Neovim Tips & Tricks

A collection of useful tips and tricks for Neovim

Sasa Markovic
smalltux@yahoo.com

December 8, 2025

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

Table of Contents

Introduction LXXIII

1 Advanced functions 1

1.1 Advanced CSV Column Navigation and Manipulation 1
1.2 Advanced PHP Function Documentation Viewer 2
1.3 Automatic Java Import Generation 3
1.4 Custom Function to Open File in Visual Studio 5
1.5 Dynamic Footnote Insertion with Vim Function 6
1.6 Dynamically Display Current Function Name 7
1.7 Extract Text from HTML Files 9
1.8 Flexible Compilation Function 10
1.9 Flexible Heading Underline Function 11
1.10 Flexible Underline Command 12
1.11 Flexible Underline Function 12
1.12 Generate Java Bean Getters and Setters 13
1.13 Generate Web Forum Compatible HTML Snippet 14
1.14 Move Characters to Specific Column Range 15
1.15 Screen-Aware Command Execution 16
1.16 Smart Preprocessor Directive Commenter 16
1.17 Text Extraction from HTML/URL 18

2 Advanced mappings 21

2.1 Abbreviations vs mappings 21
2.2 Access Python Documentation in Vim 21
2.3 Add Blank Line Without Moving Cursor 22
2.4 Advanced CapsLock Key Behavior 22
2.5 Advanced Command and Function Mapping Techniques 23
2.6 Advanced HTML Tag Completion 24
2.7 Advanced Pasted Text Selection with Preserve Mode 25
2.8 Advanced Single Character Insertion with Repeat 25

2.9 Advanced Word Swapping Mapping ... 26
2.10 Align #endif with Corresponding #if Directive ... 26
2.11 Alternative Abbreviation Expansion Method ... 27
2.12 Alternative Cursor Position Preservation Mapping ... 27
2.13 Alternative Escape Key Mappings ... 28
2.14 Auto-Close Brackets & Navigate Out ... 28
2.15 Auto-Close Brackets and Braces ... 29
2.16 Auto-Close Quotes in HTML/XML Attributes ... 29
2.17 Auto-Generate Function Headers Quickly ... 30
2.18 Auto-Insert Closing Characters with Smart Behavior ... 30
2.19 Auto-Insert Function Headers ... 31
2.20 Auto-close Brackets and Braces in Insert Mode ... 31
2.21 Auto-continue Python module omnicompletion ... 32
2.22 Auto-pair mappings ... 33
2.23 Automate Java Getter/Setter Generation ... 34
2.24 Automatic Closing Characters for Brackets and Quotes ... 34
2.25 Automatic Closing Characters with Smart Insertion ... 35
2.26 Automatic Code Block Commenting ... 35
2.27 Automatic Comment Mapping for Different Filetypes ... 37
2.28 Automatically Create and Update Cscope Database ... 37
2.29 Avoid Mapping Frequently Used Navigation Keys ... 38
2.30 Break Undo Chain with Modifier Keys ... 39
2.31 Buffer-local keymaps with Lua ... 39
2.32 Camel/Snake Case Conversion Functions ... 40
2.33 Capitalize Words Without Moving Cursor ... 41
2.34 Clipboard Keymappings for macOS ... 41
2.35 Command-line mappings ... 42
2.36 Compile File with F7 Keybinding ... 42
2.37 Conditional Abbreviation Expansion with Prompt ... 43
2.38 Conditional Closing Brace Insertion ... 44
2.39 Conditional Closing Character Insertion ... 44
2.40 Conditional Date Abbreviation Expansion ... 45
2.41 Conditional Key Remapping with AutoHotkey ... 46
2.42 Conditional Pair Mapping for Closing Characters ... 46
2.43 Configure Shift-Tab Key Behavior in Vim ... 47
2.44 Context-Specific Key Remapping with AutoHotkey ... 48
2.45 Context-aware mappings ... 48
2.46 Contextual F1 Help with Smart Behavior ... 49
2.47 Convenient Macro Playback Mapping ... 50
2.48 Convert Quoted Printable Email to Plain Text ... 50
2.49 Copy and Replace Words or Lines Repeatedly ... 51

2.50 Create Aligned Comment Boxes Quickly 51
2.51 Create Context-Aware Mappings for GUI and Console 52
2.52 Create Custom Keymaps for Language Mapping 52
2.53 Create Custom Language Keymaps 53
2.54 Create Custom Option Toggle Function 53
2.55 Create Custom Text Objects in Vim/Neovim 54
2.56 Create Granular Undo Breakpoints in Insert Mode 55
2.57 Create Mode-Specific Key Mappings 55
2.58 Create Mode-Specific Key Mappings in Vim 56
2.59 Create Recursive Vim Mappings for Repetitive Editing 56
2.60 Create Unique Plugin Mappings with <Plug> 57
2.61 Create a Function to Toggle Options 58
2.62 Cross-Platform Clipboard Key Mappings 58
2.63 Ctrl-S: Smart File Saving Shortcut 59
2.64 Custom Alt Key Mappings for Special Characters 60
2.65 Custom AutoCAD MTEXT Editing Menu in Vim 60
2.66 Custom Backspace Function for Edge Cases 61
2.67 Custom Folding Key Mappings 62
2.68 Custom Jump List Navigation Function 62
2.69 Custom Jump List Selection Function 63
2.70 Custom Search Text Object for Older Vim 64
2.71 Custom Toolbar URL Launcher 65
2.72 Cycle Numeric Options Intelligently 65
2.73 Debug Code Abbreviations for PHP 66
2.74 Diagnose and Clear Problematic Key Mappings 66
2.75 Diagnose and Fix Backspace Key Mappings 67
2.76 Disable Accidentally Triggered F1 Help Key 67
2.77 Disable Alt Key Menu Shortcuts in Vim 68
2.78 Disable Middle Mouse Button Paste 68
2.79 Disable Unwanted Built-in Vim Commands 69
2.80 Disable Unwanted Vim Key Mappings 69
2.81 Dynamic Commenting Operator for Different Filetypes 70
2.82 Dynamic File Preview in Index/Listing 71
2.83 Dynamic Key Handler with Multiple Contexts 71
2.84 Dynamic Search Mode Cycling 72
2.85 Easily Capitalize Words and Regions 74
2.86 Easily Move Lines Up or Down 74
2.87 Easy Block Selection with Alt+Left Mouse 75
2.88 Easy French Character Input Mappings 76
2.89 Easy French Character Mappings 76
2.90 Enhanced Block Selection with Matchit 77

2.91 Enhanced Mouse Interaction in Vim ... 77
2.92 Execute External Commands with Key Mapping ... 78
2.93 Execute Files and URLs from Buffer Text ... 79
2.94 Execute Files and URLs from Buffer Text in Windows ... 80
2.95 Execute Normal Mode Commands from Insert Mode ... 80
2.96 Exit Insert Mode Without Moving Cursor ... 81
2.97 Expand Existing Abbreviation Quickly ... 81
2.98 Explicit Undo Chain Break with <End> ... 82
2.99 Expression mappings ... 82
2.100 Fast Keycode Mapping in Terminal Vim ... 83
2.101 Find Unused Keys for Custom Mappings ... 83
2.102 Fix Meta-Keys in Terminal Vim ... 84
2.103 Fix x Command in Virtual Edit Mode ... 85
2.104 Flexible Caps Lock Mapping with AutoHotkey ... 85
2.105 Flexible Case Conversion Commands ... 86
2.106 Flexible Insert Mode Command Execution ... 86
2.107 Flexible Mappings for GUI and Console Vim ... 87
2.108 Generate Code Templates with Vim ... 88
2.109 Generate PHP Accessor and Setter Methods ... 88
2.110 Handle Numeric Keypad Escape Sequences ... 89
2.111 Handle Shift-Tab for Unindent and Completion ... 90
2.112 Improve Completion Popup Menu Navigation ... 90
2.113 Insert Empty Line in Normal Mode ... 91
2.114 Insert Mode Alt Key Mappings for Quick Editing ... 92
2.115 Insert Multiple Blank Lines Easily ... 92
2.116 Insert Space Without Moving Cursor ... 94
2.117 Insert-Mode Only Caps Lock Toggle ... 94
2.118 Invert Number Row Keys for Faster Typing ... 95
2.119 Invoke Function with Count Prefix ... 96
2.120 Launch and View Remote Batch Job Results ... 97
2.121 Macro Playback with Function Keys ... 98
2.122 Manual XML Formatting Mapping ... 98
2.123 Map Ctrl-Backspace to Delete Previous Word ... 98
2.124 Map Semicolon to Command Mode ... 99
2.125 Mapping special characters ... 99
2.126 Mapping with arguments ... 100
2.127 Move Function Parameters with Macros ... 101
2.128 Move Lines Up/Down with Alt Keys ... 101
2.129 Multi-Language Comment/Uncomment Mappings ... 102
2.130 Multiple key mappings ... 103
2.131 Numeric Keypad Escape Sequence Mapping ... 103

2.132 One-Key Header/Source File Toggle 104
2.133 Open Online Documentation for Word Under Cursor 104
2.134 Open Web Links Under Cursor 106
2.135 Open Windows Help Files with Custom Keymapping 106
2.136 Open Windows Help Files with Vim Keymaps 107
2.137 Operator-pending mappings 108
2.138 Postfix Abbreviations for Quick Code Snippets 108
2.139 Preserve Original Semicolon Functionality 109
2.140 Prevent Accidental Command Typos 109
2.141 Prevent Accidental Key Presses 110
2.142 Prevent Accidental Text Deletion in Insert Mode 110
2.143 Quick Abbreviation Expansion Mapping 111
2.144 Quick Braces Insertion in Insert Mode 111
2.145 Quick C Debugging Print Statement Macro 111
2.146 Quick Comment/Uncomment for Web Languages 112
2.147 Quick Compile Mapping with F7 113
2.148 Quick Dictionary Lookup in Vim 114
2.149 Quick Directory Change Mapping 114
2.150 Quick Escape Key Alternatives in Insert Mode 115
2.151 Quick Escape from Insert Mode 116
2.152 Quick File Save Mapping 116
2.153 Quick File Upload with FTPeel Integration 116
2.154 Quick Global Substitution Mapping 117
2.155 Quick HTML Anchor Creation from Two Lines 118
2.156 Quick HTML/URL Preview in Browser 118
2.157 Quick Insert Character at End of Line 119
2.158 Quick Insert Mode Commands via Double Letters 120
2.159 Quick Internet Search for Word Under Cursor 120
2.160 Quick Java Compilation and Execution Mappings 121
2.161 Quick MSDN Documentation Lookup 122
2.162 Quick Macro Playback with Q 122
2.163 Quick Markdown Preview with External Tools 123
2.164 Quick Number Conversion Under Cursor 123
2.165 Quick Omni and User Completion 124
2.166 Quick One-Key Session Command Mapping 125
2.167 Quick PHP Function Inline Help 126
2.168 Quick Python Documentation in Vim 126
2.169 Quick Single Character Insertion 127
2.170 Quick Single Word Insertion Mapping 128
2.171 Quick Spellcheck Mappings 128
2.172 Quick Syntax Highlighting Reset Mapping 129

2.173 Quick System Clipboard Keymaps 129
2.174 Quick Text Replacement Workflow 130
2.175 Quick Toggle for Vim Options 131
2.176 Quick Uuencode/Decode Text in Vim 132
2.177 Quick Web Search from Vim 132
2.178 Quick Word 'Stamping' Across Document 133
2.179 Quick Word Replacement Mapping 133
2.180 Quick Word Replacement Without Typing 134
2.181 Quick Word Replacement with Last Yanked Text 134
2.182 Quick Word Search Across Files 135
2.183 Quick Word Search Across Project 135
2.184 Quick Word/Line Change with Auto-Exit 136
2.185 Quick Word/Selection Replacement Mappings 136
2.186 Quick XML Element Creation Mapping 137
2.187 Quickfix-Powered Code Review Workflow 137
2.188 Quickly Comment/Uncomment Multiple Lines 138
2.189 Quickly Insert Non-Commented Lines 140
2.190 Quickly Insert Single Character 140
2.191 Quickly Introduce and Extract Variables 141
2.192 Quickly Launch and Edit Remote Batch Job Results 141
2.193 Quickly Replace Words Without Typing 142
2.194 Recursive abbreviations 143
2.195 Remap Caps Lock to Escape on macOS 144
2.196 Remap CapsLock to Escape or Ctrl 144
2.197 Remap CapsLock to Escape or Ctrl in Neovim 144
2.198 Remap Extra Keys on Non-US Keyboards 145
2.199 Remap Help Navigation for AZERTY Keyboards 146
2.200 Repeat Last Command at Start of Change 146
2.201 Repeat Last Edit Without Moving Cursor 146
2.202 Repeatable Single Character Insertion 147
2.203 Restore Cursor Position After Repeat Command 148
2.204 Run Macro on Matching Lines 148
2.205 Run Python Scripts from Vim 149
2.206 Run Windows Program with K Keymap 149
2.207 Run and Preview Interpreter Output in Split Window 149
2.208 SQL Command History Management in Vim 151
2.209 Safely List and Manage Vim Key Mappings 151
2.210 Script-local mappings 152
2.211 Silent and no-remap mappings 153
2.212 Simple RCS Version Control Mappings 153
2.213 Simulate Shift-Arrow Text Selection in Terminals 154

2.214 Smart Bracket and Parenthesis Insertion for Perl ... 154
2.215 Smart Tab Insertion with Context ... 155
2.216 Smart Table Editing with Tab Navigation ... 156
2.217 Smart Table Navigation with Custom Field Movement ... 156
2.218 Smart Toggle Line Comment ... 158
2.219 Swap ; and : for Faster Command Entry ... 159
2.220 Terminal mode mappings ... 159
2.221 Toggle Clojure Comments Easily ... 160
2.222 Toggle Less-like Scrolling Mode ... 161
2.223 Toggle Line Length Highlighting ... 162
2.224 Toggle Long Line Highlighting ... 163
2.225 Toggle Option Flags Dynamically ... 163
2.226 Toggle Whitespace Ignore in Diff ... 164
2.227 Toggle Word Completion with F12 Key ... 165
2.228 TortoiseSVN Commands from Vim ... 166
2.229 Troubleshoot Backspace Key Mappings ... 167
2.230 Turkish Keyboard Layout Optimization ... 168
2.231 Use <Leader> for Plugin-Friendly Key Mappings ... 168
2.232 Use Ctrl-O for Single Normal Mode Command in Insert Mode ... 169
2.233 Use Ctrl-[as Escape Alternative ... 169
2.234 Use Symbolic Key Notation for Mappings ... 170
2.235 Useful Vim Mappings for Productivity ... 170
2.236 Virtual Cursor Movement in Insert Mode ... 171
2.237 Visible Screen ROT13 Mapping with Cursor Preservation ... 171
2.238 Visual Mode Smart Paste Without Register Overwrite ... 172
2.239 Visual Selection Case Conversion ... 172
2.240 Visual mode mappings ... 173
2.241 Windows-style Word Navigation Shortcuts ... 173
2.242 Write Current Key Mappings to File ... 174

3 Advanced neovim ... 177

3.1 Advanced PHP Documentation Lookup Function ... 177
3.2 Advanced Syntax Group Identification Command ... 178
3.3 Automatic Desktop-Specific Vim Instances ... 178
3.4 Buffer-local variables with vim.b ... 179
3.5 Command preview and substitution ... 179
3.6 Custom Keyword Completion with Wildcards ... 180
3.7 Custom completion sources ... 180
3.8 Deep inspection with vim.inspect ... 181

3.9 Event loop and scheduling ... 182
3.10 Extmarks for persistent highlighting ... 182
3.11 Filetype detection API ... 182
3.12 Git Merge Conflict Resolution with Vimdiff ... 183
3.13 Global variables with vim.g ... 183
3.14 Health check system ... 184
3.15 Highlight group API ... 184
3.16 Keymap API with descriptions ... 184
3.17 Lua heredoc syntax ... 185
3.18 Lua require and module system ... 185
3.19 Namespace management ... 185
3.20 Option management with vim.opt ... 186
3.21 Quick Number Value Inspection ... 186
3.22 RPC and job control (vim.system) ... 187
3.23 Ring buffer for undo history ... 187
3.24 Runtime path manipulation ... 188
3.25 Secure mode and restrictions ... 188
3.26 Smart Function Signature Preview ... 188
3.27 Snippet expansion API ... 190
3.28 Tab-local variables with vim.t ... 190
3.29 Treesitter API access ... 191
3.30 UI events and hooks ... 191
3.31 Use Lists for Handling Newlines in Vim Scripts ... 191
3.32 User commands with Lua ... 192
3.33 Virtual text annotations ... 192
3.34 Window configuration API ... 193
3.35 Window-local variables with vim.w ... 193

4 Advanced options ... 195
4.1 Automatic session restoration ... 195
4.2 Automatic text wrapping ... 195
4.3 Backup and swap file locations ... 196
4.4 Clipboard integration ... 196
4.5 Complete options configuration ... 197
4.6 Cursor line and column ... 197
4.7 Diff options configuration ... 197
4.8 Fold column display ... 198
4.9 Incremental command preview ... 198
4.10 Line break at word boundaries ... 199
4.11 Mouse support in terminal ... 199

4.12 Persistent undo across sessions ... 200
4.13 Scroll context lines ... 200
4.14 Search highlighting timeout ... 200
4.15 Show invisible characters ... 201
4.16 Show line numbers relatively ... 201
4.17 Smart case searching ... 202
4.18 Spell checking configuration ... 202
4.19 Virtual editing mode ... 203
4.20 Wildmenu enhanced completion ... 203

