script
:source ~/.config/nvim/helpers.vim
:runtime! plugin/**/*.vim        " source all plugins

" Script-local variables and functions
let s:script_var = "private"

function! s:ScriptFunction()
  return "Only accessible within this script"
endfunction

" Export public interface
function! MyModule#PublicFunction()
  return s:ScriptFunction()
endfunction
```

60.17 String formatting and printf

Category: Vim Script
Tags: string, format, printf, sprintf, output

Format strings using `printf()` and `string()` functions.

Example

```
:let name = "Neovim"
:let version = 0.8
:let message = printf("Welcome to %s v%.1f", name, version)

" Number formatting
:echo printf("%d", 42)           " integer
:echo printf("%04d", 7)          " zero-padded: 0007
:echo printf("%.2f", 3.14159)    " float: 3.14
:echo printf("%s", string([1,2,3]))  " convert to string
```

60.18 String operations and concatenation

Category: Vim Script
Tags: string, concatenation, operation, manipulation

Use . for string concatenation, `len()` for length, `split()` and `join()` for array operations.

Example

```
:let name = "Neo" . "vim"        " concatenation
:let length = len("hello")       " string length
:let words = split("a,b,c", ",") " split string
:let text = join(['a','b'], '-') " join array
:echo toupper("hello")           " HELLO
:echo tolower("WORLD")           " world
```

60.19 System command execution

Category: Vim Script
Tags: system, command, execute, shell, external

Execute system commands and capture output using `system()` and related functions.

Example

```
" Execute command and get output
:let output = system('ls -la')
:let files = split(system('find . -name "*.vim"'), '\n')

" Check command success
:let result = system('grep pattern file.txt')
:if v:shell_error == 0
  echo "Command succeeded"
:else
  echo "Command failed"
:endif
```

60.20 User command definition

Category: Vim Script
Tags: command, user-command, define, custom

Create custom user commands with parameters and completion.

Example

```
" Simple command
:command! Hello echo "Hello World"

" Command with arguments
:command! -nargs=1 Greet echo "Hello " . <args>

" Command with range
:command! -range LineCount echo <line2> - <line1> + 1
```

```
" Command with completion
:command! -nargs=1 -complete=file EditConfig edit ~/.config/<args>
```

CHAPTER 61
Visual mode

61.1 Repeating changes with gv and dot command

Category: Visual
Tags: visual, repeat, gv, dot, reselect, workflow

Use `gv` to reselect the last visual selection at a new location, then use `.` to repeat the same change. Perfect for applying identical modifications to different non-contiguous blocks.

Example

```
" Workflow example:
" 1. Select text block (V or Ctrl+v)
" 2. Make change (e.g., S) to wrap in parentheses)
" 3. Move cursor to different location
" 4. Use gv to reselect same-sized block
" 5. Use . to repeat the wrapping action
```

61.2 Reselect last visual selection

Category: Visual
Tags: visual, reselect, selection, repeat

Use `gv` to reselect the last visual selection area.

Example

```
gv  " reselect last visual selection
```

61.3 Visual block append

Category: Visual
Tags: visual, block, append, column

Use `Ctrl+v` to select visual block, then `A` to append text to end of each selected line.

Example

```
Ctrl+v  " select visual block
A       " append to end of all lines
text    " type text to append
Esc     " apply to all lines
```

61.4 Visual mode - Ex commands

Category: Visual
Tags: visual, ex, command, range

Press : in visual mode to run Ex commands on the selected range. The range '<,'> is automatically inserted.

Example

```
" After making visual selection:
:               " enters :'<,'> for range
:s/old/new/g    " substitute in selection
:sort           " sort selected lines
:w newfile.txt  " write selection to file
```

61.5 Visual mode - corner and edge movement

Category: Visual
Tags: visual, corner, block, movement

Use o to move cursor to opposite corner of selection, O to move to other corner in block mode.

Example

```
" In visual mode:
o   " move cursor to opposite corner of selection
O   " move to other corner (block mode only)
```

61.6 Visual mode - joining and substitution

Category: Visual
Tags: visual, join, substitute, replace

Use J to join selected lines with spaces, gJ to join without spaces, s to substitute selection, r to replace each character.

Example

```
" After making visual selection:
J    " join lines with spaces
gJ   " join lines without spaces
s    " substitute selected text
rx   " replace each character with 'x'
```

61.7 Visual mode - operators and transformations

Category: Visual
Tags: visual, operator, transform, case

Apply operators to visual selections: `c` (change), `d` (delete), `y` (yank), `~` (toggle case), `u` (lowercase), `U` (uppercase).

Example

```
" After making visual selection:
c   " change selected text
d   " delete selected text
y   " yank selected text
~   " toggle case of selection
u   " make selection lowercase
U   " make selection uppercase
```

61.8 Visual mode - paste and replace

Category: Visual
Tags: visual, paste, replace, register

Use `p` or `P` to replace visual selection with register contents. This is useful for swapping text.

Example

```
" Copy text first, then select other text:
p   " replace selection with register contents
P   " same as p in visual mode
```

61.9 Visual mode - tag and keyword operations

Category: Visual
Tags: visual, tag, keyword, jump

Use `Ctrl+]` to jump to tag of selected text, `K` to run keywordprg on selection.

Example

```
" After selecting text:
Ctrl+]  " jump to tag of selected text
K       " run help/man on selected keyword
```

61.10 Visual mode - toggle and change types

Category: Visual
Tags: visual, toggle, change, type

Use `v`, `V`, `Ctrl+v` in visual mode to change selection type or exit. Use `Ctrl+g` to toggle between Visual and Select mode.

Example