5 Advanced search patterns ... 205
5.1 Anchors and word boundaries ... 205
5.2 Atom and group matching ... 205
5.3 Branch and alternation ... 205
5.4 Case sensitivity control ... 206
5.5 Change ”Last, First” to ”First Last” ... 206
5.6 Character classes in search ... 206
5.7 Column and line position matching ... 207
5.8 Composing complex patterns ... 207
5.9 Lookahead and lookbehind patterns ... 207
5.10 Mark position matching ... 208
5.11 Multiline pattern matching ... 208
5.12 Non-greedy matching ... 208
5.13 Pattern modifiers and flags ... 209
5.14 Quantifiers in search patterns ... 209
5.15 Recursive patterns ... 209
5.16 Search and replace using custom lua functions ... 210
5.17 Search and replace with expressions ... 210
5.18 Search context and ranges ... 210
5.19 Search history and repetition ... 211
5.20 Search in specific file types ... 211
5.21 Search with confirmation ... 211
5.22 Special characters and escaping ... 211
5.23 Very magic mode shortcuts ... 212
5.24 Virtual column matching ... 212
5.25 Zero-width assertions ... 212

6 Advanced text manipulation ... 215
6.1 Advanced register chaining and manipulation ... 215
6.2 Advanced text objects for precise selections ... 215

6.3 Expression register for calculations ... 216
6.4 Zero-width assertions in search patterns ... 217

7 Advanced tricks ... 219
7.1 AES256 File Encryption with GPG ... 219
7.2 Auto Highlight Current Word When Idle ... 219
7.3 Decode MIME Base64 and Quoted-Printable Encoding ... 221
7.4 Enhanced Hex Mode Editing in Vim ... 221
7.5 Preview HTML Text in Vim Buffer ... 223

8 Autocommands ... 225
8.1 Advanced HTML Tag Completion Function ... 225
8.2 Always Start on First Line of Git Commit ... 226
8.3 Append SQL Commands to History File ... 226
8.4 Auto Open Quickfix Window After Make ... 227
8.5 Auto Save Files When Focus Is Lost ... 227
8.6 Auto-Close Completion Preview Automatically ... 228
8.7 Auto-Close Quickfix Window When Last Buffer ... 228
8.8 Auto-Close Vim When Quickfix is Last Window ... 229
8.9 Auto-Generate Java Class Template ... 230
8.10 Auto-Insert Image Sizes in HTML ... 230
8.11 Auto-Insert Java Package Name ... 231
8.12 Auto-Position Cursor for Email Editing ... 232
8.13 Auto-Save and Highlight Unsaved Buffers on Focus Lost ... 233
8.14 Auto-Update Copyright Notice in Files ... 234
8.15 Auto-Update ctags on File Write ... 235
8.16 Auto-backup important files ... 236
8.17 Auto-chmod executable scripts ... 237
8.18 Auto-compile on save ... 237
8.19 Auto-format code on save ... 238
8.20 Auto-reload changed files ... 239
8.21 Auto-resize windows on terminal resize ... 240
8.22 Auto-save Buffer Periodically ... 240
8.23 Auto-toggle relative numbers ... 241
8.24 Autocompile JavaScript with Closure Compiler ... 242
8.25 Autocompile LaTeX on Write ... 243
8.26 Autoconvert Spaces to Tabs in Makefiles ... 243
8.27 Automate File Explorer Startup and Cleanup ... 244
8.28 Automated PHP Syntax Checking on Write ... 245
8.29 Automatic Backup on File Write ... 246

8.30 Automatic Binary File Hex Mode Detection ... 247
8.31 Automatic C/C++ Header Include Guards ... 248
8.32 Automatic Filetype Detection for ClearCase Files ... 249
8.33 Automatic Git Project Session Management ... 250
8.34 Automatic Header File Template Generation ... 251
8.35 Automatic Hex Mode for Binary Files ... 252
8.36 Automatic JavaScript Compilation with Error Handling ... 253
8.37 Automatic Last Modified Timestamp Update ... 254
8.38 Automatic Perforce File Checkout ... 255
8.39 Automatic Tags Generation for Python Projects ... 256
8.40 Automatic View and Fold Persistence ... 257
8.41 Automatic XML Formatting ... 258
8.42 Automatically Check for External File Changes ... 258
8.43 Automatically Decompile Java Class Files ... 259
8.44 Automatically Open Swap Files Read-Only ... 260
8.45 Automatically Refresh Firefox When Saving HTML/CSS ... 260
8.46 Automatically Set Compiler by Filetype ... 261
8.47 Automatically Set Listing Files as Read-Only ... 262
8.48 Automatically Sort and Deduplicate Quickfix List ... 262
8.49 Automatically Update Last Modified Timestamp ... 264
8.50 Change Color Scheme for Different Directories ... 264
8.51 Change directory to current file with autocommand ... 265
8.52 Check File Changes on Focus ... 265
8.53 Colorscheme Cycling Autocommand ... 266
8.54 Colorscheme for Focused Vim Instance ... 266
8.55 Conditional File Encoding Auto-Command ... 267
8.56 Conditional Fold Handling for File Types ... 268
8.57 Context-Specific Menus for Different Filetypes ... 268
8.58 Create directory on save ... 269
8.59 Cursor Line Only in Active Window ... 270
8.60 Debug Window Autocommands Interactively ... 270
8.61 Detect New Window Creation Reliably ... 271
8.62 Detect New Window Creation with Window-Local Variable ... 272
8.63 Detect Perl Batch Files in Vim ... 272
8.64 Detect and Mark Buffer as Modified When File Deleted ... 273
8.65 Dynamic Color Scheme Based on Buffer/Directory ... 274
8.66 Dynamic Color Schemes Based on Context ... 274
8.67 Dynamic Folding with Indent and Manual Override ... 275
8.68 Dynamic Language-Specific Commenting Functions ... 276
8.69 Dynamically Adjust Screen Brightness in Vim ... 277
8.70 Filetype-Specific Commenting Function ... 277

8.71 Filetype-Specific Dictionary Completion ... 278
8.72 Filetype-Specific Periodic Saving ... 279
8.73 GPG File Encryption Autocommands ... 279
8.74 Handle File Change Notifications Safely ... 280
8.75 Highlight long lines ... 280
8.76 Highlight yanked text ... 281
8.77 Jump to last cursor position ... 282
8.78 Language-Specific Tab Settings ... 283
8.79 Lua autocommands with pattern matching ... 283
8.80 MoinMoin File Type Detection ... 284
8.81 Periodic Background Command Execution ... 285
8.82 Persistent Command Window Line Position ... 285
8.83 Persistent Doxygen Comment Configuration ... 286
8.84 Remove Trailing Whitespace on Save ... 286
8.85 Restore Cursor to Specific Line on File Open ... 287
8.86 Restore Mode When Switching Tab Pages ... 288
8.87 Set Default Filetype for New Buffers ... 289
8.88 Set Fortran Source Form Manually ... 290
8.89 Set Syntax for Files in Specific Directory ... 290
8.90 Set UTF-8 Encoding for Mac Quicklook ... 291
8.91 Set file type based on content ... 292
8.92 Smart View Saving with Filters ... 292
8.93 Speed Up Searches by Disabling Autocmds ... 294
8.94 Spell check for specific file types ... 294
8.95 Template insertion for new files ... 295
8.96 Throttle Frequent Autocommands ... 296
8.97 Track Cursor Position in Insert Mode ... 297

9 Buffer management ... 299

9.1 Advanced CSV File Navigation and Manipulation ... 299
9.2 Advanced File Reload with Cursor Position ... 300
9.3 Advanced Hex Editing in Vim/Neovim ... 300
9.4 Automatic Write on Buffer Switch ... 302
9.5 Automatically Open All Buffers in Tabs ... 302
9.6 Automatically Switch to Existing Vim Instance ... 303
9.7 Buffer namespaces for isolated state ... 304
9.8 Buffer-local autocommands ... 305
9.9 Buffer-local keymaps ... 306
9.10 Buffer-local variables ... 306
9.11 Build a simple notes buffer system ... 307

9.12 Close all buffers ... 309
9.13 Close all buffers except the current one ... 309
9.14 Control buffer hiding behavior ... 309
9.15 Create Floating Buffer Menu in GVim ... 310
9.16 Create a GDB Console Buffer in Vim ... 311
9.17 Create custom buffer picker ... 311
9.18 Create floating window with scratch buffer ... 313
9.19 Create listed buffer ... 314
9.20 Create read-only buffer ... 314
9.21 Create scratch buffer ... 315
9.22 Create terminal buffer programmatically ... 315
9.23 Customize Buffer Menu Display for Quick Navigation ... 317
9.24 Cycle Through Buffers Efficiently ... 317
9.25 Cycle Through Buffers Intelligently ... 319
9.26 Delete Buffer Without Closing Window ... 320
9.27 Delete buffer safely ... 321
9.28 Diff Current Buffer with Saved File ... 322
9.29 Easy Buffer Switching with Tab Completion ... 323
9.30 Efficient Buffer Navigation ... 325
9.31 Efficient Buffer Switching with Tab Support ... 325
9.32 Enhanced Alternate File Navigation ... 326
9.33 Execute Command Across Multiple Buffers ... 326
9.34 Execute Commands Across Buffers Safely ... 327
9.35 Fast Buffer Navigation with Ctrl-^ ... 328
9.36 Filter Buffer Lines by Search Pattern ... 328
9.37 Flexible Buffer Switching with Function ... 329
9.38 Get buffer by name or number ... 331
9.39 Handle Modified Buffers Flexibly ... 332
9.40 Improved Hex Mode Editing in Vim ... 332
9.41 List Buffers Sorted Alphabetically ... 334
9.42 Open Multiple Files in Separate Tabs ... 335
9.43 Optimize Vim for Large File Performance ... 336
9.44 Persist Buffer List Between Vim Sessions ... 336
9.45 Preserve Window View When Switching Buffers ... 337
9.46 Prevent Tab Opening for Help and Explore ... 338
9.47 Python Help in Preview Window ... 339
9.48 Quick Buffer List and Selection Mapping ... 339
9.49 Quick Buffer List and Switch Mapping ... 340
9.50 Quick Buffer List with Timed Display ... 340
9.51 Quick Buffer Navigation Shortcuts ... 341
9.52 Quick Buffer Navigation with Mouse in GVim ... 341

9.53 Quick Buffer and Tab Management ... 342
9.54 Quick Jump to Recent Buffers ... 342
9.55 Quick SVN Diff in Vim Buffer ... 344
9.56 Reload File Without Losing Undo History ... 345
9.57 Replace Buffer List When Switching Projects ... 345
9.58 Restore Original Buffer After Buffer Operations ... 346
9.59 Restore Original Buffer After Multi-Buffer Command ... 346
9.60 Run Command Across Multiple Buffers ... 347
9.61 Save All Buffers at Once ... 347
9.62 Set buffer type for special buffers ... 348
9.63 Single Window Buffer Navigation ... 348
9.64 Smart Buffer Switching with Tabs ... 349
9.65 Store Grep Results in a Scratch Buffer ... 349
9.66 Suppress File Change Warnings in Buffer ... 350
9.67 Sync Buffer Modifiability with File Readonly State ... 351
9.68 Toggle All Buffers in Tabs ... 351
9.69 Use Temporary Files for SQL Query Results ... 352
9.70 Version Control Diff Functions ... 353
9.71 Watch buffer for changes ... 353

10 Builtin functions ... 355

10.1 Alternative Command Path Lookup in Vim ... 355
10.2 Buffer and window information ... 355
10.3 Buffer content functions ... 356
10.4 Convert Numbers Between Hex and Decimal ... 356
10.5 Create Custom Help Tags Programmatically ... 357
10.6 Cursor and mark functions ... 358
10.7 Date and time functions ... 358
10.8 Detect Comments Across Filetypes ... 359
10.9 File and directory functions ... 359
10.10 Flexible External Command Processing with system() ... 360
10.11 Fold information functions ... 360
10.12 Generate a Customizable Calendar in Vim ... 361
10.13 Get Cursor Position as Byte Percentage ... 362
10.14 Get file type and encoding ... 362
10.15 Highlighting and syntax functions ... 363
10.16 Input and interaction functions ... 363
10.17 Insert Current Date/Time with Strftime() ... 364
10.18 Line and column functions ... 365
10.19 List and dictionary functions ... 365

10.20 Mathematical functions ... 366
10.21 Measure Command Execution Time ... 366
10.22 Path manipulation functions ... 367
10.23 Print Full-Width Messages Without 'Press Enter' ... 368
10.24 Recreate Vim Temporary File Directory ... 368
10.25 Register manipulation functions ... 369
10.26 Regular expression functions ... 369
10.27 Save and Restore Cursor Position Precisely ... 370
10.28 Search and match functions ... 370
10.29 String manipulation functions ... 371
10.30 System and environment functions ... 371
10.31 Type checking functions ... 372
10.32 Window and tab functions ... 372

11 Clever tricks ... 375

11.1 Advanced Number Increment/Decrement ... 375
11.2 Alternative substitute delimiters ... 376
11.3 Auto-indent current block ... 376
11.4 Auto-indent entire document ... 376
11.5 C/C++ Macro Expansion in Vim ... 376
11.6 Calculate Math Expressions In-Editor ... 378
11.7 Calculation with expression register ... 380
11.8 Center line after jump ... 380
11.9 Change directory to current file ... 380
11.10 Change until character ... 381
11.11 Confirm Before Quitting in Insert Mode ... 381
11.12 Convert Between Hex and Decimal Numbers ... 382
11.13 Convert Between Hex and Decimal Quickly ... 383
11.14 Create Comment Boxes with External Utility ... 383
11.15 Create HTML Links with Automatic Title Fetch ... 384
11.16 Create Personal Memo File with Vim Help-Style Links ... 385
11.17 Create Postfix Abbreviations for Code Snippets ... 386
11.18 Create Text Abbreviations for Frequent Typos ... 386
11.19 Create word frequency table ... 387
11.20 Dynamic Mail Signature Rotation in Vim ... 387
11.21 Dynamic Page Count for Hardcopy Printing ... 388
11.22 Dynamic ZIP Code Lookup in Vim ... 389
11.23 Efficient Debug Print Statements in C++ ... 389
11.24 Enhanced repeat with cursor positioning ... 390
11.25 Evaluate Expressions Across Vim Modes ... 391

11.26 Evaluate Python Expressions in Vim ... 391
11.27 Execute Accidentally Inserted Normal Mode Commands ... 392
11.28 Execute Python Code Inside Vim Buffer ... 393
11.29 Extract Phone Number Under Cursor ... 394
11.30 File encoding in status line ... 394
11.31 Fix Syntax Highlighting with F12 Mapping ... 395
11.32 G-commands - case conversion ... 395
11.33 G-commands - display command output ... 396
11.34 G-commands - execute application ... 396
11.35 G-commands - format keeping cursor ... 396
11.36 G-commands - join without space ... 396
11.37 G-commands - mark navigation without jumplist ... 397
11.38 G-commands - middle of line ... 397
11.39 G-commands - put and leave cursor ... 397
11.40 G-commands - repeat substitute ... 398
11.41 G-commands - screen line movement ... 398
11.42 G-commands - search and select ... 398
11.43 G-commands - search variations ... 399
11.44 G-commands - select modes ... 399
11.45 G-commands - sleep ... 399
11.46 G-commands - undo branches ... 400
11.47 G-commands - virtual replace ... 400
11.48 Generate Unicode Character Table ... 400
11.49 Group Lines with Matching Column Pattern ... 401
11.50 Highlight Current Word for Easy Cursor Tracking ... 402
11.51 Identify Syntax Highlighting Group Under Cursor ... 403
11.52 Inline Visual Mode Calculator ... 404
11.53 Insert Current Directory Name Quickly ... 405
11.54 Insert Filename as Classname in Insert Mode ... 406
11.55 Insert Numbered Lists Easily ... 406
11.56 Inspect Character Class in Vim ... 407
11.57 Launch Multiple Vim Instances in Windows ... 407
11.58 Line completion in insert mode ... 408
11.59 List lines matching last search ... 408
11.60 Make Echo Persistent in Vim Scripts ... 409
11.61 Obscure Vim Command Obfuscation Trick ... 409
11.62 Open URLs from Current Line in Browser ... 410
11.63 Open file under cursor ... 410
11.64 Persistent Echo Messages in Vim ... 411
11.65 Platform-Specific URL Opening ... 412
11.66 Preserve Cursor Position During Script Undo ... 412

11.67 Preserve Search Register When Running Commands ... 413
11.68 Quick Hex and Decimal Number Conversion ... 414
11.69 Quick LaTeX Environment Creation in Vim ... 414
11.70 Quick Number Base Conversion ... 415
11.71 Quick Number Type Detection and Conversion ... 416
11.72 Quick Word Count in Vim ... 417
11.73 Quick Word Count in Vim/Neovim ... 418
11.74 Quick number increment ... 418
11.75 Quick substitute word ... 418
11.76 Quickly Create Perl Subroutines in Vim ... 419
11.77 Quickly Insert Image Size in HTML Tags ... 420
11.78 Quickly Obscure Text with Rot13 Encoding ... 420
11.79 Recursive Macros for Automated Text Manipulation ... 421
11.80 Repeat Last Change Efficiently ... 421
11.81 Repeat last Ex command with @: ... 422
11.82 Reverse Block Order Using Macros ... 422
11.83 Rot13 Encoding for Visible Screen Only ... 423
11.84 Rotate Email Signatures Dynamically ... 423
11.85 Save each line to separate files ... 424
11.86 Scroll windows together ... 424
11.87 Search for lines NOT matching pattern ... 425
11.88 Split line at cursor ... 425
11.89 Swap assignment statement sides ... 426
11.90 Swap two characters ... 426
11.91 Toggle File Path Slashes Quickly ... 426
11.92 Toggle text case inside a HTML tag ... 427
11.93 Transfer Text Between Vim Instances ... 428
11.94 Use Vim as a Bookmark Manager ... 428
11.95 Vim-Like Browser Navigation Extensions ... 429
11.96 Visual line selection shortcut ... 429
11.97 Z-commands - spelling corrections ... 430