```
" In visual mode:
v       " change to character-wise or exit visual
V       " change to line-wise or exit visual
Ctrl+v  " change to block-wise or exit visual
Ctrl+g  " toggle Visual/Select mode
```

61.11 Visual selection modes

Category: Visual
Tags: visual, selection, mode

Use `v` for character-wise visual mode, `V` for line-wise visual mode, and `Ctrl+v` for block-wise visual mode.

Example

```
v      " character visual
V      " line visual
Ctrl+v " block visual
```

61.12 Yank and delete in visual mode

Category: Visual
Tags: yank, delete, visual

Use `y` to yank (copy) selected text and `d` to delete selected text in visual mode.

Example

```
y  " yank selected text
d  " delete selected text
```

61.13 Yank highlighting

Category: Visual
Tags: yank, highlight, autocmd

Create an autocmd to highlight yanked text briefly for visual feedback.

Example

```
:lua vim.api.nvim_create_autocmd("TextYankPost", {callback = function()
↪  vim.highlight.on_yank() end})
```

CHAPTER 62

Visual mode (advanced)

62.1 Incremental number sequences with g<C-a>

Category: Visual Mode Advanced
Tags: visual, block, numbers, increment, sequence

Use visual block mode with `g<C-a>` to create incremental number sequences. Perfect for creating numbered lists or incrementing multiple values.

Example

```
Ctrl+v          " start visual block mode
jjj             " select column of numbers (like 1,1,1,1)
g<Ctrl+a>       " increment sequentially (becomes 1,2,3,4)
```

62.2 Visual block append to varying line lengths

Category: Visual Mode Advanced
Tags: visual, block, append, variable, length

Use `$` in visual block mode to select to end of lines, then `A` to append despite varying lengths.

Example

```
Ctrl+v          " start visual block mode
jjj             " select multiple lines
$               " extend to end of longest line
A               " append to end of each line
text            " text to append
<Esc>           " apply to all lines
```

62.3 Visual block column editing

Category: Visual Mode Advanced
Tags: visual, block, column, edit, replace

Use visual block mode with c to change the same column range on multiple lines.

Example

```
Ctrl+v          " start visual block mode
jjj             " select lines vertically
lll             " extend selection horizontally
c               " change selected block
new_text        " replacement text
<Esc>           " apply to all lines
```

62.4 Visual block column insertion

Category: Visual Mode Advanced
Tags: visual, block, column, insert, prepend

Use visual block mode with I to insert text at the beginning of each selected line.

Example

```
Ctrl+v          " start visual block mode
jjj             " select multiple lines
I               " insert at beginning of each line
text            " type text to insert
<Esc>           " apply to all selected lines
```

62.5 Visual line operations

Category: Visual Mode Advanced
Tags: visual, line, operation, move, duplicate

Perform line-level operations on visual line selections.

Example

```
" After visual line selection (V):
J               " join selected lines with spaces
gJ              " join selected lines without spaces
:m +3           " move selected lines 3 positions down
:t .            " duplicate selected lines after current
```

62.6 Visual mode column operations

Category: Visual Mode Advanced
Tags: visual, column, arithmetic, calculation, block

Perform arithmetic operations on columns of numbers.

Example

```
Ctrl+v          " visual block mode
jjj             " select column of numbers
g Ctrl+a        " increment each number by 1, 2, 3...
g Ctrl+x        " decrement each number by 1, 2, 3...
```

62.7 Visual mode incremental selection

Category: Visual Mode Advanced
Tags: visual, incremental, selection, extend, treesitter

Use incremental selection for smart code selection (with treesitter).

Example

```
" In normal mode (with treesitter):
gnn             " start incremental selection
grn             " increment selection to next node
grm             " increment selection to scope
grc             " increment selection to clause
```

62.8 Visual mode indentation and alignment

Category: Visual Mode Advanced
Tags: visual, indent, align, format, block

Use visual mode for precise indentation and alignment operations.

Example

```
" After visual selection:
>               " indent selected text right
<               " indent selected text left
=               " auto-format selected text
gq              " format text to textwidth
```

62.9 Visual mode line manipulation

Category: Visual Mode Advanced
Tags: visual, line, manipulate, duplicate, move

Advanced line manipulation techniques in visual mode.

Example

```
" After visual line selection:
:co +5          " copy selected lines 5 lines down
:m -2           " move selected lines 2 lines up
:t $            " copy selected lines to end of file
:d              " delete selected lines
```

62.10 Visual mode macro application

Category: Visual Mode Advanced
Tags: visual, macro, apply, lines, batch

Apply macros to each line of a visual selection.

Example

```
" First record a macro (e.g., in register 'a')
qa              " start recording
... commands ...
q               " stop recording

" Then apply to visual selection:
V               " select lines
jjj             " extend selection
:normal @a      " apply macro 'a' to each line
```

62.11 Visual mode pattern matching

Category: Visual Mode Advanced
Tags: visual, pattern, select, match, extend

Select text based on patterns and extend selections intelligently.

Example

```
/pattern        " search for pattern
n               " go to next match
v               " start visual selection
n               " extend selection to next match
//e            " extend selection to end of current match
```

62.12 Visual mode rectangle operations

Category: Visual Mode Advanced
Tags: visual, rectangle, block, operation, column

Advanced rectangle/block operations for precise editing.

Example

```
Ctrl+v          " start visual block
G               " select to end of file (column)
I//             " insert // at beginning of each line
<Esc>           " apply to create comment block

Ctrl+v          " visual block
$               " to end of lines
A;              " append semicolon to each line
<Esc>           " apply to all lines
```

62.13 Visual mode register operations

Category: Visual Mode Advanced
Tags: visual, register, yank, paste, specific

Work with specific registers in visual mode.

Example

```
" After visual selection:
"ay             " yank selection to register 'a'
"Ay             " append selection to register 'a'
"+y             " yank to system clipboard
"*y             " yank to X11 selection
"0p             " paste from yank register (over selection)
```

62.14 Visual mode search and replace

Category: Visual Mode Advanced
Tags: visual, search, replace, substitute, scope

Perform search and replace operations within visual selections.

Example

```
" After visual selection:
:s/old/new/g    " replace in selection
:s/\%Vold/new/g " replace only within selection bounds
:g/pattern/d    " delete lines matching pattern in selection
:v/pattern/d    " delete lines NOT matching pattern
```

62.15 Visual mode smart selection

Category: Visual Mode Advanced
Tags: visual, smart, selection, expand, contract

Smart selection expansion and contraction techniques.

Example

```
" In visual mode:
aw              " expand to select a word
ap              " expand to select a paragraph
a(              " expand to select around parentheses
i{              " contract to select inside braces
```

62.16 Visual mode sorting and filtering

Category: Visual Mode Advanced
Tags: visual, sort, filter, lines, unique

Apply sorting and filtering operations to visual selections.

Example

```
" After visual selection:
:sort           " sort selected lines alphabetically
:sort n         " sort selected lines numerically
:sort u         " sort and remove duplicates
:sort!          " sort in reverse order
```

62.17 Visual mode text transformation

Category: Visual Mode Advanced
Tags: visual, transform, text, case, format

Apply various text transformations to visual selections.

Example

```
" After visual selection:
u               " convert to lowercase
U               " convert to uppercase
~               " toggle case
g?              " ROT13 encoding
Ctrl+a          " increment numbers in selection
Ctrl+x          " decrement numbers in selection
```

62.18 Visual mode text wrapping

Category: Visual Mode Advanced
Tags: visual, wrap, text, format, width

Control text wrapping and formatting in visual selections.

Example

```
" After visual selection:
gw              " wrap lines to textwidth
gwap            " wrap around paragraph
gqq             " format current line
gqap            " format around paragraph
```

62.19 Visual mode with external filters

Category: Visual Mode Advanced
Tags: visual, filter, external, command, process

Filter visual selections through external commands.

Example

```
" After visual selection:
!sort           " sort through external sort command
!uniq           " remove duplicates with uniq
!wc -l          " replace selection with line count
!column -t      " format as table with column command
```

62.20 Visual mode with folds

Category: Visual Mode Advanced
Tags: visual, fold, unfold, selection, code

Work with code folds in visual mode.

Example

```
" After visual selection:
zf              " create fold from selection
:fold           " create fold from selected lines
zo              " open fold under cursor
zc              " close fold under cursor
```

62.21 Visual mode with global commands

Category: Visual Mode Advanced
Tags: visual, global, command, pattern, execute

Execute global commands on visual selections.

Example

```
" After visual selection:
:g/pattern/normal @a    " run macro 'a' on matching lines
:g/TODO/s/old/new/g     " replace in TODO lines only
:v/important/d          " delete lines not containing 'important'
```

62.22 Visual mode with jumps and changes

Category: Visual Mode Advanced
Tags: visual, jump, change, navigate, selection

Create visual selections based on jump and change lists.

Example

```
Ctrl+o          " go to previous jump position
v               " start visual selection
Ctrl+i          " extend selection to next jump position
g;              " go to previous change
V               " line visual from previous change
g,              " extend to next change
```

62.23 Visual mode with marks

Category: Visual Mode Advanced
Tags: visual, mark, position, range, selection

Use marks to create precise visual selections.

Example

```
ma              " set mark 'a'
... move cursor ...
v'a             " visual select from current to mark 'a'
V`a             " visual line select from current to mark 'a'
```

62.24 Visual mode word selection shortcuts

Category: Visual Mode Advanced
Tags: visual, word, select, expand, quick

Quick methods to visually select words and expand selections.

Example

```
viw             " select inner word
vaw             " select a word (with spaces)
viW             " select inner WORD (space-separated)
vaW             " select a WORD
gv              " reselect last visual selection
```

62.25 Visual selection with text objects

Category: Visual Mode Advanced
Tags: visual, text, object, combine, selection

Combine visual mode with text objects for precise selections.

Example

```
vaw             " visually select a word
vap             " visually select a paragraph
vi(             " visually select inside parentheses
va{             " visually select around braces
vit             " visually select inside HTML/XML tags
```

CHAPTER 63

Window management

63.1 Advanced window operations

Category: Window Management
Tags: window, equalize, rotate, maximize, advanced

Use `Ctrl+w =` to equalize windows, `Ctrl+w r` to rotate windows, `Ctrl+w |` to maximize horizontally.

Example

```
Ctrl+w =    " equalize window sizes
Ctrl+w r    " rotate windows clockwise
Ctrl+w R    " rotate windows counter-clockwise
Ctrl+w |    " maximize current window horizontally
Ctrl+w _    " maximize current window vertically
```

63.2 Better gm command

Category: Window Management
Tags: cursor, middle, navigation, movement

Improved gm command that moves cursor to the middle of the physical line, ignoring whitespace.

Example