12 Clipboard ... 431

12.1 Auto Copy Mouse Selection to Clipboard ... 431
12.2 Automatic Visual Selection Clipboard Copy ... 431
12.3 Clipboard Access in Cygwin Vim ... 432
12.4 Copy Entire Buffer to Clipboard ... 433
12.5 Copy Filename to Clipboard ... 433
12.6 Copy to Windows Clipboard in Cygwin Vim ... 434
12.7 Cross-Platform Filename Copying ... 435

12.8 Cross-Platform System Clipboard Integration ... 436
12.9 Cross-Session Clipboard Sharing ... 436
12.10 Disable Auto Copy Clipboard Function ... 438
12.11 Easy System Clipboard Integration ... 438
12.12 Easy Windows Clipboard Interaction ... 439
12.13 Mac OS X Clipboard Integration ... 439
12.14 Mac OS clipboard sharing ... 439
12.15 Middle-Click Paste Handling ... 440
12.16 Paste Clipboard Contents in New Vim Window ... 440
12.17 Paste from Windows Clipboard in Cygwin Vim ... 441
12.18 Paste from Windows Clipboard in Vim ... 442
12.19 Preserve register when pasting over selection ... 442
12.20 Quick Clipboard Copy/Paste Mappings ... 443
12.21 Quick Cross-Window Copy and Paste ... 443
12.22 Quick External Paste Buffer Editing ... 444
12.23 Quick System Clipboard Copy and Paste ... 445
12.24 Set system clipboard from Lua ... 446
12.25 Swap Between Vim and System Clipboard ... 446
12.26 System Clipboard Integration ... 446
12.27 System clipboard: handling yank and delete motions differently ... 447
12.28 Transfer Text Between Vim Sessions Using Clipboard ... 448
12.29 Use System Clipboard in Vim ... 448
12.30 Use xclip for Efficient Linux Clipboard Operations ... 449
12.31 Using System Clipboard Registers ... 450

13 Command line ... 451

13.1 Advanced Command Line Recall ... 451
13.2 Advanced Vim/GVim Command Line Options ... 451
13.3 Asynchronously Run External Commands in Windows ... 452
13.4 Base64 Decoding in Vim Using System Commands ... 452
13.5 Capture Ex Command Output in New Tab ... 453
13.6 Capture External Command Output Flexibly ... 454
13.7 Chain Multiple Vim Commands with Pipe ... 455
13.8 Check Remote Vim Session Mode and Context ... 455
13.9 Combine Multiple Code Checking Tools in Quickfix ... 455
13.10 Command Line Completion List Extraction ... 457
13.11 Command Window Ctrl-D Completion ... 457
13.12 Command Window Enhanced Completion ... 458
13.13 Command line editing ... 459

13.14 Command-Line Help Search Functions ... 460
13.15 Command-line completion modes ... 460
13.16 Command-line cursor movement ... 461
13.17 Command-line deletion operations ... 461
13.18 Command-line history with filtering ... 462
13.19 Command-line literal insertion ... 462
13.20 Command-line mode switching ... 462
13.21 Command-line register insertion ... 463
13.22 Command-line window access ... 463
13.23 Command-line word manipulation ... 463
13.24 Compile Current Buffer with Flexible Make Command ... 464
13.25 Compile Current File with Quickfix Support ... 464
13.26 Complex Vim Command-Line Chaining ... 465
13.27 Configure Quickfix for Visual Studio Build Errors ... 465
13.28 Create Convenient Shell Aliases ... 466
13.29 Create Custom Count Command ... 466
13.30 Create Custom Vim Commands with Autocompletion ... 467
13.31 Custom Make Command with Quickfix ... 468
13.32 Disable Command-Line Abbreviations Temporarily ... 468
13.33 Dynamically Replace Grep with Custom Search Program ... 469
13.34 Easily Capture Shell Command Output in Buffer ... 469
13.35 Enhanced Command Window Navigation ... 470
13.36 Enhanced Command-Line Completion ... 471
13.37 Escape Shell Command Special Characters ... 471
13.38 Escape Shell Commands Properly ... 472
13.39 Execute Commands Across Line Ranges ... 472
13.40 Execute Shell Command in Explorer Directory ... 473
13.41 Find Perl Path Dynamically ... 474
13.42 Fix Common Command-Line Typos ... 474
13.43 Flexible Line Range Operations in Vim ... 475
13.44 Fuzzy File Name Completion ... 475
13.45 Get Shell Command for Current Directory ... 476
13.46 Get VIMRUNTIME Path in Bash Scripts ... 476
13.47 Insert Buffer Contents into Command Line ... 477
13.48 Keep Command Window Open After Execution ... 477
13.49 List and Filter Loaded Vim Scripts ... 478
13.50 Manipulate Environment Variables in Vim ... 479
13.51 Mass File Editing with Arguments ... 480
13.52 Master Command-Line History Navigation ... 480
13.53 Perl Syntax Checking in Vim ... 481
13.54 Perl-Based Command Evaluation in Vim ... 481

13.55 Powerful Command-Line History Navigation ... 482
13.56 Powerful Global Command for Line Matching ... 482
13.57 Powerful Range-Based Text Editing ... 483
13.58 Quick Buffer and Global Commands ... 483
13.59 Quick Compile Current File with GCC ... 484
13.60 Quick Copy and Paste with pbcopy/pbpaste ... 484
13.61 Quick File Opening with Shell Command Shortcuts ... 485
13.62 Quick File Peek Without Opening Buffer ... 485
13.63 Quick Firefox URL Navigation from Vim ... 486
13.64 Quick Launch Browser from Vim ... 486
13.65 Quick Number Conversion with User Commands ... 487
13.66 Quick PHP Syntax Checking in Vim/Neovim ... 487
13.67 Quick Postgres Query Execution in Vim ... 488
13.68 Quick Vim Help in Full Window ... 488
13.69 Quick Word/WORD Paste in Command Line ... 489
13.70 Quick XML Validation with xmllint ... 489
13.71 Quick XML Wellformedness Check ... 490
13.72 Quickly Describe SQL Table from Vim ... 490
13.73 Quickly Find Full Command Path in Scripts ... 490
13.74 Quickly Inspect Vim Environment ... 491
13.75 Quickly Navigate Directories from Command Line ... 491
13.76 Quickly Open Files with Whereis Command ... 492
13.77 Quickly Repeat Previous Command ... 493
13.78 Read and Modify Environment Variables in Vim ... 493
13.79 Remote Vim Session Control via Client-Server ... 494
13.80 Remove Last Path Component in Command Line ... 495
13.81 Repeat Ex Command on Multiple Blocks ... 495
13.82 Repeat Ex Commands with Single Keystroke ... 496
13.83 Repeat Last Ex Command Efficiently ... 496
13.84 Replace Built-in Commands with Command Abbreviations ... 497
13.85 Run Commands Within Current Function ... 497
13.86 Run Java Class from Current File ... 498
13.87 Selective Line Deletion with Exceptions ... 498
13.88 Silent External Command Execution ... 499
13.89 Source Vim Script Directly from Command Line ... 499
13.90 Use Quickfix Window for Compile Errors ... 500
13.91 Use Quickfix with Doxygen for Error Navigation ... 500
13.92 Verbose Command Execution Debugging ... 501

14 Command line (advanced) ... 503

14.1 Command line abbreviations and shortcuts ... 503
14.2 Command line advanced search operations ... 503
14.3 Command line advanced substitution techniques ... 503
14.4 Command line buffer and window targeting ... 504
14.5 Command line completion customization ... 504
14.6 Command line conditional execution ... 504
14.7 Command line custom command creation ... 505
14.8 Command line debugging and inspection ... 505
14.9 Command line environment variable integration ... 505
14.10 Command line error handling ... 506
14.11 Command line expression evaluation ... 506
14.12 Command line external command integration ... 507
14.13 Command line filename completion variations ... 507
14.14 Command line history search and filtering ... 507
14.15 Command line job control and async ... 508
14.16 Command line macro recording and playback ... 508
14.17 Command line range shortcuts ... 508
14.18 Command line register manipulation ... 509
14.19 Command line script execution ... 509
14.20 Command line substitution flags and modifiers ... 509
14.21 Command line terminal integration ... 510
14.22 Command line window operations ... 510

15 Community tips ... 511

15.1 Advanced completion shortcuts ... 511
15.2 Buffer-specific settings ... 511
15.3 Builtin completion without plugins ... 511
15.4 Command abbreviations ... 512
15.5 Command-line window editing ... 512
15.6 Dynamic plugin management ... 513
15.7 Efficient whitespace cleanup ... 513
15.8 Environment-aware configuration ... 513
15.9 Help in new tab workflow ... 514
15.10 Insert mode line manipulation ... 514
15.11 Insert mode navigation ... 514
15.12 Mark-based navigation workflow ... 515
15.13 Modular configuration loading ... 515
15.14 Motion-based editing patterns ... 515
15.15 Quick fold navigation ... 516

15.16 Register operations mastery ... 516
15.17 Session workflow optimization ... 517
15.18 Split window mastery ... 517
15.19 Tab-based workflow ... 517

16 Completion ... 519

16.1 API/Module completion ... 519
16.2 Async completion with vim.schedule ... 520
16.3 Cached completion for performance ... 521
16.4 Completion items with detailed info ... 523
16.5 Completion with external command ... 524
16.6 Context-aware completion ... 525
16.7 Custom completion function basics ... 526
16.8 File path completion ... 527
16.9 Fuzzy matching completion ... 529
16.10 LSP-style completion with documentation ... 530
16.11 Multi-source completion chaining ... 531
16.12 Snippet-style completion ... 533

17 Configuration ... 535

17.1 Add 'Edit with Vim' Context Menu in Windows ... 535
17.2 Add Context Menu Option to Open Vim in Directory ... 535
17.3 Add Modeline for Perl Batch Files ... 536
17.4 Add Vim Context Menu in Nemo File Manager ... 536
17.5 Add Vim to Windows Explorer Context Menu ... 537
17.6 Advanced CTags Configuration for ANT Files ... 538
17.7 Advanced Syntax Sync for Large Files ... 538
17.8 Alternate Neovim startup configuration ... 539
17.9 Alternative Bell Suppression Method ... 539
17.10 Alternative CapsLock to Control Mapping ... 539
17.11 Alternative CapsLock to Escape Mapping ... 540
17.12 Alternative Escape Key Mapping ... 540
17.13 Always Display Status Line ... 540
17.14 Auto tab completion ... 541
17.15 Auto-Reload Vimrc on Save ... 542
17.16 Auto-continue Doxygen Comments in C/C++ ... 542
17.17 Auto-detect CPU cores for parallel make builds ... 543
17.18 Autoload Cscope Database Recursively ... 543
17.19 Automated Patch Backup with Patchmode ... 544
17.20 Automatically Add Modeline to Files ... 545

17.21 Automatically Add Python Paths to Vim Path ... 545
17.22 Automatically Continue Doxygen Comments ... 546
17.23 Automatically Reload Changed Files ... 547
17.24 Automatically Save Global Variables to viminfo ... 547
17.25 Automatically Save Session and Viminfo on Exit ... 548
17.26 Automatically Source Session on Startup ... 549
17.27 Bright Background Colors in Linux Console ... 549
17.28 Build Vim on Ubuntu with GUI Support ... 550
17.29 Build Vim with Custom Compilation Info ... 550
17.30 Building Vim with Extensive Features ... 551
17.31 Building Vim with Python Support on Windows ... 551
17.32 Cache and Reset Vim Options Dynamically ... 552
17.33 Cache and Restore Vim Option Preferences ... 553
17.34 Chain Commands in Vimrc Mappings ... 554
17.35 Check Vim Installation Details ... 555
17.36 Check Vim Installation Features ... 555
17.37 Check and Set Buffer Filetype ... 556
17.38 Check plugin key mapping usage ... 556
17.39 Choose the Right Vim Build for Your System ... 557
17.40 Choosing the Right Vim Version ... 557
17.41 Clone Vim Source Using Mercurial ... 558
17.42 Commenting Configuration Files ... 558
17.43 Compile Vim with Python Support Separately ... 558
17.44 Completely Disable Syntax Highlighting and Search Highlights ... 559
17.45 Completely Silence Vim Bells ... 559
17.46 Comprehensive List of Special Keys ... 560
17.47 Conditionally Source Runtime Scripts ... 560
17.48 Configure 256-Color Terminal Support in Vim ... 561
17.49 Configure Automatic Indentation Methods ... 561
17.50 Configure Backspace Behavior in Insert Mode ... 562
17.51 Configure COBOL Syntax Highlighting in Neovim ... 562
17.52 Configure Clipboard Behavior ... 563
17.53 Configure Diff Options Flexibly ... 563
17.54 Configure Errorformat for Intel Fortran Compiler ... 564
17.55 Configure External Diff Program ... 564
17.56 Configure External Diff Tool ... 565
17.57 Configure Fonts in Neovim on macOS ... 566
17.58 Configure GUI Font in Neovim ... 566
17.59 Configure IPython Editor in Vim ... 567
17.60 Configure Indentation Styles in Vim/Neovim ... 567

17.61 Configure Man Page Viewing ... 568
17.62 Configure Mouse Behavior ... 569
17.63 Configure Number Format Options ... 569
17.64 Configure Path for File Searching ... 570
17.65 Configure SSH Tunneling for SVN ... 570
17.66 Configure Syntax Highlighting Options ... 571
17.67 Configure Terminal Color Highlighting ... 571
17.68 Configure Terminal Colors on Solaris ... 572
17.69 Configure Terminal Cursor for Different Modes ... 572
17.70 Configure Unicode and UTF-8 Encoding ... 573
17.71 Configure Vim Color Support on HP-UX ... 574
17.72 Configure Vim Language and Encoding Support ... 574
17.73 Configuring External Diff Program ... 575
17.74 Convert Note Files to Vim Help Documents ... 575
17.75 Create Consistent Color Scheme Between Vim and GVim ... 576
17.76 Create Custom Color Scheme Extending Existing Scheme ... 577
17.77 Create Custom Syntax Highlighting for Files ... 578
17.78 Create Desktop-Specific Vim Instances ... 579
17.79 Create TODO Tag Lists with Ctags ... 579
17.80 Create Vim Translation Files ... 580
17.81 Create and Install Language Translation Files ... 580
17.82 Cross-Platform GUI Font Configuration ... 580
17.83 Cross-Terminal Cursor Shape Detection ... 581
17.84 Custom Filetype Detection for Unique File Extensions ... 582
17.85 Custom Make Command for Specific Compiler ... 583
17.86 Custom Syntax Highlighting for File Types ... 583
17.87 Custom Syntax Highlighting for Mixed Languages ... 584
17.88 Custom Syntax Highlighting for Specific Filetypes ... 584
17.89 Customize Completion Sources ... 585
17.90 Customize Cursor Color and Shape Across Modes ... 585
17.91 Customize Errorformat for Different Compilers ... 586
17.92 Customize File Path Detection ... 586
17.93 Customize File Path Detection with isfname ... 587
17.94 Customize Perl Documentation Lookup ... 587
17.95 Customize Python Triple-Quoted Comments ... 588
17.96 Customize Syntax Highlighting Colors ... 588
17.97 Customized Vim Configuration for Productivity ... 589
17.98 Cycle Numeric Options with Flexible Range ... 590
17.99 Debug Unexpected Vim Option Settings ... 590
17.100 Detect Filetype by File Contents ... 591
17.101 Detect GUI Running in Vim/Neovim ... 592

17.102 Diagnose Backspace Key Mapping Issues 592
17.103 Diagnose Vim Menu Bar Placement Issues 593
17.104 Diagnose and Clear Problematic Backspace Mappings 593
17.105 Disable ANSI Escape Sequences in Man Pages 594
17.106 Disable Auto-Indentation for Specific Filetypes 594
17.107 Disable Automatic Comment Continuation 595
17.108 Disable Automatic Comment Insertion 595
17.109 Disable Default Filetype Plugins 596
17.110 Disable IME Input Mode in Windows 596
17.111 Disable Mouse Visual Mode Selection 597
17.112 Disable Python Omnicomplete Preview Window 597
17.113 Disable Terminal Flow Control for Vim 598
17.114 Disable Vim Bells and Visual Flashes 598
17.115 Discover Special Key Names in Vim 599
17.116 Display Indentation Level in Status Line 599
17.117 Display Non-ASCII Characters in Console 600
17.118 Download Vim for Windows 600
17.119 Dynamic Color Scheme for Context Awareness 601
17.120 Dynamic Cscope Database Generation 601
17.121 Dynamic Doxygen Comment Configuration 602
17.122 Dynamic File-Specific Vim Settings with Modelines 603
17.123 Dynamic Include Path for C++ Intellisense 603
17.124 Dynamic PHP Function Documentation Lookup 604
17.125 Dynamic Syntax Sync for Performance 604
17.126 Dynamically Detect and Handle Screen Terminal 605
17.127 Dynamically Manage Vim Path Settings 605
17.128 Easy vimrc File Management 606
17.129 Enable 256 Color Support in Vim/Neovim 607
17.130 Enable Filetype Detection and Plugins 607
17.131 Enable Mouse Support in Terminal Vim 608
17.132 Enable Omni Completion for Smart Autocompletion 608
17.133 Enable Windows Shortcuts in GVim 609
17.134 Enable Windows Shortcuts in gVim 609
17.135 Enable Windows-like Selection Mode 609
17.136 Enhance Vim Usability with Essential Settings 610
17.137 Enhanced COBOL Syntax Highlighting for Modern COBOL 610
17.138 Enhanced Linux Console Color Configuration 611
17.139 Ensure Vim Compatibility Mode is Off 611
17.140 Environment Variable Management for Custom Vim 612
17.141 Environment variables in configuration 612
17.142 Escape Spaces in Font Names 613