```
function! s:Gm()
  execute 'normal! ^'
  let first_col = virtcol('.')
  execute 'normal! g_'
  let last_col  = virtcol('.')
  execute 'normal! ' . (first_col + last_col) / 2 . '|'
endfunction
nnoremap <silent> gm :call <SID>Gm()<CR>
onoremap <silent> gm :call <SID>Gm()<CR>
```

63.3 Change cursor shape in modes

Category: Window Management
Tags: cursor, shape, mode, visual

Configure different cursor shapes for different modes to provide visual feedback.

Example

```
set guicursor=n-v-c:block,i-ci-ve:ver25,r-cr:hor20,o:hor50
" Block in normal, vertical bar in insert, horizontal in replace
```

63.4 Close all other windows

Category: Window Management
Tags: window, close, only, single

Use `:only` or `:on` to close all windows except the current one, making it take up the full screen.

Example

```
:only   " close all other windows (keep current)
:on     " short form of :only
Ctrl+w o " normal mode shortcut for :only
```

63.5 Diff mode for file comparison

Category: Window Management
Tags: diff, compare, vimdiff, merge

Use Vim's diff mode to compare and merge files effectively.

Example

```
:vimdiff file1 file2    " start vimdiff from command line
:diffthis               " make current window part of diff
:diffoff                " turn off diff mode
]c                      " next difference
[c                      " previous difference
do                      " diff obtain (get change from other)
dp                      " diff put (put change to other)
:diffget                " get changes from other buffer
:diffput                " put changes to other buffer
```

63.6 Fast window resizing

Category: Window Management
Tags: resize, window, keys, mapping

Use mapped keys for fast window resizing without complex key combinations.

Example

```
" Map + and - for easy window resizing
if bufwinnr(1)
  map + <C-W>+
  map - <C-W>-
endif
```

63.7 Focus mode for writing

Category: Window Management
Tags: focus, writing, distraction, zen

Create a distraction-free environment for writing and focused editing.

Example

```
" Simple focus mode
:set laststatus=0     " hide statusline
:set nonumber         " hide line numbers
:set norelativenumber " hide relative numbers
:set signcolumn=no    " hide sign column

" Toggle function
function! ToggleFocusMode()
  if &laststatus == 2
    set laststatus=0 nonumber norelativenumber signcolumn=no
  else
    set laststatus=2 number relativenumber signcolumn=yes
  endif
endfunction
nnoremap <F12> :call ToggleFocusMode()<CR>
```

63.8 Keep cursor centered

Category: Window Management
Tags: cursor, center, scroll, display

Keep cursor centered vertically on screen for better visibility while editing.

Example

```
" Keep cursor centered when scrolling
nnoremap <C-d> <C-d>zz
nnoremap <C-u> <C-u>zz
nnoremap n nzz
nnoremap N Nzz

" Or automatic centering
set scrolloff=999
```

63.9 Keep window when closing buffer

Category: Window Management
Tags: buffer, close, window, preserve

Use `:bp|bd #` to close buffer without closing the window layout.

Example

```
:bp|bd #        " go to previous buffer, delete current
:enew|bd #      " create new buffer, delete previous
```

63.10 Move window to tab

Category: Window Management
Tags: window, tab, move

Use `Ctrl+w T` to move current window to a new tab page.

Example

```
Ctrl+w T  " move current window to new tab
```

63.11 Move windows

Category: Window Management
Tags: window, move, position

Use `Ctrl+w H/J/K/L` to move current window to far left/bottom/top/right.

Example

```
Ctrl+w H  " move window to far left
Ctrl+w J  " move window to bottom
Ctrl+w K  " move window to top
Ctrl+w L  " move window to far right
```

63.12 Quick file explorer

Category: Window Management
Tags: explorer, netrw, files, browse

Use built-in file explorer (netrw) for quick file navigation and management.

Example

```
:Explore            " open file explorer in current window
:Sexplore           " open file explorer in horizontal split
:Vexplore           " open file explorer in vertical split
:Texplore           " open file explorer in new tab
:e.                 " edit current directory

" In netrw:
" <Enter> - open file/directory
" - - go up one directory
" D - delete file
" R - rename file
" % - create new file
```

63.13 Resize windows incrementally

Category: Window Management
Tags: window, resize, increment

Use `Ctrl+w +` to increase height, `Ctrl+w -` to decrease height, `Ctrl+w >` to increase width, `Ctrl+w <` to decrease width.

Example

```
Ctrl+w +  " increase window height
Ctrl+w -  " decrease window height
Ctrl+w >  " increase window width
Ctrl+w <  " decrease window width
```

63.14 Smart window navigation keymaps

Category: Window Management
Tags: navigation, keymap, window, split, lua

Create intuitive window navigation keymaps using Ctrl+hjkl for seamless movement between splits.

Example

```
-- Simple window navigation with Ctrl+hjkl:
vim.keymap.set('n', '<C-h>', '<C-w>h', { desc = 'Move to left window' })
vim.keymap.set('n', '<C-j>', '<C-w>j', { desc = 'Move to window below' })
vim.keymap.set('n', '<C-k>', '<C-w>k', { desc = 'Move to window above' })
vim.keymap.set('n', '<C-l>', '<C-w>l', { desc = 'Move to right window' })

-- Window resizing with arrow keys:
vim.keymap.set('n', '<C-Up>', ':resize +2<CR>', { desc = 'Increase window
↪  height' })
vim.keymap.set('n', '<C-Down>', ':resize -2<CR>', { desc = 'Decrease window
↪  height' })
vim.keymap.set('n', '<C-Left>', ':vertical resize -2<CR>', { desc =
↪  'Decrease window width' })
vim.keymap.set('n', '<C-Right>', ':vertical resize +2<CR>', { desc =
↪  'Increase window width' })

-- Quick window management:
vim.keymap.set('n', '<leader>wv', '<C-w>v', { desc = 'Split window
↪  vertically' })
vim.keymap.set('n', '<leader>ws', '<C-w>s', { desc = 'Split window
↪  horizontally' })
vim.keymap.set('n', '<leader>wq', '<C-w>q', { desc = 'Close current window'
↪  })
vim.keymap.set('n', '<leader>wo', '<C-w>o', { desc = 'Close other windows'
↪  })
```

63.15 Special window commands

Category: Window Management
Tags: window, special, file, tag

Use `Ctrl+w f` to split and open file under cursor, `Ctrl+w ]` to split and jump to tag, `Ctrl+w x` to exchange windows.

Example

```
Ctrl+w f  " split and open file under cursor
Ctrl+w ]  " split and jump to tag
Ctrl+w x  " exchange current window with another
```

63.16 Tab management

Category: Window Management
Tags: tabs, navigation, workspace, organize

Use tabs as workspaces to organize different projects or contexts.

Example

```
:tabnew          " create new tab
:tabclose        " close current tab
:tabonly         " close all other tabs
:tabn            " next tab
:tabp            " previous tab
gt               " next tab (normal mode)
gT               " previous tab (normal mode)
:tab split       " open current buffer in new tab
```

63.17 Window closing

Category: Window Management
Tags: window, close, quit

Use `Ctrl+w c` to close current window, `Ctrl+w o` to close all windows except current, `Ctrl+w q` to quit current window.

Example

```
Ctrl+w c  " close current window
Ctrl+w o  " close all other windows
Ctrl+w q  " quit current window
```

63.18 Window commands from Ex mode

Category: Window Management
Tags: wincmd, window, command, ex, mode

Use `:wincmd {key}` to execute window commands from Ex mode, useful in scripts and mappings.

Example

```
:wincmd j     " same as Ctrl+w j (move to window below)
:wincmd =     " same as Ctrl+w = (equalize windows)
:wincmd o     " same as Ctrl+w o (close other windows)
:wincmd v     " same as Ctrl+w v (vertical split)
:wincmd s     " same as Ctrl+w s (horizontal split)
```

63.19 Window navigation basics

Category: Window Management
Tags: window, navigation, movement

Use `Ctrl+w h/j/k/l` to move to left/down/up/right windows, `Ctrl+w w` to cycle through

windows, `Ctrl+w p` for previous window.

Example

```
Ctrl+w h  " move to left window
Ctrl+w j  " move to window below
Ctrl+w k  " move to window above
Ctrl+w l  " move to right window
Ctrl+w w  " cycle to next window
Ctrl+w p  " go to previous window
```

63.20 Window navigation without prefix

Category: Window Management
Tags: navigation, window, mapping, efficient

Map window navigation to single keys for faster movement between splits.

Example

```
" Map Alt+hjkl for window navigation
nnoremap <A-h> <C-w>h
nnoremap <A-j> <C-w>j
nnoremap <A-k> <C-w>k
nnoremap <A-l> <C-w>l

" Or use leader key combinations
nnoremap <leader>h <C-w>h
nnoremap <leader>j <C-w>j
nnoremap <leader>k <C-w>k
nnoremap <leader>l <C-w>l
```

63.21 Window position navigation

Category: Window Management
Tags: window, position, navigation

Use `Ctrl+w t` to go to top window and `Ctrl+w b` to go to bottom window.

Example

```
Ctrl+w t  " go to top window
Ctrl+w b  " go to bottom window
```

63.22 Window splitting strategies

Category: Window Management
Tags: split, window, layout, organize

Create and organize window splits for efficient multi-file editing.

Example

```
:split          " horizontal split
:vsplit         " vertical split
:new            " new horizontal split with empty buffer
:vnew           " new vertical split with empty buffer
:sp filename    " split and open specific file
:vsp filename   " vertical split and open specific file
```

CHAPTER 64

Workflow patterns

64.1 Add lines to multiple files with cfdo

Category: Workflow
Tags: cfdo, append, multiple, files, quickfix

Use `:cfdo` with `append()` function to add lines at specific positions across multiple files in the quickfix list.

Example

```
" Add line after line 4 in all quickfix files
:cfdo call append(4, '"status": "not started"') | update

" Add multiple lines with execute and normal
:cfdo execute 'norm 5GO"status": "not started",' | update
```

64.2 Backup and recovery workflow

Category: Workflow Patterns
Tags: backup, recovery, safety, workflow, protection

Systematic backup and recovery workflow patterns for data protection.

Example

```
" Automatic backup workflow
function! CreateBackup()
  let backup_dir = expand('~/.local/share/nvim/backups/')
  let timestamp = strftime('%Y%m%d_%H%M%S')
  let backup_file = backup_dir . expand('%:t') . '.' . timestamp

  if !isdirectory(backup_dir)
    call mkdir(backup_dir, 'p')
  endif

  execute 'write ' . backup_file
  echo "Backup saved: " . backup_file
endfunction
```

```
:command! Backup call CreateBackup()

" Recovery workflow
function! ShowBackups()
  let backup_dir = expand('~/.local/share/nvim/backups/')
  let current_file = expand('%:t')
  execute 'edit ' . backup_dir
  execute '/' . current_file
endfunction

:command! Recovery call ShowBackups()

" Auto-backup on save
:autocmd BufWritePre *.{py,js,lua,vim} call CreateBackup()
```

64.3 Build and deployment workflow

Category: Workflow Patterns
Tags: build, deployment, release, workflow, automation

Integrated build and deployment workflow patterns.

Example