17.143 Essential Vim Configuration Settings ... 613
17.144 Ex commands - autocmds and events ... 614
17.145 Ex commands - highlight and syntax ... 615
17.146 Ex commands - mappings and abbreviations ... 615
17.147 Ex commands - runtime and sourcing ... 616
17.148 Extend Keyword Characters for Namespaces ... 617
17.149 Extend iskeyword for XML Namespaces ... 617
17.150 Fine-tune Terminal Color Palette ... 617
17.151 Fix Arrow Keys in Remote Shell Terminals ... 618
17.152 Fix Arrow Keys in Remote Shell Vim ... 618
17.153 Fix Arrow Keys in Visual Mode on Windows ... 619
17.154 Fix Backspace Behavior in Insert Mode ... 619
17.155 Fix Backspace Key in PuTTY Terminal ... 620
17.156 Fix Backspace in PuTTY Terminal Vim ... 620
17.157 Fix Broken Syntax Highlighting ... 621
17.158 Fix Crontab Editing in Vim/Neovim ... 621
17.159 Fix Cygwin Executable Path in Windows Taskbar Vim ... 622
17.160 Fix Esc Key and Terminal Keymappings in Screen ... 623
17.161 Fix Escape Key Timing in Screen Terminal ... 623
17.162 Fix Meta/Alt Keys in Terminal Vim ... 624
17.163 Fix Missing GVim Menu Bar in GNOME ... 624
17.164 Fix Modeline File Encoding Detection ... 625
17.165 Fix PuTTY Numeric Keypad Behavior ... 626
17.166 Fix Shell Command Issues in FreeBSD GVim ... 626
17.167 Fix Syntax Highlighting Globally ... 626
17.168 Fix Syntax Highlighting Reliably ... 627
17.169 Fix Syntax Highlighting with Sync Methods ... 628
17.170 Fix Temp File Errors on Windows ... 629
17.171 Fix Temporary File Errors on Windows ... 629
17.172 Fix Temporary File Issues in Windows Vim ... 629
17.173 Fix Terminal Arrow Key Navigation Issues ... 630
17.174 Fix Trackpoint Scrolling in Vim ... 630
17.175 Fix Vim Configuration in Cygwin ... 631
17.176 Fix Vim File Association Issues ... 631
17.177 Flexible CSV Delimiter Configuration ... 632
17.178 Flexible Compiler Plugin Management ... 632
17.179 Flexible Error Message Parsing ... 633
17.180 Flexible File Path Configuration ... 634
17.181 Flexible Indentation Strategies in Neovim ... 634
17.182 Flexible LaTeX Compilation with Makefiles ... 635
17.183 Flexible LaTeX Compilation with Rubber and Make ... 636

17.184 Flexible Number Format Handling ... 636
17.185 Flexible Option Flag Management ... 637
17.186 Flexible Tab to Spaces Conversion ... 637
17.187 Force File Encoding to Latin1 ... 638
17.188 Force Syntax Highlighting for Custom File Extensions ... 639
17.189 GUI-Specific Configuration ... 639
17.190 GUI-Specific Vim Configuration ... 640
17.191 Generate Project CTags Quickly ... 640
17.192 Handle Character Encoding When Switching to UTF-8 ... 641
17.193 Handle Encoding Transitions Safely ... 641
17.194 Handle Filenames with Spaces ... 642
17.195 Handle Terminal Encoding for Non-Unicode Terminals ... 642
17.196 Hidden buffers option ... 643
17.197 Home key smart mapping ... 643
17.198 Ignore Whitespace Differences in Diff Mode ... 644
17.199 Improve Vim Learning and Productivity ... 645
17.200 Improve XML/XSLT Attribute Completion ... 645
17.201 Improved Java Compilation Error Handling ... 646
17.202 Increase Menu Item Limit ... 646
17.203 Increase Redraw Time for Large Files ... 647
17.204 Install Full-Featured Vim on Linux ... 647
17.205 Install Multiple Vim Versions Simultaneously ... 648
17.206 Integrate PHP Manual into Vim Help ... 648
17.207 Integrate Vim as Diff Tool for TortoiseCVS ... 649
17.208 Internationalize Vim Messages and Menus ... 649
17.209 Investigate Mappings and Abbreviations ... 649
17.210 J2ME Development Vim Configuration ... 650
17.211 Java Build Error Handling in Vim ... 651
17.212 Language-Specific Compiler and Error Handling ... 652
17.213 Launch Vim Inside Visual Studio MDI ... 652
17.214 Lightweight Project Sessions per Tab ... 653
17.215 Live Word Count in Status Line ... 653
17.216 Manage Filetype Detection Priorities ... 654
17.217 Manage Language and Encoding Switches ... 655
17.218 Manage Multiple Project Sessions ... 655
17.219 Manage Multiple Vim Versions with Symbolic Links ... 656
17.220 Managing Vim Source Code with Mercurial ... 656
17.221 Manually Add Missing X11 Fonts for Vim ... 657
17.222 Manually Set Syntax Highlighting ... 657
17.223 Map CapsLock to Escape in X11 Systems ... 658
17.224 Markdown code block syntax highlighting ... 658

17.225 Minimal Vim Configuration Testing 659
17.226 Modify C/C++ Comment Behavior 659
17.227 Modify Option Flags Dynamically 660
17.228 Multiple Vim Instances for Different File Types 660
17.229 OS-Specific Configuration 661
17.230 Open Qt Documentation from Vim 661
17.231 Optimize Java Code Indentation in Neovim 662
17.232 Optimize Java Indentation and Syntax Highlighting 663
17.233 Organize Language-Specific Vim Settings 664
17.234 Organize Swap and Backup Files in Dedicated Directories 664
17.235 Override Color Scheme Colors Dynamically 665
17.236 Override Terminal Color Capabilities 666
17.237 Per-project configuration with .nvim.lua 666
17.238 Persist Color Scheme Across Vim Sessions 667
17.239 Persist Vim Window Size and Position Across Sessions 667
17.240 Preserve Hard Links When Editing Files 668
17.241 Preserve Indent When Typing # in Insert Mode 668
17.242 Prevent Multiple Plugin Loads 669
17.243 Prevent Multiple Vim Instances 669
17.244 Prevent Vim GUI from Forking Processes 670
17.245 Print Files Using KDE Print Dialog 670
17.246 Profile Vim Startup Time 671
17.247 Quick Access to Vimrc File 671
17.248 Quick CapsLock to Escape Command 672
17.249 Quick Java Compilation in Vim/Neovim 672
17.250 Quick PHP Function Documentation Lookup 673
17.251 Quick Spelling Correction Setup 674
17.252 Quick Vim Configuration Editing and Reloading 674
17.253 Quick Vim Configuration Management 675
17.254 Quickly Disable Syntax Highlighting During Editing 675
17.255 Reduce 'Hit ENTER' Prompts in Vim/Neovim 676
17.256 Remap Caps Lock to Escape or Ctrl 676
17.257 Remap CapsLock to Escape System-Wide 676
17.258 Remap CapsLock to Escape/Ctrl Systemwide 677
17.259 Remove '=' from Filename Characters 677
17.260 Remove Specific GUI Elements 678
17.261 Remove from option value 678
17.262 Resolve Remote SSH Vim Key Mapping Issues 678
17.263 Restore Cursor Position Across Editing Sessions 679
17.264 Safely Customize Syntax Files 680
17.265 Safely Manage Plugin Configurations 680

17.266 Safely Source and Modify Vimrc 681
17.267 Safely Store and Manage Key Mappings 682
17.268 Sandbox mode for safe testing 682
17.269 Save Macros Persistently 683
17.270 Save and Restore Macros Across Sessions 683
17.271 Secure Password File with Encryption and Folding 683
17.272 Secure Vim Configuration for Encrypted Files 684
17.273 Set Custom VIMRUNTIME Path in Config 685
17.274 Set Fortran Source Form for .for Files 685
17.275 Set Initial Vim Window Size 686
17.276 Set Terminal Type for Color Support 686
17.277 Set Terminal Type for Color Vim 687
17.278 Set Vim Language and Encoding for Internationalization 687
17.279 Set Vim as Default Editor for Unknown File Types 688
17.280 Set Vim as Default PostgreSQL Editor 688
17.281 Shorten Long Messages in Vim 689
17.282 Show File Encoding in Status Line 689
17.283 Show Longest Common Completion Text 690
17.284 Show Matching Braces Instantly 690
17.285 Smart Language-Specific Line Commenting 691
17.286 Specify Custom SCP Port 692
17.287 Speed Up Vim Startup in Remote Sessions 692
17.288 Speed up vimgrep with noautocmd 693
17.289 Swap CapsLock and Escape Keys 693
17.290 Systematic Vim Debugging via Bisection 693
17.291 Terminal-Specific Cursor Shape and Color 694
17.292 Toggle Boolean Options Easily 695
17.293 Toggle Caps Lock in Vim with Custom Functions 695
17.294 Toggle Editing Helpers with Function Key 696
17.295 Toggle GUI Elements in Neovim 697
17.296 Toggle Line Numbers Quickly 698
17.297 Toggle Options Easily 699
17.298 Toggle Paste Mode to Preserve Indentation 699
17.299 Toggle Vim Option Flags Dynamically 700
17.300 Toggle Vim Options Easily 701
17.301 Toggle Word Completion and Paste Mode 701
17.302 Toggle Wrapping with Cursor Movement Handling 702
17.303 Toggle and Manage Vim Boolean Options 703
17.304 Troubleshoot Color Scheme Loading Issues 704
17.305 Troubleshoot Terminal Color Issues 704
17.306 Ubuntu Development Package Dependencies 705

17.307 Understand Vim's Indentation Options ... 705
17.308 Unique Swap Files Across Projects ... 706
17.309 Update Vim Source with Mercurial ... 706
17.310 Use SuperTab for Easy Omni Completion ... 707
17.311 Use Unix Line Endings for Cross-Platform Vim Scripts ... 707
17.312 Use rsync to Sync Vim Runtime Files ... 707
17.313 Use scriptencoding for Vimrc Character Sets ... 708
17.314 Use scriptencoding for Vimrc Encoding ... 708
17.315 User-Specific File Associations in Windows ... 709
17.316 Verbose Option Checking Without Noise ... 709
17.317 Verifying Python Support in Vim/Neovim ... 710
17.318 Version Control for Vim Configuration ... 710
17.319 Version-Independent Vim Installation on Windows ... 711
17.320 Version-Safe Script Loading ... 711
17.321 View Java Docs Directly in Vim ... 712
17.322 View runtime paths ... 712
17.323 Vim-Style Editing in Visual Studio ... 713
17.324 Windows Vim Download Options ... 713
17.325 Windows Vim Installation After Compilation ... 713
17.326 Windows-like Behavior in Vim ... 714

18 Cut and paste ... 715
18.1 Cut/delete word ... 715
18.2 Paste text ... 715
18.3 Paste with automatic indentation ... 715
18.4 Yank line ... 716
18.5 Yank word ... 716

19 Diagnostics ... 717
19.1 Configure diagnostic display with virtual text ... 717
19.2 Find mapping source ... 718
19.3 Find option source ... 718
19.4 Health diagnostics ... 718
19.5 View messages ... 718

20 Display ... 719
20.1 Automatic Status Line Update with Timer ... 719
20.2 Center Cursor Vertically in Window ... 719
20.3 Conceal text with syntax highlighting ... 720

20.4 Conditionally Set Command Height ... 721
20.5 Configure Scrolloff for Context ... 721
20.6 Custom Syntax Highlighting Colors ... 722
20.7 Custom Syntax Highlighting for Printing ... 722
20.8 Customize GVim Colors to Match Console Vim ... 723
20.9 Customize Line Number Appearance ... 723
20.10 Customize PuTTY Colors for Vim Desert Scheme ... 724
20.11 Display Character ASCII/Unicode Value ... 724
20.12 Display Date and Time in Status Line ... 725
20.13 Dynamic Color Scheme Switcher ... 725
20.14 Dynamic Color Testing Across Terminal and GUI ... 726
20.15 Dynamic Cursor Line Highlighting After Jumps ... 727
20.16 Dynamic Font Size Adjustment in GUI ... 728
20.17 Ex commands - folding display ... 729
20.18 Ex commands - status line and tabs ... 730
20.19 Force Redraw to Prevent Message Disappearance ... 730
20.20 Get Character Information Quickly ... 731
20.21 Get Character Value in Command Mode ... 731
20.22 Highlight Code Between Matching Curly Braces ... 731
20.23 Highlight Current Line and Column ... 732
20.24 Highlight Cursor Line in Insert Mode ... 733
20.25 Highlight Debug Blocks in C/C++ Code ... 733
20.26 Highlight Lines Exceeding Length Limit ... 735
20.27 Highlight Long Lines ... 735
20.28 Highlight Matching Braces ... 736
20.29 Highlight Multiple Words in Different Colors ... 736
20.30 Highlight Multiple Words with Different Colors ... 737
20.31 Highlight Problematic Whitespace ... 738
20.32 Highlight Unwanted Whitespace ... 739
20.33 Highlight Valid IP Addresses ... 740
20.34 Highlight and Manage Unwanted Whitespace ... 740
20.35 Ignore Whitespace in Diff Mode ... 741
20.36 Improve Long Line Readability ... 742
20.37 Improve Wrapped Line Handling ... 742
20.38 Print Syntax Highlighted Buffer in Black and White ... 743
20.39 Print with Line Numbers ... 743
20.40 Quick Background Color Toggle ... 744
20.41 Quick Syntax Color Customization ... 744
20.42 Quickly Customize Syntax Highlighting Colors ... 744
20.43 Quickly Highlight Current Line ... 745
20.44 Relative Line Numbers for Easy Navigation ... 745

20.45 Screen Drawing and Color Output in Vim Scripts ... 746
20.46 Show ASCII/Unicode Value of Character ... 747
20.47 Show Whitespace with List Chars ... 747
20.48 Smart Line Number and Column Display ... 748
20.49 Soft Line Wrapping ... 748
20.50 Toggle Between Light and Dark Color Schemes ... 749
20.51 Toggle Cursor Line and Column Highlighting ... 749
20.52 Toggle Diff Highlighting in Merge Conflicts ... 750
20.53 Toggle Tab Visualization Function ... 750
20.54 Toggle invisible characters ... 751
20.55 Truncate Long Echoed Messages ... 752
20.56 View and Test Available Colors in Vim/Neovim ... 752
20.57 Visualize Tab Characters in Vim ... 753
20.58 Visually Highlight Unwanted Whitespace ... 754

21 Edit ... 755

21.1 Adding prefix/suffix to multiline text easily ... 755
21.2 Common operators with motions ... 755
21.3 Operator-pending mode - cancel operations ... 755
21.4 Operator-pending mode - force operation type ... 756
21.5 Operator-pending mode basics ... 756
21.6 Operator-pending mode with text objects ... 756
21.7 Redraw screen ... 757
21.8 Repeat last change ... 757
21.9 Show file information ... 757
21.10 Substitute characters ... 758
21.11 Time-based undo navigation ... 758

22 Editing ... 759

22.1 Add Unicode Combining Characters to Text ... 759
22.2 Advanced Word Replacement Techniques ... 760
22.3 Alternative Word Deletion Methods ... 760
22.4 Auto Spelling Correction with Abbreviations ... 761
22.5 Auto-Close HTML Tags with Omni Completion ... 762
22.6 Autocomplete Words in Insert Mode ... 762
22.7 Block Commenting in Lisp ... 763
22.8 Block Line Sorting with Global Command ... 763
22.9 Calculate expressions ... 763
22.10 Case Modification Techniques ... 764
22.11 Comment Code with Flexible Delimiters ... 764

22.12 Compact Multiple Blank Lines ... 765
22.13 Condense Multiple Blank Lines ... 765
22.14 Consistent Yank to End of Line ... 766
22.15 Convert Between CamelCase and snake_case ... 766
22.16 Convert Enum to String Table Automatically ... 767
22.17 Convert Variable Naming Styles ... 768
22.18 Copy and Replace Lines Across Document ... 768
22.19 Copy and Replace Words Across File ... 769
22.20 Copy and Replace Words Across Multiple Locations ... 769
22.21 Copy and Replace Words Efficiently ... 770
22.22 Copy and Replace Words Quickly ... 770
22.23 Copy and move lines to marks ... 771
22.24 Delete Lines Matching a Pattern ... 771
22.25 Delete XML/HTML Tags Efficiently ... 772
22.26 Delete to Custom Delimiter Efficiently ... 772
22.27 Dictionary-Based Word Completion ... 773
22.28 Duplicate Previous Line Word by Word ... 773
22.29 Easy Character and Word Swapping ... 774
22.30 Easy Comment/Uncomment for Multiple Languages ... 775
22.31 Easy Word Replacement Across Files ... 776
22.32 Edit file at specific line ... 776
22.33 Efficient Text Completion and Navigation ... 776
22.34 Efficient Undo and Redo ... 777
22.35 Enter Characters by Unicode Value ... 777
22.36 Ex commands - joining and splitting ... 778
22.37 Ex commands - line operations ... 778
22.38 Ex commands - marks and jumps ... 779
22.39 Ex commands - sorting and formatting ... 779
22.40 Ex commands - undo and redo ... 779
22.41 Execute normal commands without mappings ... 780
22.42 Expand Abbreviations Without Extra Characters ... 780
22.43 Extend Word Completion with Precision ... 781
22.44 Fast Print Statement Generation ... 781
22.45 Filter Text Using External Shell Commands ... 781
22.46 Fix Accidental 4 Instead of $ in PHP ... 782
22.47 Fix Numeric Prefix Mistake in PHP Variables ... 782
22.48 Flexible Case Conversion in Text ... 783
22.49 Flexible Case Conversion with Visual Selection ... 783
22.50 Generate Numbered Lists Quickly ... 784
22.51 Global command with normal mode operations ... 785
22.52 HTML Entity Conversion Utility ... 785

22.53 Increment Numbers in Visual Block 786
22.54 Increment Numbers in Visual Selection 787
22.55 Increment and Decrement Numbers Quickly 787
22.56 Increment search results 788
22.57 Increment/Decrement Numbers Flexibly 788
22.58 Insert Current Filename 789
22.59 Insert Filename in Different Modes 790
22.60 Insert Filename in Neovim 790
22.61 Insert Line Numbers in File 791
22.62 Insert Multiple Lines Efficiently 791
22.63 Insert Numbered List with Range Function 792
22.64 Insert Special Characters Quickly 793
22.65 Insert Special Characters with Digraphs 793
22.66 Insert Unicode Characters Easily 794
22.67 Insert at beginning/end 794
22.68 Insert multiple lines 795
22.69 Insert newline without entering insert mode 795
22.70 Insert single character 796
22.71 Inspect Unicode Character Details 796
22.72 Join Lines Matching a Pattern Globally 797
22.73 Join Lines with Customizable Spacing 797
22.74 Line Replacement Across Multiple Locations 797
22.75 Merge Terminal-Split Long Lines 798
22.76 Move Comment to End of Line Efficiently 798
22.77 Move line to end of paragraph 799
22.78 Move lines to marks 799
22.79 Number Lines Sequentially in Visual Selection 799
22.80 Number Lines Using External Utility 800
22.81 Number Lines with External Filter 801
22.82 Open new line 801
22.83 Outline Manipulation Techniques 801
22.84 Paste with Automatic Indentation 802
22.85 Powerful Text Case Manipulation 803
22.86 Preserve Cursor Position When Pasting 803
22.87 Put (paste) operations 803
22.88 Put text above or below current line 804
22.89 Quick Email Quote Formatting Techniques 804
22.90 Quick Email Thread Cleanup 805
22.91 Quick Footnote Insertion Abbreviations 805
22.92 Quick Heading Underline with Dashes 806
22.93 Quick Minify File in Vim 806

22.94 Quick PHP Debug Code Insertion 807
22.95 Quick Paragraph Line Joining 807
22.96 Quick Paragraph Sorting in Normal Mode 808
22.97 Quick Separating Line Abbreviations 808
22.98 Quick Text Filtering with External Commands 809
22.99 Quick Word Completion in Insert Mode 809
22.100 Quick Word Deletion in Insert Mode 810
22.101 Quick Word Replacement with Stamping 810
22.102 Quick Word and Line Editing Commands 810
22.103 Quickly Add/Delete Empty Lines 811
22.104 Quickly Change Words and Lines 812
22.105 Quickly Comment Lines in Visual Mode 813
22.106 Quickly Insert Current Date or Time 813
22.107 Reflow Paragraphs Quickly 814
22.108 Remove All Blank Lines in a File 814
22.109 Remove Automatic Comment Leaders Quickly 814
22.110 Remove Diacritical Signs from Characters 815
22.111 Remove Empty Lines Efficiently 816
22.112 Repeat Last Change with Dot Command 816
22.113 Restore Cursor Position in Previous Session 817
22.114 Return to last exit position 817
22.115 Reverse Lines Using External Command 818
22.116 Reverse Lines in Buffer 818
22.117 Reverse Lines in a Specific Range 819
22.118 Select non-uniform strings across lines 819
22.119 Smart Word Replacement Across Files 819
22.120 Start Editing Blank Lines with Correct Indent 820
22.121 Substitute character 820
22.122 Substitute in all buffers 821
22.123 Swap Characters and Words Easily 821
22.124 Temporarily Disable Auto-Indent for Pasting 822
22.125 Temporarily Disable Auto-Indentation for Pasting 822
22.126 Toggle Paste Mode Easily 822
22.127 Trigger Omni Completion in Insert Mode 823
22.128 Use Omni Completion Shortcuts 823
22.129 Use Text Filters to Process Line Ranges 824
22.130 Wrap text in HTML tags 824
22.131 Yank (copy) operations 825

23 Ex commands (advanced) ... 827

23.1 Advanced substitute flags ... 827
23.2 Append text after line ... 827
23.3 Browse with file dialog ... 827
23.4 Center align text ... 828
23.5 Change lines with text entry ... 828
23.6 Change tab settings and convert ... 828
23.7 Command history navigation ... 829
23.8 Execute on non-matching lines ... 829
23.9 Global with range ... 829
23.10 Insert text before line ... 829
23.11 Join lines together ... 830
23.12 Keep jump list during operation ... 830
23.13 Keep marks during operation ... 830
23.14 Left align text ... 831
23.15 List old files ... 831
23.16 Load saved view ... 831
23.17 Lock marks during operation ... 831
23.18 Make session file ... 832
23.19 Nested global commands ... 832
23.20 Put register contents ... 832
23.21 Quit with error code ... 833
23.22 Range with patterns ... 833
23.23 Return to normal mode ... 833
23.24 Right align text ... 833
23.25 Save current view ... 834
23.26 Sort lines alphabetically ... 834
23.27 Substitute confirmation ... 834
23.28 Substitute with backreferences ... 835
23.29 Substitute with expressions ... 835
23.30 Write all and quit all ... 835
23.31 Write and exit ... 836
23.32 Yank lines to register ... 836

24 Ex commands (comprehensive) ... 837

24.1 Buffer list navigation ... 837
24.2 Close all windows except current ... 837
24.3 Copy lines to another location ... 837
24.4 Create new empty buffer ... 838
24.5 Delete buffers ... 838

24.6 Delete specific lines ... 838
24.7 Execute normal mode commands ... 839
24.8 Find file in path ... 839
24.9 First and last files in argument list ... 839
24.10 Go to specific buffer by number ... 840
24.11 Internal grep with vimgrep ... 840
24.12 Jump to tag definition ... 840
24.13 List all sourced scripts ... 840
24.14 Location list navigation ... 841
24.15 Move lines to another location ... 841
24.16 Next file in argument list ... 841
24.17 Previous file in argument list ... 842
24.18 Previous tag in stack ... 842
24.19 Quickfix list navigation ... 842
24.20 Quit all windows/buffers ... 843
24.21 Recover file from swap ... 843
24.22 Repeat last Ex command ... 843
24.23 Run grep and jump to matches ... 844
24.24 Save all modified buffers ... 844
24.25 Set local options ... 844
24.26 Show argument list ... 844
24.27 Show version information ... 845
24.28 Source Vim scripts ... 845

25 Ex commands (extended) ... 847

25.1 Add buffer to list ... 847
25.2 AutoGroup management ... 847
25.3 Call functions ... 847
25.4 Change working directory ... 848
25.5 Check file path existence ... 848
25.6 Conditional execution ... 848
25.7 Create abbreviations ... 849
25.8 Define variables ... 849
25.9 Echo text and expressions ... 849
25.10 File information ... 850
25.11 Function definition ... 850
25.12 Help search ... 850
25.13 Include jump ... 851
25.14 Include list ... 851
25.15 Introduction screen ... 851

25.16 Key mapping ... 851
25.17 Language settings ... 852
25.18 Make and build ... 852
25.19 Match highlighting ... 852
25.20 Menu creation ... 853
25.21 Neovim health check ... 853
25.22 Print lines ... 853
25.23 Runtime file loading ... 854
25.24 Show all marks ... 854
25.25 Show all messages ... 854
25.26 Show digraphs ... 854
25.27 Show jump list ... 855
25.28 Show registers content ... 855
25.29 Spell checking commands ... 855
25.30 Tag selection ... 856
25.31 Unlet variables ... 856

26 Exit ... 857
26.1 Exit with error code ... 857
26.2 Quit Vim ... 857

27 Extmarks ... 859
27.1 Add sign column signs with extmarks ... 859
27.2 Add virtual text with extmarks ... 859
27.3 Build a simple word abbreviation system ... 860
27.4 Clear extmarks from buffer ... 862
27.5 Create basic extmark ... 862
27.6 Create custom highlighting with extmark priorities ... 862
27.7 Create inline diagnostics with extmarks ... 863
27.8 Create line number decorations with extmarks ... 864
27.9 Create virtual lines with extmarks ... 864
27.10 Get all extmarks in range ... 865
27.11 Hide text with extmark conceal ... 865
27.12 Highlight text ranges with extmarks ... 866
27.13 Make extmarks persistent across edits ... 867
27.14 Monitor extmark changes with events ... 867
27.15 Replace keywords visually with extmarks ... 868

28 File operations ... 871

28.1 Add Modeline to Permanently Protect File ... 871
28.2 Add Vim to Windows Send To Menu ... 871
28.3 Associate Extensionless Files with Vim in Windows ... 872
28.4 Auto-Detect and Reload Mixed Line Endings ... 872
28.5 Automatic Remote File Synchronization ... 873
28.6 Automatically Create Backup and Tmp Directories ... 873
28.7 Automatically Create Directories When Editing Files ... 875
28.8 Batch Convert Line Endings Across Files ... 876
28.9 Batch Convert Line Endings for Multiple Files ... 876
28.10 Batch Convert Line Endings in Multiple Files ... 877
28.11 Batch Edit Multiple Files from File List ... 877
28.12 Binary File Editing Mode ... 878
28.13 Break Hard Links When Editing Files ... 878
28.14 Browse and Edit JAR Files Directly in Vim/Neovim ... 879
28.15 Browse for files with dialog ... 879
28.16 Bulk Create Directories from Text List ... 880
28.17 Bulk Open Files from Filename List ... 880
28.18 Bulk Rename Files Using Vim ... 881
28.19 Capture Change Locations ... 882
28.20 Central Backup Directory with Directory Creation ... 882
28.21 Check file existence in scripts ... 883
28.22 Compile LaTeX with Rubber and Make in Vim ... 883
28.23 Compress Vim Documentation Files ... 884
28.24 Convert File Line Endings Easily ... 884
28.25 Convert OEM to ANSI Character Encoding ... 885
28.26 Convert PDF to Text in Vim ... 885
28.27 Create Tar Archive of Open Buffers ... 886
28.28 Create Timestamped Temporary File Copy ... 886
28.29 Create Vim Patches Easily ... 887
28.30 Create a Personal Knowledge Base in Vim ... 887
28.31 Create and Restore Vim Sessions Easily ... 888
28.32 Create and Submit Vim Patches Easily ... 888
28.33 Cross-Platform File Directory Opener ... 889
28.34 Customize Netrw Directory Listing Style ... 890
28.35 Decompile and View Java Class Files in Vim ... 890
28.36 Delete Current File and Close Buffer ... 891
28.37 Delete Files Directly from Vim ... 891
28.38 Drag and Drop Files into Vim Command Line ... 892
28.39 Easy File Encoding Switching ... 892

28.40 Edit Files on FTP Server with Custom Port ... 893
28.41 Edit Files on Non-Standard Port FTP Server ... 893
28.42 Edit Remote Files Seamlessly with Netrw ... 894
28.43 Edit Remote Files via Network Protocols ... 894
28.44 Edit Remote Files via SCP ... 895
28.45 Edit Remote Files with SCP in Vim ... 895
28.46 Encrypt Files Securely in Vim ... 896
28.47 Ex commands - file permissions and attributes ... 896
28.48 Ex commands - read and write operations ... 897
28.49 Flexible File Find Function ... 897
28.50 Flexible Project File Search with Ignore Patterns ... 898
28.51 Flexible Vim File Launch Options ... 900
28.52 Generate C Source from Binary Files ... 900
28.53 Handle Filenames with Spaces in Path ... 901
28.54 Hex Dump Binary Files ... 901
28.55 Hex Dump Binary Files with xxd ... 902
28.56 Incremental Backups with Timestamped Files ... 902
28.57 Incremental File Backups with Unique Timestamps ... 903
28.58 Insert File or Command Output into Buffer ... 903
28.59 Insert Template Files into Buffer ... 904
28.60 Insert current date ... 905
28.61 Launch Files with Default Application ... 905
28.62 List Files in Current Directory Tree ... 905
28.63 Modifier Keys for Drag and Drop in Vim ... 907
28.64 Move Files Directly in Vim File Explorer ... 907
28.65 Open All Files in Tabs by Default ... 908
28.66 Open Binary Files in Vim ... 908
28.67 Open Current File's Directory in File Explorer ... 909
28.68 Open Current Visual Studio File in Vim ... 909
28.69 Open File at Specific Line ... 910
28.70 Open File with Clipboard Paste ... 910
28.71 Open File's Directory in Vim ... 911
28.72 Open Files in Existing Vim Instance ... 911
28.73 Open Files in New Tabs Across Desktop ... 912
28.74 Open Files in New Tabs Across Desktops ... 912
28.75 Open Files in New Tabs Automatically ... 913
28.76 Open Files in New Tabs via Remote Command ... 913
28.77 Open Files in Tabs Across Multiple Instances ... 914
28.78 Open Files in Tabs from Command Line ... 914
28.79 Open Files in Tabs with Path Search ... 915
28.80 Open Files via URLs in Vim ... 915

28.81 Open Files with Wildcard Patterns 916
28.82 Open Finder for macOS Users 916
28.83 Open Multiple Files from List File 916
28.84 Open Multiple Files in Same Vim Instance on macOS 917
28.85 Open Multiple Files in Tabs Automatically 917
28.86 Open Windows Explorer from Current Buffer 918
28.87 Open or Jump to File in New Tab 918
28.88 Optimize Large File Loading Performance 919
28.89 Path separator conversion 920
28.90 Preserve Missing End-of-Line When Saving Files 920
28.91 Preserve Modified Flag After Writing to File 922
28.92 Project File Search with Wildmenu 923
28.93 Protect Reference Files from Accidental Changes 924
28.94 Quick Backup of Current File with Timestamp 924
28.95 Quick Directory Exploration in Vim 925
28.96 Quick File Backup Mapping 926
28.97 Quick File Copy Within Vim 926
28.98 Quick File Encoding Detection in Bash 927
28.99 Quick File Extension Manipulation 927
28.100 Quick File and Editing Commands 928
28.101 Quick HTML File Preview in Browser 928
28.102 Quick Note Taking with Vim Alias 929
28.103 Quick Source/Header File Switching 929
28.104 Quick Vim Directory Navigation 930
28.105 Quickly Find and Edit Files Across Large Codebases 930
28.106 Quickly Switch Between Source and Header Files 931
28.107 Recover Files After Crash or Power Failure 931
28.108 Recover Overwritten File Using Swap File 932
28.109 Recursive File Finding in Vim 932
28.110 Reload File with Different Encoding 933
28.111 Reload file from disk 934
28.112 Restore Last Edited File and Cursor Position 934
28.113 Save and Restore Session Automatically 935
28.114 Save as 936
28.115 Save file 936
28.116 Save multiple files at once 937
28.117 Secure File Encryption in Vim 937
28.118 Secure GPG File Editing in Vim/Neovim 938
28.119 Set Working Directory to Current File 939
28.120 Sync Disk After Every Write to Prevent Data Loss 939
28.121 Sync Vim Runtime Files with rsync 940

28.122 Track File Open Dialog to Current File's Directory ... 940
28.123 Transparently Edit Encrypted GPG Files in Vim ... 941
28.124 Update file only if changed ... 942
28.125 Windows File Associations with Vim/Neovim ... 942
28.126 Write file and create all directories form the full file path ... 942
28.127 Zip Open Buffers Quickly ... 943

29 Filetype specific tips ... 945

29.1 Binary and hex file editing ... 945
29.2 C/C++ header and implementation switching ... 945
29.3 CSS and SCSS productivity shortcuts ... 946
29.4 Configuration file syntax highlighting ... 946
29.5 Detect Inform Header Files in Vim ... 947
29.6 Docker and container file editing ... 948
29.7 Git commit message formatting ... 948
29.8 Go language specific features ... 949
29.9 HTML and XML tag manipulation ... 949
29.10 JSON formatting and validation ... 950
29.11 Java class and package navigation ... 950
29.12 JavaScript/TypeScript development setup ... 951
29.13 Log file analysis and navigation ... 951
29.14 Lua script configuration ... 952
29.15 Markdown writing and formatting ... 952
29.16 Python indentation and formatting ... 953
29.17 Rust development optimization ... 953
29.18 SQL query formatting and execution ... 954
29.19 Shell script development ... 954
29.20 Template file creation ... 955
29.21 XML and configuration file handling ... 955
29.22 YAML configuration editing ... 956

30 Folding ... 957

30.1 Advanced Folding Techniques in Vim/Neovim ... 957
30.2 Advanced Syntax Folding for Java Files ... 958
30.3 Automatically Fold Perl Subroutines ... 958
30.4 Create fold from selection ... 959
30.5 Custom Fold Text for Comments ... 959
30.6 Custom Function Folding in C/C++ ... 960
30.7 Custom Java Folding with Expression Function ... 961
30.8 Custom Perl Folding with Fold Expression ... 961