```
" Build workflow shortcuts
:nnoremap <leader>bb :!make build<CR>
:nnoremap <leader>bt :!make test<CR>
:nnoremap <leader>bc :!make clean<CR>
:nnoremap <leader>br :!make run<CR>

" Deployment workflow
function! DeploymentChecklist()
  :put ='## Deployment Checklist'
  :put ='- [ ] Tests pass'
  :put ='- [ ] Version bumped'
  :put ='- [ ] Changelog updated'
  :put ='- [ ] Documentation updated'
  :put ='- [ ] Backup created'
endfunction

" Release workflow
:nnoremap <leader>rv :!git tag v<C-R>=input('Version: ')<CR><CR>
:nnoremap <leader>rp :!git push --tags<CR>
```

64.4 Code quality and standards workflow

Category: Workflow Patterns
Tags: quality, standards, lint, format, workflow

Systematic code quality and standards enforcement workflow patterns.

Example

```
" Code quality checks
:nnoremap <leader>ql :!eslint %<CR>        " JavaScript linting
:nnoremap <leader>qf :!prettier --write %<CR>  " Format file
:nnoremap <leader>qp :!pylint %<CR>       " Python linting
:nnoremap <leader>qr :!rubocop %<CR>      " Ruby linting

" Pre-commit quality workflow
function! PreCommitChecks()
  :!eslint .
  :!prettier --check .
  :!npm test
  echo "Pre-commit checks completed"
endfunction

:command! PreCommit call PreCommitChecks()

" Quality metrics tracking
function! QualityReport()
  :put ='# Code Quality Report - ' . strftime('%Y-%m-%d')
  :put ='## Lint Issues:'
  :r !eslint . --format compact | wc -l
  :put ='## Test Coverage:'
  :r !npm run coverage | tail -1
endfunction
```

64.5 Code review and annotation workflow

Category: Workflow Patterns
Tags: review, annotation, comment, feedback, collaboration

Systematic code review and annotation patterns for collaboration.

Example

```
" Code review annotations
:nnoremap <leader>rc A  // REVIEW:
:nnoremap <leader>rt A  // TODO:
:nnoremap <leader>rf A  // FIXME:
:nnoremap <leader>rq A  // QUESTION:

" Extract review comments
:vimgrep /\/\/ REVIEW:/j **/*.py
:vimgrep /\/\/ TODO:/j **/*.py

" Review checklist workflow
function! ReviewChecklist()
  :tabnew REVIEW.md
  :put ='## Code Review Checklist'
  :put ='- [ ] Logic correctness'
  :put ='- [ ] Performance considerations'
  :put ='- [ ] Error handling'
```

```
  :put ='- [ ] Test coverage'
endfunction
```

64.6 Code review and collaboration workflow

Category: Workflow Patterns
Tags: collaboration, review, feedback, team, workflow

Systematic collaboration and code review workflow patterns.

Example

```
" Collaboration markers
:nnoremap <leader>cr A  // CR:
:nnoremap <leader>ca A  // APPROVED:
:nnoremap <leader>cq A  // QUESTION:
:nnoremap <leader>cs A  // SUGGESTION:

" Review workflow
function! StartReview(branch)
  execute '!git checkout ' . a:branch
  :tabnew REVIEW_NOTES.md
  :put ='# Code Review: ' . a:branch
  :put ='## Files Changed:'
  :r !git diff --name-only HEAD~1
  :put =''
  :put ='## Comments:'
endfunction

:command! -nargs=1 Review call StartReview(<args>)
```

64.7 Configuration management workflow

Category: Workflow Patterns
Tags: configuration, dotfiles, settings, management, sync

Systematic configuration and dotfile management patterns.

Example

```
" Configuration editing shortcuts
:nnoremap <leader>ev :edit $MYVIMRC<CR>
:nnoremap <leader>sv :source $MYVIMRC<CR>
:nnoremap <leader>ep :edit ~/.zshrc<CR>
:nnoremap <leader>et :edit ~/.tmux.conf<CR>

" Configuration backup workflow
function! BackupConfig()
  :!cp $MYVIMRC ~/.config/backup/init.vim.$(date +%Y%m%d)
```

```
  echo "Configuration backed up"
endfunction

" Dotfile synchronization
:nnoremap <leader>ds :!cd ~/dotfiles && git add . && git commit -m "Update
 ↪  config" && git push<CR>
```

64.8 Documentation workflow

Category: Workflow Patterns
Tags: documentation, writing, markdown, workflow, notes

Efficient documentation and note-taking workflow patterns.

Example

```
" Documentation templates
function! InsertDocTemplate()
  :put ='# ' . expand('%:t:r')
  :put =''
  :put ='## Overview'
  :put =''
  :put ='## Usage'
  :put =''
  :put ='## Examples'
  :put =''
  :put ='## API'
endfunction

" Quick note-taking
:nnoremap <leader>nd :edit
 ↪  ~/notes/daily/<C-R>≥strftime('%Y-%m-%d')<CR>.md<CR>
:nnoremap <leader>np :edit ~/notes/projects/<C-R>≥expand('%:h:t')<CR>.md<CR>

" Link insertion for markdown
:nnoremap <leader>ml i[]()<Left><Left><Left>
```

64.9 Error handling and debugging patterns

Category: Workflow Patterns
Tags: error, debugging, troubleshoot, workflow, pattern

Systematic error handling and debugging workflow patterns.

Example

```
" Debug logging workflow
function! AddDebugLogging()
  let line_num = line('.')
```