30.9 Customize Fold Text with Advanced Display ... 963
30.10 Double-Click to Toggle Folds ... 964
30.11 Dynamic Folding for Java Code Blocks ... 964
30.12 Dynamic Indent Folding with Manual Override ... 965
30.13 Dynamic Indent and Manual Folding ... 966
30.14 Flexible Folding Methods in Neovim ... 966
30.15 Flexible Java Code Folding with Expression ... 967
30.16 Fold Away Lines Not in Quickfix List ... 968
30.17 Fold C# Regions Easily ... 969
30.18 Fold C-Style Comments Automatically ... 969
30.19 Fold C/C++ Functions Including Prototypes ... 970
30.20 Fold File Listings with Dynamic Hierarchy ... 970
30.21 Fold INI File Sections Automatically ... 972
30.22 Fold Lines Matching Search Pattern ... 972
30.23 Fold Multiple Empty Lines in a File ... 973
30.24 Fold Text by Regex Search Pattern ... 973
30.25 Fold VB.NET Code with Region Markers ... 974
30.26 Fold levels ... 975
30.27 Indent-Based Folding for Plain Text ... 976
30.28 MoinMoin Wiki Folding with Vim ... 976
30.29 Navigate XML Project Files with Folding ... 977
30.30 Navigate XML Project Hierarchies with Folds ... 978
30.31 Open All Folds Automatically on File Open ... 979
30.32 Open and close all folds ... 980
30.33 Prevent Syntax Folding Disruptions in Insert Mode ... 980
30.34 Quick Fold Marker Wrapping ... 981
30.35 Quick Java API Folding ... 982
30.36 Quick Java API View via Folding ... 982
30.37 Smart Folding for Diff Files ... 983
30.38 Toggle fold ... 984
30.39 Use Built-in Folding for Outlining ... 984
30.40 Use Vim as a Powerful Outline Processor ... 984
30.41 Use Vim-Style Folding in Documentation ... 985
30.42 VB.NET Style Folding in Vim/Neovim ... 986
30.43 XML Syntax Folding in Vim ... 987
30.44 Z-commands - create folds ... 987

31 Formatting ... 989

31.1 Add Line Numbers Using Perl Filter ... 989
31.2 Align Numbers at Decimal Point ... 989

31.3 Align Text into Columns ... 990
31.4 Align Text into Columns Automatically ... 990
31.5 Automatic Table Field Alignment ... 991
31.6 Automatic Text Wrapping for Plain Text ... 991
31.7 Automatically Indent XML Files ... 992
31.8 Chop Long Lines Using fmt Command ... 992
31.9 Clean Paragraphs for Copy-Paste to Word ... 993
31.10 Clean Up HTML Files with Tidy ... 994
31.11 Collapse Multi-Line Paragraphs ... 994
31.12 Comment lines by filetype ... 995
31.13 Convert CSV to Readable Columns ... 995
31.14 Convert Code to Syntax-Highlighted HTML ... 996
31.15 Convert Spaces to Tabs Efficiently ... 996
31.16 Environment-Aware Paragraph Formatting in LaTeX ... 997
31.17 Expand Tabs to Spaces in Entire File ... 997
31.18 Flexible Line Formatting ... 998
31.19 Format Code Blocks and Functions Easily ... 998
31.20 Format Only Long Lines in Text ... 999
31.21 Format Paragraph Without Moving Cursor ... 999
31.22 Format XML Files Using xmllint ... 1000
31.23 Format with Treesitter ... 1000
31.24 Poor men's JSON formatter ... 1000
31.25 Powerful Text Formatting with Par ... 1001
31.26 Pretty Format XML with External Tools ... 1002
31.27 Quick Line Formatting with gqq ... 1003
31.28 Quick XML Formatting with Python ... 1003
31.29 Quick XML Indentation Without XSLT ... 1004
31.30 Quick XML/HTML Indentation Trick ... 1004
31.31 Reformat Email Quotations Quickly ... 1005
31.32 Regex-Based Text Alignment Function ... 1005
31.33 Reindent Entire File or Selected Text ... 1006
31.34 Unicode Text Formatting Workaround ... 1007
31.35 Wrap Lines at 80 Chars Without Breaking Words ... 1007

32 Fun ... 1009

32.1 Flip a coin ... 1009
32.2 Funny event ... 1009
32.3 Help! ... 1009
32.4 Holy Grail ... 1010
32.5 Matrix like effect ... 1010

32.6 Random quote generator: 1010
32.7 Reverse lines in file 1010
32.8 Show all whitespace, but nicely 1011
32.9 Shuffle lines 1011
32.10 Smile! 1011
32.11 Sort randomly 1012
32.12 Speaking French? 1012
32.13 Surprise yourself! 1012
32.14 What is the meaning of life, the universe and everything 1012

33 G commands 1013
33.1 Change case with gu and gU 1013
33.2 Edit file under cursor with gf 1013
33.3 Format text with gq 1014
33.4 Format without moving cursor with gw 1014
33.5 Go to previous/next tab with gT and gt 1014
33.6 Increment numbers in sequence with g CTRL-A 1014
33.7 Insert at line start with gI 1015
33.8 Join lines without spaces with gJ 1015
33.9 Move to middle of screen line with gm 1015
33.10 Move to screen line positions with g0, g^, g$ 1016
33.11 Navigate screen lines with gj and gk 1016
33.12 Open URL or file with gx 1016
33.13 Repeat last command with g. 1017
33.14 Reselect last visual selection with gv 1017
33.15 Return to last insert position with gi 1017
33.16 Select last search match with gn 1017
33.17 Show ASCII value with ga 1018
33.18 Swap character case with g~ 1018

34 Global 1019
34.1 Global command - advanced patterns 1019
34.2 Global command - copy matching lines 1019
34.3 Global command - execute on matching lines 1019
34.4 Global command - move matching lines 1020
34.5 Global command with line ranges 1020
34.6 Global command with normal mode commands 1020
34.7 Open documentation for word under the cursor 1021
34.8 Open terminal 1021

35 Help 1023

35.1 Ex commands - help and documentation 1023
35.2 Ex commands - help navigation 1023
35.3 Man pages 1023
35.4 Master help index 1024
35.5 Search help by pattern 1024

36 Indentation 1025

36.1 Adjust Indent in Insert Mode 1025
36.2 Auto indent 1025
36.3 Block Indent with Matching Brace 1026
36.4 Block Indentation with Brace Matching 1026
36.5 Correct Indentation in Blank Lines 1027
36.6 Disable Auto-Indentation for Current File 1027
36.7 Disable Auto-Indentation for Specific File Types 1027
36.8 Easy Indentation with Cursor Position Preserved 1028
36.9 Flexible Indentation and Tab Settings 1029
36.10 Improve PHP and HTML Indentation Support 1029
36.11 Increment/Decrement Block Indentation 1030
36.12 Indent with Tabs, Align with Spaces 1031
36.13 Preserve Indent Across Blank Lines 1032
36.14 Preserve Indentation When Typing Hash 1033
36.15 Quick Block Indentation Commands 1034
36.16 Quickly Reindent Code Blocks 1034
36.17 Quickly Reindent Entire File or Selected Text 1035
36.18 Reindent Code Blocks Efficiently 1036
36.19 Select and Manipulate Code Blocks by Indentation 1036
36.20 Smart Dedent to Previous Indent Level 1037
36.21 Smart Tabs for Consistent Indentation 1039
36.22 Use Filetype-Based Indentation 1039

37 Insert 1041

37.1 Adjust indentation in insert mode 1041
37.2 Control undo granularity in insert mode 1041
37.3 Copy character from line above/below 1041
37.4 Exit insert mode alternatives 1042
37.5 Insert above cursor 1042
37.6 Insert calculation result 1042
37.7 Insert character by decimal value 1043

37.8 Insert digraphs ... 1043
37.9 Insert mode completion ... 1043
37.10 Insert mode completion subcommands ... 1044
37.11 Insert mode cursor movement with insertion point ... 1044
37.12 Insert mode line break ... 1044
37.13 Insert tab character alternatives ... 1045
37.14 New line insertion ... 1045
37.15 Paste in insert mode ... 1045
37.16 Paste in insert mode with register ... 1045
37.17 Repeat last inserted text ... 1046
37.18 Replace mode ... 1046
37.19 Scroll window in insert mode ... 1046
37.20 Trigger abbreviation manually ... 1047

38 Insert mode (advanced) ... 1049

38.1 Advanced completion modes ... 1049
38.2 Completion menu navigation ... 1049
38.3 Delete operations in insert mode ... 1050
38.4 Insert mode abbreviation control ... 1050
38.5 Insert mode buffer operations ... 1050
38.6 Insert mode case conversion ... 1051
38.7 Insert mode folding control ... 1051
38.8 Insert mode formatting and alignment ... 1051
38.9 Insert mode macro operations ... 1052
38.10 Insert mode marks and jumps ... 1052
38.11 Insert mode navigation without exiting ... 1052
38.12 Insert mode register shortcuts ... 1053
38.13 Insert mode search operations ... 1053
38.14 Insert mode terminal integration ... 1053
38.15 Insert mode text objects ... 1054
38.16 Insert mode window operations ... 1054
38.17 Literal character insertion ... 1054
38.18 Smart indentation in insert mode ... 1055
38.19 Temporary normal mode from insert mode ... 1055
38.20 Word movement in insert mode ... 1055

39 Integration tips ... 1057

39.1 API and webhook integration ... 1057
39.2 Browser and documentation integration ... 1057
39.3 Build system integration ... 1058

39.4 Cloud platform integration ... 1058
39.5 Continuous Integration integration ... 1059
39.6 Database integration and querying ... 1059
39.7 Development server integration ... 1060
39.8 Docker and container integration ... 1060
39.9 Documentation generation integration ... 1061
39.10 Email and notification integration ... 1061
39.11 External editor integration ... 1062
39.12 Git integration and workflow ... 1062
39.13 Git integration with gitsigns plugin ... 1063
39.14 Issue tracking integration ... 1063
39.15 Launch Vim Inside Visual Studio ... 1064
39.16 Monitoring and logging integration ... 1064
39.17 Open Files in gVim from Source Navigator ... 1065
39.18 Package manager integration ... 1065
39.19 REST API testing integration ... 1065
39.20 SSH and remote development integration ... 1066
39.21 System clipboard integration ... 1066
39.22 Terminal multiplexer integration ... 1067
39.23 Testing framework integration ... 1067
39.24 Version control system integration ... 1068
39.25 Visual Studio .NET Vim Integration ... 1068

40 Lsp ... 1071
40.1 LSP code actions ... 1071
40.2 LSP format document ... 1071
40.3 LSP implementation ... 1071
40.4 LSP incoming calls ... 1072
40.5 LSP list workspace folders ... 1072
40.6 LSP remove workspace folder ... 1072
40.7 LSP rename ... 1072
40.8 LSP show signature help ... 1073
40.9 Toggle LSP inlay hints ... 1073

41 Lua ... 1075
41.1 Access Neovim API functions ... 1075
41.2 Access environment variables in Lua ... 1075
41.3 Call Lua functions from Vimscript ... 1076
41.4 Call Vimscript functions from Lua ... 1076
41.5 Check if list is empty ... 1076

41.6 Check if table contains key ... 1077
41.7 Choose init.vim or init.lua ... 1077
41.8 Create abstract base classes with error checking ... 1077
41.9 Create autocommand groups ... 1079
41.10 Create autocommands with nvim_create_autocmd ... 1079
41.11 Create buffer-local keymaps ... 1079
41.12 Create classes with metatables ... 1080
41.13 Create keymaps with vim.keymap.set ... 1081
41.14 Create private state with closures ... 1081
41.15 Create singleton pattern with metatables ... 1082
41.16 Create user commands in Lua ... 1083
41.17 Debug Lua values ... 1084
41.18 Deep merge tables with vim.tbl_deep_extend ... 1084
41.19 Defer execution with vim.schedule ... 1085
41.20 Defer function execution ... 1085
41.21 Execute Lua from command line ... 1086
41.22 Execute Vim commands from Lua ... 1086
41.23 Execute normal mode commands from Lua ... 1086
41.24 Filter and map tables ... 1087
41.25 Format buffer with LSP ... 1087
41.26 Get LSP clients for buffer ... 1088
41.27 Get current mode in Lua ... 1088
41.28 Implement __tostring metamethod ... 1088
41.29 Implement class inheritance with metatables ... 1089
41.30 Implement operator overloading with metamethods ... 1090
41.31 Load Lua modules with require ... 1092
41.32 Lua keymaps ... 1092
41.33 Method chaining with metatables ... 1092
41.34 Module structure best practices ... 1094
41.35 Plugin configuration with metatables ... 1094
41.36 Reload Lua modules during development ... 1095
41.37 Run current Lua file ... 1096
41.38 Run inline Lua code ... 1096
41.39 Safely load modules with pcall ... 1096
41.40 Schedule callbacks with vim.schedule ... 1097
41.41 Set buffer-local options ... 1097
41.42 Set buffer-local variables ... 1097
41.43 Set cursor position in Lua ... 1098
41.44 Set global variables in Lua ... 1098
41.45 Set options with vim.o ... 1098
41.46 Set options with vim.opt ... 1099

41.47 Set tabpage-local variables ... 1099
41.48 Set window-local options ... 1099
41.49 Set window-local variables ... 1100
41.50 Split and join strings ... 1100
41.51 Trim whitespace from strings ... 1100
41.52 Use Lua heredoc in Vimscript ... 1101
41.53 Use vim.loop for async operations ... 1101
41.54 Use vim.notify for notifications ... 1102
41.55 Use vim.ui.input for user input ... 1102
41.56 Use vim.ui.select for user selection ... 1102
41.57 Validate function arguments ... 1103
41.58 View loaded Lua modules ... 1103
41.59 Work with buffer text in Lua ... 1103
41.60 Work with quickfix list in Lua ... 1104

42 Macros ... 1105

42.1 Edit macro in command line ... 1105
42.2 Execute macro ... 1105
42.3 Macro for data transformation ... 1105
42.4 Make existing macro recursive ... 1106
42.5 Quick macro shortcuts ... 1106
42.6 Record recursive macro by including the self-reference ... 1107
42.7 Record recursive macro that calls itself until a condition is met ... 1107
42.8 Run macro on multiple files ... 1107
42.9 Run macro over visual selection ... 1108
42.10 Save macro in vimrc ... 1108
42.11 View macro contents ... 1108

43 Marks ... 1109

43.1 Cut or Copy Lines Without Line Counting ... 1109
43.2 Master Vim Marks for Quick Navigation ... 1109
43.3 Navigate Between Marks Efficiently ... 1110
43.4 Navigate Between Marks Quickly ... 1111
43.5 Quick File Bookmarking with Context ... 1111
43.6 Remember Last Position in Help Files ... 1112
43.7 Use Marks for Quick Navigation ... 1112
43.8 Use Special Marks for Quick Reference ... 1113
43.9 Useful Special Marks for Quick Navigation ... 1114

44 Modern neovim api 1115

44.1 Advanced autocommand patterns and groups 1115
44.2 Buffer-local configurations and mappings 1115
44.3 Create floating windows with Neovim API 1116
44.4 Custom diagnostic configuration 1117
44.5 Custom user commands with completion 1117

45 Movement 1119

45.1 Alternative movement keys 1119
45.2 Basic cursor movement 1119
45.3 Center cursor on screen 1119
45.4 Change list navigation 1120
45.5 Character search on line 1120
45.6 Document navigation 1120
45.7 Jump list navigation 1121
45.8 Jump multiple lines with arrow keys 1121
45.9 Jump to definition 1121
45.10 Jump to specific line 1122
45.11 Last non-blank character motion (g_) 1122
45.12 Line navigation 1122
45.13 Line number movement 1122
45.14 Matching brackets 1123
45.15 Middle of screen 1123
45.16 Page movement 1123
45.17 Page scrolling with cursor positioning 1124
45.18 Paragraph movement 1124
45.19 Repeat character search 1124
45.20 Screen position navigation 1125
45.21 Screen scrolling 1125
45.22 Sentence movement 1125
45.23 Suspend and background 1125
45.24 Word movement 1126
45.25 Word movement alternatives 1126
45.26 Z-commands - horizontal scrolling 1126
45.27 Z-commands - redraw with cursor positioning 1127
45.28 Z-commands - window height adjustment 1127

46 Navigation 1129

46.1 Advanced Cursor Jumping Techniques 1129

46.2 Advanced File and Buffer Navigation 1129
46.3 Advanced Find and Till Commands 1130
46.4 Advanced Movement and Modification Markers 1130
46.5 Advanced Search and Movement 1131
46.6 Advanced Tab Navigation Shortcuts 1131
46.7 Advanced Tag Search and Browsing 1132
46.8 Alternative Help Navigation Mapping 1133
46.9 Alternative Tag Navigation 1133
46.10 Buffer switching shortcuts 1133
46.11 Center Cursor on Movement 1134
46.12 Center Screen on Search Navigation 1134
46.13 Configure Matchit for ColdFusion Tag Jumping 1135
46.14 Context-Sensitive H/L Movement 1135
46.15 Create Custom Help-Style Tags in Source Code 1136
46.16 Create Custom Tags with Line and Column Precision 1137
46.17 Dynamically Build Tags from Path 1138
46.18 Easy Help Navigation on AZERTY Keyboards 1139
46.19 Easy Help Tag Navigation for German Keyboards 1139
46.20 Efficient Cscope Navigation in Vim/Neovim 1140
46.21 Efficient Tab Navigation Shortcuts 1141
46.22 Efficient Tab Navigation and Management 1142
46.23 Efficient Tab Page Navigation 1142
46.24 Efficient Tag Navigation in Vim/Neovim 1143
46.25 Efficient Text Navigation Commands 1143
46.26 Efficient Text Navigation and Manipulation 1144
46.27 Efficient Vim Navigation and Editing Basics 1144
46.28 Enable Caret Browsing for Vim-Like Text Navigation 1145
46.29 Enhanced Bracket and Brace Navigation 1145
46.30 Enhanced Matching with % Key 1146
46.31 Enhanced Word Search with Smart Matching 1146
46.32 Enhanced navigation with Flash.nvim 1147
46.33 Fast Jump to PHP Variables on a Line 1148
46.34 Fast Window Scrolling with Alt Key Mappings 1148
46.35 Fast buffer access 1149
46.36 Find Current Verilog Module Name 1149
46.37 Flexible File Navigation 1150
46.38 Flexible Python Test File Toggle 1150
46.39 Flexible Tag Search and Navigation 1151
46.40 Function Definition Search 1152
46.41 Fuzzy Search Multiple Vim Contexts 1152
46.42 Fuzzy Search Tags with Regular Expressions 1153

46.43 Generate Tags for Current File ... 1153
46.44 Generate and Use Custom Tag Files ... 1154
46.45 Go to declaration ... 1154
46.46 Highlight and Navigate CSV Columns Easily ... 1155
46.47 Highlight and Navigate CSV Columns Efficiently ... 1156
46.48 Improve Last Position Jump for Easy Vim ... 1156
46.49 Inline Java Documentation Browsing ... 1157
46.50 Interactive Jump List Selection ... 1157
46.51 Jump Between C++ Methods Quickly ... 1158
46.52 Jump between functions ... 1159
46.53 Jump to File from Diff View ... 1159
46.54 Jump to Lines with Same/Lower Indentation ... 1160
46.55 Jump to Local and Global Definitions Easily ... 1162
46.56 Jump to Matching Braces and Code Blocks ... 1162
46.57 Jump to Matching Braces and Syntax Elements ... 1163
46.58 Jump to PDF Page from LaTeX File ... 1163
46.59 Jump to Random Line in Buffer ... 1164
46.60 Jump to Variable Declaration ... 1165
46.61 Jump to a Random Line in Buffer ... 1165
46.62 Jump to block boundaries ... 1166
46.63 Keyboard-Driven Web Browsers with Vim Keybindings ... 1166
46.64 LSP go to references ... 1167
46.65 Master Vim Movement Shortcuts ... 1167
46.66 Move Between Words in Insert Mode ... 1167
46.67 Move Cursor by Display Lines in Wrapped Text ... 1168
46.68 Move Word Under Cursor with Alt-Arrows ... 1168
46.69 Multiple Ways to Navigate to a Specific Line ... 1169
46.70 Navigate File Change History ... 1170
46.71 Navigate Help Labels with Tab Keys ... 1170
46.72 Navigate Jump List Efficiently ... 1171
46.73 Navigate Jump List with Keyboard Shortcuts ... 1171
46.74 Navigate LaTeX Sections Efficiently ... 1172
46.75 Navigate Lines with Same Indentation ... 1173
46.76 Navigate Vim Help on AZERTY Keyboards ... 1174
46.77 Navigate Wrapped Lines Smoothly ... 1174
46.78 Navigate to Function Start and Back ... 1175
46.79 Navigate to Next/Previous Open Fold ... 1175
46.80 Navigate to alternate file ... 1176
46.81 Open Files Under Cursor Quickly ... 1177
46.82 Open Files Under Cursor in Different Windows ... 1177
46.83 Open Files Under Cursor with Multiple Methods ... 1178

46.84 Open Files Using Filename Under Cursor 1178
46.85 Open Files with Line Numbers 1179
46.86 Open Man Pages for Word Under Cursor 1180
46.87 Page Up/Down While Keeping Cursor Position 1180
46.88 Position Line at Screen Top/Middle/Bottom 1181
46.89 Powerful Tag Navigation in Vim/Neovim 1182
46.90 Powerful Text Editing with Vim Motions 1183
46.91 Quick Change to Current File's Directory 1183
46.92 Quick Entity Definition Jump in DTDs 1184
46.93 Quick Indentation Level Navigation 1184
46.94 Quick Jump to Function Definition 1185
46.95 Quick Jump to Keyword Occurrences 1185
46.96 Quick Man Page Lookup for Word Under Cursor 1187
46.97 Quick Navigation Between C++ Methods 1187
46.98 Quick Navigation in Help Buffers 1187
46.99 Quick Navigation to Current Day in Calendar 1188
46.100 Quick Navigation with Word Labels 1189
46.101 Quick Open Perl Module Source 1190
46.102 Quick PHP Variable Navigation 1191
46.103 Quick Python Module-Test File Switching 1191
46.104 Quick Recovery When Spell Checking 1192
46.105 Quick Reference Documents in Tabs 1192
46.106 Quick Tab Navigation and Management 1193
46.107 Quickfix Window Navigation Mappings 1194
46.108 Quickly Get Current Function Name 1194
46.109 Quickly Open Perl Module Source 1195
46.110 Quickly Search Vim Help Topics 1195
46.111 Random Line Selection with Range Support 1196
46.112 Read Javadoc Directly in Vim 1197
46.113 Recursive File Finding with Path Option 1197
46.114 Recursive File Search with Fuzzy Matching 1198
46.115 Recursive Project File Finding 1198
46.116 Screen-Based Movement Commands 1198
46.117 Scroll Without Moving Cursor 1199
46.118 Smart Middle of Line Movement 1200
46.119 Square bracket navigation - C comments 1200
46.120 Square bracket navigation - changes and diffs 1201
46.121 Square bracket navigation - definitions and includes 1201
46.122 Square bracket navigation - folds 1201
46.123 Square bracket navigation - list definitions 1202
46.124 Square bracket navigation - member functions 1202

46.125 Square bracket navigation - preprocessing ... 1202
46.126 Square bracket navigation - sections ... 1203
46.127 Square bracket navigation - show definitions ... 1203
46.128 Square bracket navigation - spelling ... 1203
46.129 Square bracket navigation - unmatched brackets ... 1204
46.130 Toggle netrw file explorer ... 1204
46.131 Track and Search Recently Visited Directories ... 1204
46.132 Use Alt/Meta Keys for Quick Navigation ... 1205
46.133 Use Alt/Meta Keys in Terminal Vim ... 1206
46.134 Use Environment Variables for Dynamic File Paths ... 1206
46.135 Use K to View PHP Function Help ... 1207
46.136 Use g; and g, for Edit Location Navigation ... 1207
46.137 Use g_ to Navigate End of Line in Langmap ... 1207
46.138 View Man Pages Directly in Vim/Neovim ... 1208
46.139 View Multiple Man Page Chapters ... 1208
46.140 View jump list ... 1209
46.141 Vim-Style Browser Navigation Extensions ... 1209
46.142 Windows-like Ctrl+Left/Right Word Navigation ... 1209
46.143 Wrap Cursor Movement Between Lines ... 1210

47 Neovim features ... 1211

47.1 Auto commands with Lua ... 1211
47.2 Built-in snippet support ... 1211
47.3 Built-in terminal ... 1212
47.4 Diagnostic API ... 1212
47.5 Extended marks ... 1212
47.6 Floating windows API ... 1213
47.7 Health checks ... 1213
47.8 Lua configuration ... 1213
47.9 Multiple cursors simulation ... 1214
47.10 Quick fix navigation ... 1214
47.11 RPC and job control (jobstart) ... 1214
47.12 Statusline and tabline API ... 1215
47.13 Tree-sitter text objects ... 1215
47.14 User commands ... 1215
47.15 Virtual text ... 1216

48 Neovim terminal ... 1217

48.1 Hidden terminal processes ... 1217
48.2 Split terminal workflows ... 1217

48.3 Terminal REPL workflows ... 1217
48.4 Terminal and quickfix integration ... 1218
48.5 Terminal autocmd events ... 1218
48.6 Terminal buffer job control ... 1218
48.7 Terminal buffer naming ... 1219
48.8 Terminal color and appearance ... 1219
48.9 Terminal debugging integration ... 1219
48.10 Terminal environment variables ... 1220
48.11 Terminal mode key mappings ... 1220
48.12 Terminal output processing ... 1220
48.13 Terminal plugin integration ... 1221
48.14 Terminal process communication ... 1221
48.15 Terminal scrollback and history ... 1221
48.16 Terminal session persistence ... 1222
48.17 Terminal size and dimensions ... 1222
48.18 Terminal window management ... 1222
48.19 Terminal with specific shell ... 1223
48.20 Terminal with working directory ... 1223

49 Normal mode (advanced) ... 1225
49.1 Buffer navigation shortcuts ... 1225
49.2 Case conversion commands ... 1225
49.3 Change case of text ... 1225
49.4 Change operations ... 1226
49.5 Completion in insert mode trigger ... 1226
49.6 Delete characters and words ... 1226
49.7 Digraph insertion ... 1227
49.8 Ex mode and command execution ... 1227
49.9 File under cursor operations ... 1227
49.10 Filter through external command ... 1228
49.11 Fold operations ... 1228
49.12 Format text ... 1228
49.13 Go to column ... 1229
49.14 Increment and decrement numbers ... 1229
49.15 Indent and outdent ... 1229
49.16 Insert at line ends/beginnings ... 1229
49.17 Join lines with space control ... 1230
49.18 Line completion and duplication ... 1230
49.19 Mark commands ... 1230
49.20 Open new lines ... 1231

49.21 Put operations ... 1231
49.22 Record and replay macros ... 1231
49.23 Repeat last command ... 1232
49.24 Replace single character ... 1232
49.25 Search under cursor ... 1232
49.26 Spelling navigation ... 1233
49.27 Tag navigation ... 1233
49.28 Undo and redo ... 1233
49.29 Visual selection commands ... 1234
49.30 Window navigation ... 1234
49.31 Yank operations ... 1234

50 Performance ... 1235

50.1 Disable unused features ... 1235
50.2 Lazy load plugins ... 1235
50.3 Lazy-load plugins for faster startup ... 1236
50.4 Memory usage monitoring ... 1236
50.5 Optimize file type detection ... 1237
50.6 Optimize line numbers ... 1237
50.7 Optimize updatetime ... 1237
50.8 Profile Lua code ... 1238
50.9 Profile startup time ... 1238
50.10 Reduce redraw frequency ... 1238
50.11 Syntax highlighting limits ... 1239
50.12 Use swap files efficiently ... 1239

51 Performance (advanced) ... 1241

51.1 Autocommand optimization ... 1241
51.2 Buffer and window optimization ... 1242
51.3 Completion system optimization ... 1242
51.4 Concurrent operations optimization ... 1243
51.5 Diff and merge performance optimization ... 1243
51.6 Display and rendering optimization ... 1244
51.7 File I/O optimization ... 1245
51.8 LSP performance optimization ... 1245
51.9 Large file handling optimization ... 1246
51.10 Memory management and garbage collection ... 1246
51.11 Network and remote file optimization ... 1247
51.12 Optimize plugin loading strategy ... 1248
51.13 Plugin configuration caching ... 1248

51.14 Search and regex performance tuning 1249
51.15 Startup time profiling and analysis 1249
51.16 Syntax and highlighting optimization 1250

52 Plugins 1253
52.1 Add Version Info to Plugins 1253
52.2 Advanced File and Buffer Selection Plugins 1253
52.3 Advanced Matching with matchit Plugin 1254
52.4 Advanced UI Selection Plugins for Vim/Neovim 1254
52.5 Asynchronous Linting with ALE 1255
52.6 Auto-Generate Project Tags with Makefile 1255
52.7 Auto-Source Plugins in Vim 1256
52.8 Automatically Generate PHPDoc Comments 1256
52.9 C++ Code Completion with OmniCppComplete 1257
52.10 Configure SVN Integration with VCSCommand 1258
52.11 Configure TagList for ANT Build File Navigation 1258
52.12 Create Custom Calendar with Special Dates 1259
52.13 Create Custom Commands with Autocompletion 1260
52.14 Cross-Platform LaTeX Editing Plugins 1260
52.15 Debug Mode for Plugin Development 1261
52.16 Dynamic Plugin Version Selection 1261
52.17 Easily Create LaTeX Tables with Vim 1262
52.18 Easily Extend Vim Functionality with Plugins 1262
52.19 Flexible Ctags Generation for Multi-Directory Projects 1263
52.20 Generate Ctags for C/C++ Project with Dependencies 1264
52.21 Initialize Plugins Using VimEnter Autocmd 1264
52.22 Integrate Python Linters in Vim/Neovim 1265
52.23 LaTeX Backward Search Across Platforms 1266
52.24 LaTeX Backward Search Setup 1267
52.25 Manage Plugins with Plugin Managers 1267
52.26 Native Vim Plugin Management 1268
52.27 Optimize Startup by Delaying Plugin Loading 1268
52.28 Plugin Initialization in After Plugin Directory 1268
52.29 Plugin Manager Alternatives 1269
52.30 Reload Plugin During Development 1269
52.31 Simplify Omni-Completion with SuperTab 1270
52.32 Use Compiler Plugins for Linting 1270
52.33 Use External Editor with Thunderbird 1271
52.34 Use Vim as External Editor for Web Textareas 1271

53 Registers 1273
53.1 Advanced Copy and Paste with Named Registers 1273
53.2 Advanced Text Manipulation with Registers 1273
53.3 Append Multiple Lines to a Register 1274
53.4 Append or Modify Existing Macros 1274
53.5 Append to Existing Macros 1275
53.6 Append to Macro Registers Safely 1275
53.7 Clear specific register 1276
53.8 Create Recursive Macros for Bulk Text Manipulation 1276
53.9 Cycle Through Numbered Registers 1277
53.10 Cycle Through Vim Registers Easily 1277
53.11 Delete without affecting register 1278
53.12 Easy Word Replacement Using Registers 1278
53.13 Edit and Modify Recorded Macros 1278
53.14 Execute Macro Across Visual Block Selection 1279
53.15 Get Current File Name in Different Formats 1280
53.16 Get Current Filename in Vim 1280
53.17 Insert Register Content Across Multiple Lines 1281
53.18 Paste Over Text Without Losing Register 1281
53.19 Paste Registers in Command and Search Lines 1282
53.20 Paste Text as Separate Line with Easy Mapping 1282
53.21 Paste from Clipboard with Precise Cursor Placement 1283
53.22 Paste from Different Vim Registers Safely 1283
53.23 Preload Vim Registers with Frequently Used Commands 1284
53.24 Prevent Clipboard Hijacking in Vim 1284
53.25 Quick Macro Recording and Playback 1285
53.26 Record and Execute Vim Macros Efficiently 1285
53.27 Record and Replay Macros Efficiently 1286
53.28 Record and Replay Powerful Text Transformation Macros 1287
53.29 Record, Edit, and Reuse Vim Macros 1287
53.30 Recover Previous Deletes Using Numbered Registers 1288
53.31 Recover Recently Deleted Text in Insert Mode 1289
53.32 Redirect Ex Command Output to Register 1289
53.33 Reorder Lines Using Registers 1289
53.34 Run Macro Across Visual Selection 1290
53.35 Set register manually 1290
53.36 Smart Visual Mode Paste without Register Overwrite 1291
53.37 System clipboard 1291
53.38 Translate Null Characters in Vim Registers 1291
53.39 View registers 1292

54 Scripting 1293

54.1 Handle UTF-8 Encoding in Python Vim Scripts 1293
54.2 Prevent Script Reloading with Function 1293

55 Search 1295

55.1 Abbreviated HTML Attribute Quoting 1295
55.2 Advanced Duplicate Line Removal 1295
55.3 Advanced Global Command Line Filtering 1296
55.4 Advanced Negative Lookahead Searches 1296
55.5 Advanced Pattern Matching with Global Command 1296
55.6 Advanced Range Selection Techniques 1297
55.7 Advanced Search and Replace with Confirmation 1297
55.8 Advanced Search with Case Sensitivity 1298
55.9 Advanced Search with Case Sensitivity Controls 1298
55.10 Advanced Search with OR Patterns 1299
55.11 Advanced Substitution with Capture Groups 1299
55.12 Advanced Word Highlight Search Navigation 1300
55.13 Advanced Word Search Mappings 1300
55.14 Advanced Word Searching with Regular Expressions 1301
55.15 Advanced search and replace with regex 1301
55.16 Alternative Delimiters for Substitution 1302
55.17 Avoid escaping slashes in search and replace operations 1302
55.18 Block-wise Visual Substitution Without Plugin 1302
55.19 CSV Column Search and Manipulation 1303
55.20 CSV Search Within Specific Column 1304
55.21 Case-Sensitive and Insensitive Search Replace 1304
55.22 Change Case Using Regex Substitutions 1305
55.23 Comment Lines Matching a Pattern 1305
55.24 Compress Multiple Blank Lines 1306
55.25 Convenient Search and Highlight Clearing 1306
55.26 Convert C++ Comments to C Comments 1307
55.27 Convert Decimal Encoded Characters 1307
55.28 Convert HTML Attributes to Lowercase 1308
55.29 Convert HTML Tags to Lowercase 1308
55.30 Copy Search Matches Efficiently 1308
55.31 Copy Search Matches to Clipboard 1309
55.32 Count Pattern Matches in Buffer 1310
55.33 Create Outline Using Vimgrep 1310
55.34 Cross-Buffer Text Replacement via Substitute 1311
55.35 Delete Lines Containing Search Matches 1311