```
  let debug_line = 'console.log("DEBUG line ' . line_num . ':", );'
  call append(line('.'), '    ' . debug_line)
  normal! j$F:
  startinsert!
endfunction

:nnoremap <leader>dl :call AddDebugLogging()<CR>

" Error investigation workflow
function! InvestigateError()
  :tabnew ERROR_INVESTIGATION.md
  :put ='# Error Investigation - ' . strftime('%Y-%m-%d')
  :put ='## Problem:'
  :put ='## Steps to Reproduce:'
  :put ='## Expected vs Actual:'
  :put ='## Investigation:'
  :put ='## Solution:'
endfunction

:nnoremap <leader>ei :call InvestigateError()<CR>
```

64.10 Focus and distraction management

Category: Workflow Patterns
Tags: focus, distraction, productivity, workflow, zen

Patterns for maintaining focus and minimizing distractions during coding.

Example

```
" Focus mode - minimize distractions
function! FocusMode()
  :set nonumber
  :set norelativenumber
  :set signcolumn=no
  :set laststatus=0
  :set noshowcmd
  :set noshowmode
  :set colorcolumn=
  echo "Focus mode enabled"
endfunction

:command! Focus call FocusMode()

" Pomodoro integration
function! StartPomodoro(minutes)
  echo "Starting " . a:minutes . " minute focus session"
  execute "!timer " . (a:minutes * 60) . " &"
endfunction

:command! -nargs=? Pomodoro call StartPomodoro(<args> ? <args> : 25)
```

64.11 Git workflow integration

Category: Workflow Patterns
Tags: git, workflow, version, control, branch, merge

Comprehensive Git workflow patterns integrated with editing.

Example

```
" Git workflow commands
:nnoremap <leader>gs :!git status<CR>
:nnoremap <leader>ga :!git add %<CR>
:nnoremap <leader>gc :!git commit -v<CR>
:nnoremap <leader>gd :!git diff %<CR>
:nnoremap <leader>gl :!git log --oneline -10<CR>

" Branch workflow
:nnoremap <leader>gb :!git checkout -b feature/
:nnoremap <leader>gm :!git checkout main && git pull<CR>

" Commit message templates
function! CommitTemplate(type)
  :tabnew
  :put =a:type . ': '
  :put =''
  :put ='## What'
  :put =''
  :put ='## Why'
  :startinsert!
endfunction

:command! -nargs=1 Commit call CommitTemplate(<args>)
```

64.12 Knowledge management workflow

Category: Workflow Patterns
Tags: knowledge, documentation, notes, learning, workflow

Systematic knowledge capture and documentation workflow patterns.

Example

```
" Knowledge base organization
function! CreateNote(category, title)
  let note_path = '~/knowledge/' . a:category . '/' . a:title . '.md'
  execute 'edit ' . note_path
  :put ='# ' . substitute(a:title, '_', ' ', 'g')
  :put ='Created: ' . strftime('%Y-%m-%d')
  :put ='Tags: '
  :put =''
  :put ='## Summary'
  :put =''
```

```
  :put ='## Details'
  :put =''
  :put ='## Examples'
  :put =''
  :put ='## References'
endfunction

:command! -nargs=* Note call CreateNote(<f-args>)

" Quick research capture
:nnoremap <leader>nq :put ='**Q: ' . input('Question: ') . '**'<CR>
:nnoremap <leader>na :put ='**A: ' . input('Answer: ') . '**'<CR>
```

64.13 Learning and experimentation workflow

Category: Workflow Patterns
Tags: learning, experiment, playground, practice, workflow

Structured learning and experimentation workflow patterns.

Example

```
" Learning workspace setup
function! SetupPlayground(language)
  let playground_dir = '~/playground/' . a:language
  execute 'edit' playground_dir . '/experiment.py'

  " Insert learning template
  :put ='# Learning ' . a:language . ' - ' . strftime('%Y-%m-%d')
  :put ='# Goal: '
  :put =''
  :put ='# Experiment:'
  :put =''
  :put ='# Notes:'
  :put =''
  :put ='# Resources:'
endfunction

:command! -nargs=1 Playground call SetupPlayground(<args>)

" Quick snippet testing
:nnoremap <leader>lt :tabnew /tmp/test.<C-R≥input('Extension: ')<CR><CR>
```

64.14 Multi-file editing workflow

Category: Workflow Patterns
Tags: multi-file, editing, buffer, window, navigation

Efficient patterns for working with multiple files simultaneously.

Example

```
" Quick file switching workflow
nnoremap <leader>b :buffers<CR>:buffer<Space>
nnoremap <leader>f :find<Space>
nnoremap <Tab> :bnext<CR>
nnoremap <S-Tab> :bprevious<CR>

" Split and tab workflow
nnoremap <leader>v :vsplit<CR>:find<Space>
nnoremap <leader>s :split<CR>:find<Space>
nnoremap <leader>t :tabnew<CR>:find<Space>

" Quick jump between related files
nnoremap <leader>a :find %:r.*<CR>  " find files with same basename
```

64.15 Performance profiling workflow

Category: Workflow Patterns
Tags: performance, profiling, optimization, workflow, analysis

Systematic performance analysis and optimization workflow patterns.

Example

```
" Performance profiling setup
function! StartProfiling()
  :put ='# Performance Analysis - ' . strftime('%Y-%m-%d')
  :put ='## Baseline Measurements:'
  :put ='## Bottleneck Areas:'
  :put ='## Optimization Plan:'
  :put ='## Results:'
endfunction

" Performance markers
:nnoremap <leader>ps A  # PERF_START
:nnoremap <leader>pe A  # PERF_END
:nnoremap <leader>pt A  # TODO: Optimize this section

" Benchmark workflow
:nnoremap <leader>pb :!time python %<CR>
:nnoremap <leader>pm :!python -m cProfile %<CR>
```

64.16 Project workspace initialization

Category: Workflow Patterns
Tags: workspace, project, initialize, setup, session

Establish consistent project workspace patterns for efficient development.

Example

```
-- Auto-detect project root and setup workspace
local function setup_project_workspace()
  local root_patterns = {'.git', 'package.json', 'Cargo.toml', 'go.mod',
  ↪  'requirements.txt'}
  local root = vim.fs.dirname(vim.fs.find(root_patterns, {upward =
  ↪  true})[1])

  if root then
    vim.cmd('cd ' .. root)
    -- Load project-specific settings
    local project_config = root .. '/.nvimrc'
    if vim.fn.filereadable(project_config) == 1 then
      vim.cmd('source ' .. project_config)
    end
  end
end

vim.api.nvim_create_autocmd('VimEnter', { callback = setup_project_workspace
↪  })
```

64.17 Refactoring workflow patterns

Category: Workflow Patterns
Tags: refactoring, code, restructure, improve, workflow

Systematic code refactoring and improvement workflow patterns.

Example

```
" Extract function refactoring
function! ExtractFunction() range
  let function_name = input('Function name: ')
  if !empty(function_name)
    '<,'>delete
    put ='def ' . function_name . '():'
    put ='    # extracted code here'
    put =''
    normal! O    return result
  endif
endfunction

" Rename variable workflow
:nnoremap <leader>rr :%s/\<<C-r><C-w>\>//g<Left><Left>
:nnoremap <leader>rf :bufdo %s/\<<C-r><C-w>\>//ge |
↪  update<Left><Left><Left><Left><Left><Left><Left><Left><Left><Left><Left>

" Refactoring checklist
function! RefactorChecklist()
  :put ='## Refactoring Checklist'
  :put ='- [ ] Tests still pass'
  :put ='- [ ] No functionality changes'
```

```
    :put ='- [ ] Code is cleaner/more readable'
    :put ='- [ ] Performance maintained'
  endfunction
```

64.18 Resolve merge conflicts with git jump

Category: Workflow
Tags: git, merge, conflict, quickfix, resolution

Use `git jump merge` to populate quickfix list with all merge conflicts for quick navigation and resolution.

Example

```
" First, install git-jump (part of git-contrib):
" On macOS: brew install git-extras
" Or copy from git source: contrib/git-jump/

" Add to shell config:
" export PATH="$PATH:/path/to/git/contrib/git-jump"

" In Neovim, run git jump and load conflicts:
:cexpr system('git jump merge')
:copen

" Or create a command in init.lua:
vim.api.nvim_create_user_command('GitJumpMerge', function()
  vim.fn.setqflist({}, 'r', {
    title = 'Git Merge Conflicts',
    lines = vim.fn.systemlist('git jump merge')
  })
  vim.cmd('copen')
end, {})

" Usage: :GitJumpMerge
" Navigate with :cnext, :cprev
```

64.19 Search and replace workflow

Category: Workflow Patterns
Tags: search, replace, refactor, pattern, workflow

Systematic approach to search and replace operations across projects.

Example

```
" Progressive search and replace workflow
" 1. Search and review
:vimgrep /old_function/j **/*.py
```

```
:copen    " creates quickfix list based on search results
:cnext    " jump to next match
:cprev    " jump to previous match

" 2. Confirm matches visually
:cfdo %s/old_function/new_function/gc

" 3. Save all changed files
:cfdo update

" Macro for complex replacements
:let @r = 'ciwnew_name<Esc>n'  " record replacement macro
:argdo normal @r               " apply to all files in arglist
:argdo update                  " save all changes
```

64.20 Session and workspace persistence

Category: Workflow Patterns
Tags: session, workspace, persistence, restore, save

Patterns for saving and restoring work sessions and workspace state.

Example

```
" Automatic session management
function! SaveSession()
  let session_dir = '~/.config/nvim/sessions/'
  let session_name = substitute(getcwd(), '/', '_', 'g')
  execute 'mksession! ' . session_dir . session_name . '.vim'
endfunction

function! LoadSession()
  let session_dir = '~/.config/nvim/sessions/'
  let session_name = substitute(getcwd(), '/', '_', 'g')
  let session_file = session_dir . session_name . '.vim'

  if filereadable(expand(session_file))
    execute 'source ' . session_file
  endif
endfunction

" Auto-save session on exit
:autocmd VimLeave * call SaveSession()
:autocmd VimEnter * call LoadSession()
```

64.21 Testing and debugging workflow

Category: Workflow Patterns
Tags: testing, debugging, workflow, development, tdd

Integrated testing and debugging workflow patterns.

Example

```
" Test-driven development workflow
:nnoremap <leader>tt :!npm test %<CR>      " test current file
:nnoremap <leader>ta :!npm test<CR>        " test all
:nnoremap <leader>tw :!npm test -- --watch<CR>  " watch mode

" Debugging workflow
:nnoremap <leader>db Oprint(f"DEBUG: {}")<Left><Left>
:nnoremap <leader>dl :g/print.*DEBUG/d    " remove debug lines

" Error navigation workflow
:nnoremap <leader>en :cnext<CR>
:nnoremap <leader>ep :cprevious<CR>
:nnoremap <leader>el :clist<CR>
```