55.36 Delete Lines with Selective Filtering 1312
55.37 Dynamic Highlight Search and Navigation 1312
55.38 Dynamic Number Substitution 1313
55.39 Exclude Lines Containing Specific Pattern 1314
55.40 Find Whole Words in Search 1314
55.41 Find and Remove Duplicate Words 1315
55.42 Flexible CSV Column Search and Manipulation 1315
55.43 Flexible Keyword Search and Jump 1316
55.44 Flexible Multi-Word Search 1317
55.45 Flexible Multi-Word Search and Replace 1317
55.46 Flexible Multiple Word Search 1318
55.47 Flexible Search and Replace Delimiters 1318
55.48 Flexible Word Search Under Cursor 1319
55.49 Global Command for Powerful Text Manipulation 1319
55.50 Global search and replace 1320
55.51 Grep Search Across Files with Quickfix Navigation 1320
55.52 Hex/Decimal Conversion Across Text 1321
55.53 Highlight Current Word Without Moving 1322
55.54 Highlight Only Current Search Result 1322
55.55 Highlight Search Pattern in Line Listing 1323
55.56 Highlight and Remove Unwanted Whitespace 1324
55.57 Incremental Search with Live Preview 1325
55.58 Insert Newlines Around Patterns Flexibly 1325
55.59 Multi-line search pattern 1327
55.60 Operate on Search Matches in Vim 7.4+ 1327
55.61 Perform Operation on Matching Lines 1327
55.62 Powerful Multi-File Search and Replace 1328
55.63 Precise Word Replacement 1328
55.64 Quick Count of Word Under Cursor 1329
55.65 Quick HTML Attribute Quoting 1329
55.66 Quick Multi-Line Text Substitution 1330
55.67 Quick Search for Word Under Cursor 1330
55.68 Quick Text Replacement Using Command Line 1330
55.69 Quick Toggle to Remove Search Highlights 1331
55.70 Quick Word Search in Current File 1331
55.71 Quickly Filter and List Search Matches 1332
55.72 Recursive File Search with Vim 1332
55.73 Redirect Search Results to a New Window 1333
55.74 Regex Lookahead and Lookbehind 1333
55.75 Remove Duplicate Lines Efficiently 1334
55.76 Remove File Path Prefix with Advanced Search/Replace 1334

55.77 Remove Multiple Path Levels in File Lists ... 1335
55.78 Remove search highlighting ... 1335
55.79 Repeat Substitution from Cursor Position ... 1335
55.80 Repeat last search in substitution ... 1336
55.81 Replace Spaces with Tabs Globally ... 1337
55.82 Replace Word Under Cursor Quickly ... 1337
55.83 Search Only Within Visually Selected Range ... 1338
55.84 Search Paths Without Escaping Slashes ... 1339
55.85 Search and Find Lines Longer Than 80 Characters ... 1339
55.86 Search and replace in multiple files (vimgrep+cdo) ... 1340
55.87 Search backward ... 1340
55.88 Search in selection ... 1340
55.89 Search with offset ... 1341
55.90 Search word boundaries with very magic ... 1341
55.91 Sequential Line Numbering with Counter ... 1341
55.92 Simple Regex Substitution with Capture Groups ... 1342
55.93 Smart Case Sensitivity in Search ... 1342
55.94 Smart Case-Sensitive Search ... 1343
55.95 Sort Lines by Selected Substring ... 1343
55.96 Speed Up Recursive Searches ... 1344
55.97 Swap Adjacent Words with Regex ... 1344
55.98 Title Case Conversion with Substitute ... 1345
55.99 Toggle Highlighting for Current Word ... 1345
55.100 Toggle Search Highlight with Single Keypress ... 1346
55.101 Toggle Search Highlighting Easily ... 1346
55.102 Use Alternate Delimiters in Search/Replace ... 1347
55.103 Use Expressions in Substitute Commands ... 1347
55.104 Use Perl-Style Regex in Vim Substitutions ... 1348
55.105 Use Very-Magic Regex Mode ... 1349
55.106 Word Replacement with Confirmation ... 1349
55.107 Zero-Width Matching with \zs and \ze ... 1350

56 Session ... 1351

56.1 Ex commands - arglist and project files ... 1351
56.2 Ex commands - session options ... 1351
56.3 Ex commands - viminfo and shada ... 1352
56.4 Ex commands - working with multiple files ... 1352
56.5 Save and Restore Vim Session ... 1352
56.6 Session management ... 1353

57 Syntax 1355

57.1 Custom Syntax Highlighting for MPI/PVM Functions 1355
57.2 Enhanced Heredoc Highlighting for INI Files 1356
57.3 Highlight C++ Method Names in Definitions 1356
57.4 Highlight Duplicate Words in Text 1357
57.5 Highlight Python Syntax Errors 1357
57.6 Python Triple-Quoted Comments Syntax Highlighting 1358

58 System 1359

58.1 Async shell commands 1359
58.2 Confirm dangerous operations 1359
58.3 Ex commands - external command execution 1359
58.4 Ex commands - file system operations 1360
58.5 Ex commands - make and quickfix 1360
58.6 Ex commands - shell and environment 1360
58.7 Execute line as command 1361
58.8 Read command output into buffer 1361
58.9 Redirect command output 1361
58.10 Write buffer to command 1362

59 Tabs 1363

59.1 Close tab 1363
59.2 Navigate tabs 1363
59.3 Open commands in new tabs 1363
59.4 Open new tab 1364

60 Terminal 1365

60.1 Capture SQL Query Output in Vim 1365
60.2 Check Existing Vim Server Instances 1365
60.3 Create Custom Terminal Vim Wrapper Script 1366
60.4 Dynamically Rename Konsole Tab in Vim 1366
60.5 Enable Vim Server in Terminal 1367
60.6 Integrate PyUnit Testing in Vim/Neovim 1367
60.7 Interactive Database Querying with Perl in Vim 1368
60.8 Keep Terminal Open After Vim Exit 1369
60.9 Open terminal in current window 1369
60.10 Open terminal in new window 1369
60.11 Quick IPython Integration with Vim 1370
60.12 Quick Python Code Execution in Vim 1370

60.13 Quick SQL Query Execution in Vim ... 1371
60.14 Quick Shell Command Switching with Ctrl-Z ... 1371
60.15 Quick Terminal Window Maximize/Restore ... 1372
60.16 Quick Version Control Commands in Vim ... 1372
60.17 Run External Commands Asynchronously in Windows ... 1373
60.18 Run MATLAB Scripts from Vim ... 1373
60.19 Run Shell Commands in Vim Scratch Buffer ... 1374
60.20 Screen Session Vim Integration ... 1375
60.21 Send commands to terminal ... 1375
60.22 Smart Vim Server Function for Shell ... 1376
60.23 Terminal insert mode ... 1376
60.24 Terminal mode - execute one command ... 1377
60.25 Terminal mode - exit to normal mode ... 1377
60.26 Terminal mode - key forwarding ... 1377
60.27 Terminal scrollback buffer ... 1378
60.28 tmux-Compatible Cursor Configuration ... 1378

61 Text manipulation ... 1381
61.1 Align numbers at decimal point ... 1381
61.2 Binary number operations ... 1381
61.3 Comment and uncomment blocks ... 1381
61.4 Convert tabs to spaces ... 1382
61.5 Create incremental sequence with g Ctrl+a ... 1382
61.6 Delete blank lines ... 1382
61.7 Delete character operations ... 1383
61.8 Delete non-matching lines ... 1383
61.9 Duplicate lines or selections ... 1383
61.10 Filter text through external commands ... 1384
61.11 Format paragraph ... 1384
61.12 Generate increasing numbers column ... 1384
61.13 Handle common typos ... 1385
61.14 Increment/decrement numbers ... 1385
61.15 Insert column of text ... 1385
61.16 Insert numbering ... 1386
61.17 Join lines with custom separator ... 1386
61.18 Lowercase/uppercase current line ... 1386
61.19 Put text from register ... 1387
61.20 ROT13 encoding ... 1387
61.21 Remove duplicate lines ... 1387
61.22 Remove trailing whitespace ... 1387

61.23 Replace mode operations ... 1388
61.24 Reverse lines ... 1388
61.25 Text alignment and padding ... 1388
61.26 Text statistics ... 1389
61.27 Transpose characters ... 1389
61.28 Undo and redo operations ... 1389
61.29 Uppercase current word ... 1390
61.30 Word count methods ... 1390
61.31 Work with CSV files ... 1390

62 Text objects ... 1391
62.1 Select around parentheses ... 1391
62.2 Select inside quotes ... 1391
62.3 Text objects - HTML/XML tags ... 1391
62.4 Text objects - alternative bracket notation ... 1392
62.5 Text objects - angle brackets ... 1392
62.6 Text objects - around brackets ... 1392
62.7 Text objects - inside brackets ... 1393
62.8 Text objects - quoted strings ... 1393
62.9 Text objects - sentences and paragraphs ... 1393
62.10 Text objects - square brackets ... 1393
62.11 Text objects - word variations ... 1394
62.12 Text objects with operators ... 1394

63 Treesitter ... 1395
63.1 Build a treesitter-based code outline ... 1395
63.2 Check if treesitter is available for filetype ... 1396
63.3 Create custom treesitter text objects ... 1397
63.4 Enhanced text objects with treesitter ... 1398
63.5 Find all nodes of specific type ... 1399
63.6 Get treesitter node under cursor ... 1400
63.7 Get treesitter parser for buffer ... 1400
63.8 Handle treesitter injection languages ... 1401
63.9 Highlight custom patterns with treesitter ... 1402
63.10 Navigate treesitter tree programmatically ... 1402
63.11 Treesitter folding ... 1403
63.12 Treesitter incremental selection ... 1403
63.13 Treesitter install parser ... 1404
63.14 Treesitter node navigation ... 1404
63.15 Treesitter playground ... 1404

63.16 Treesitter query predicates and directives ... 1405
63.17 Treesitter swap nodes ... 1406
63.18 Use treesitter for smart text editing ... 1406
63.19 Write custom treesitter queries ... 1407

64 Ui ... 1409

64.1 Always Show Statusline in Single Window ... 1409
64.2 Beautiful transparent theme with background image ... 1409
64.3 Change Cursor Color and Shape in Different Modes ... 1410
64.4 Change highlight group on the fly ... 1410
64.5 Check highlight groups ... 1411
64.6 Color Scheme for Different Directories ... 1411
64.7 Console-Like Fonts for Neovim on Windows ... 1412
64.8 Cross-Platform Font Size Management ... 1412
64.9 Custom statusline ... 1413
64.10 Customize Mouse Cursor Shape in Vim ... 1413
64.11 Customize Omni-Completion Popup Menu Colors ... 1414
64.12 Customize Toolbar for Quick Launching ... 1414
64.13 Customize Vim Statusline with Dynamic Information ... 1414
64.14 Disable Cursor Blinking in Vim/Neovim ... 1415
64.15 Disable Mouse Interaction in Vim ... 1415
64.16 Display Non-Native File Format in Statusline ... 1416
64.17 Dynamic Cursor Shape in Different Modes ... 1417
64.18 Dynamically Adjust Font Size in GUI ... 1417
64.19 Dynamically Switch Vim Color Schemes ... 1418
64.20 Flash yanked text ... 1419
64.21 Full Mouse Support in Console Vim ... 1419
64.22 Hide/Toggle GUI Elements in Neovim ... 1419
64.23 Highlight Inserted Text in Insert Mode ... 1420
64.24 Highlight Matching Braces Instantly ... 1421
64.25 Highlight groups ... 1422
64.26 Identify Active Vim Instance by Color ... 1422
64.27 Mouse and UI Enhancements ... 1423
64.28 Non-Blinking Block Cursor in Linux Console ... 1423
64.29 Prevent Mouse Drag from Entering Visual Mode ... 1424
64.30 Print treesitter highlight group info ... 1424
64.31 Quick Font Selector Keymapping ... 1424
64.32 Quick Window Maximization Shortcut ... 1425
64.33 Quickly Adjust Font Size in Neovim GUI ... 1425
64.34 Remove Specific Toolbar Buttons ... 1426

64.35 Show Matching Braces Briefly ... 1427
64.36 Show Non-Native File Format in Statusline ... 1427
64.37 Statusline Color Indicator for Vim Modes ... 1428
64.38 Time-Based Color Scheme Selection ... 1429
64.39 Toggle GUI Elements Dynamically ... 1430
64.40 Toggle Line Numbers Easily ... 1431
64.41 Use Tearoff Menus with Keyboard Shortcuts ... 1432

65 Vimscript fundamentals ... 1433
65.1 Autocommand creation in script ... 1433
65.2 Basic variable assignment and types ... 1433
65.3 Built-in function usage ... 1434
65.4 Conditional statements and logic ... 1434
65.5 Debugging Vim scripts ... 1434
65.6 Error handling with try-catch ... 1435
65.7 Event handling and callbacks ... 1435
65.8 File and buffer operations ... 1436
65.9 Function definition and calling ... 1436
65.10 List and dictionary operations ... 1437
65.11 Loops and iteration ... 1437
65.12 Lua integration in Vim script ... 1438
65.13 Mappings in Vim script ... 1438
65.14 Option manipulation ... 1439
65.15 Regular expressions in Vim script ... 1439
65.16 Script sourcing and modules ... 1440
65.17 String formatting and printf ... 1440
65.18 String operations and concatenation ... 1440
65.19 System command execution ... 1441
65.20 User command definition ... 1441

66 Visual mode ... 1443
66.1 Alternate Mouse Selection Method ... 1443
66.2 Apply Substitutions to Visual Block Selections ... 1443
66.3 Block Replacement in Visual Mode ... 1444
66.4 Blockwise Edit Multiple Lines Simultaneously ... 1444
66.5 Create Rectangular Block of Exact Size ... 1445
66.6 Drag Text Using Arrow Key Mappings ... 1445
66.7 Easy Blockwise Mouse Selection in Vim ... 1446
66.8 Flexible Line Selection with Visual Mode ... 1446
66.9 Generate Incrementing Column Numbers ... 1447

66.10 Handle Shift Key Mistakes in Visual Mode ... 1447
66.11 Highlight Text Inside Matching Parentheses ... 1448
66.12 Increment Numbers in Visual Block Mode ... 1448
66.13 Insert Text Across Multiple Lines in Visual Block Mode ... 1449
66.14 Pad Lines for Easy Visual Block Editing ... 1449
66.15 Powerful Visual Selection and Paste Techniques ... 1450
66.16 Quick Brace Matching Visual Selection ... 1450
66.17 Quick Visual Block Commenting ... 1451
66.18 Quickly Reselect Last Visual Selection ... 1451
66.19 Repeat Last Edit Across Visual Block Selection ... 1452
66.20 Repeat Visual Block Changes Across Lines ... 1452
66.21 Replace Visual Block with Another Block ... 1453
66.22 Reverse Selected Text in Visual Mode ... 1453
66.23 Select Blocks with % Motion ... 1454
66.24 Select Different Block Types Easily ... 1454
66.25 Select Entire Function with Visual Commands ... 1455
66.26 Select Recently Pasted Text Easily ... 1455
66.27 Use Virtual Edit for Block Selection ... 1456
66.28 Visual Mode Block Indent Preservation ... 1456
66.29 Visual Mode Bracket Surrounding ... 1457
66.30 Visual Mode Text Swapping ... 1457
66.31 Visual block append ... 1458
66.32 Visual mode - corner and edge movement ... 1458
66.33 Visual mode - joining and substitution ... 1458
66.34 Visual mode - operators and transformations ... 1459
66.35 Visual mode - paste and replace ... 1459
66.36 Visual mode - tag and keyword operations ... 1460
66.37 Visual mode - toggle and change types ... 1460
66.38 Yank and delete in visual mode ... 1460
66.39 Yank highlighting ... 1461

67 Visual mode (advanced) ... 1463

67.1 Incremental number sequences with g<C-a> ... 1463
67.2 Visual block append to varying line lengths ... 1463
67.3 Visual block column editing ... 1463
67.4 Visual block column insertion ... 1464
67.5 Visual line operations ... 1464
67.6 Visual mode column operations ... 1464
67.7 Visual mode incremental selection ... 1465
67.8 Visual mode indentation and alignment ... 1465

67.9 Visual mode line manipulation ... 1465
67.10 Visual mode macro application ... 1466
67.11 Visual mode pattern matching ... 1466
67.12 Visual mode rectangle operations ... 1466
67.13 Visual mode register operations ... 1467
67.14 Visual mode search and replace ... 1467
67.15 Visual mode smart selection ... 1468
67.16 Visual mode sorting and filtering ... 1468
67.17 Visual mode text transformation ... 1468
67.18 Visual mode text wrapping ... 1469
67.19 Visual mode with external filters ... 1469
67.20 Visual mode with folds ... 1469
67.21 Visual mode with global commands ... 1470
67.22 Visual mode with jumps and changes ... 1470
67.23 Visual mode with marks ... 1470
67.24 Visual mode word selection shortcuts ... 1471
67.25 Visual selection with text objects ... 1471

68 Window management ... 1473

68.1 Alt Key Window Management Shortcuts ... 1473
68.2 Alternative Window Navigation Shortcuts ... 1473
68.3 Auto-Resize Window for Diff Mode ... 1474
68.4 Better gm command ... 1474
68.5 Buffer Split Management ... 1475
68.6 Change cursor shape in modes ... 1476
68.7 Close Tab and Focus Left Tab Automatically ... 1476
68.8 Close all other windows ... 1477
68.9 Control New Window Split Position ... 1477
68.10 Customize GVim Popup Menu for Window Management ... 1478
68.11 Display Python Docs in Preview Window ... 1478
68.12 Dynamic Tag Preview with Full-Height Vertical Window ... 1479
68.13 Dynamic Window Resizing Keymaps ... 1480
68.14 Dynamic Window Resizing for Line Numbers ... 1481
68.15 Dynamic Window Resizing with F1 Key ... 1481
68.16 Dynamically Resize Quickfix Window Height ... 1483
68.17 Easily Compare Two Files in Vim/Neovim ... 1483
68.18 Easily Turn Off Diff Mode ... 1484
68.19 Enable Mouse Focus Across Splits ... 1485
68.20 Execute SQL Queries in Split Window ... 1485
68.21 Focus mode for writing ... 1486

68.22 Go to Nth-from-Last Window 1486
68.23 Keep Quickfix Window at Fixed Height 1487
68.24 Keep cursor centered 1488
68.25 Keep window when closing buffer 1488
68.26 Maximize Current Window Quickly 1489
68.27 Maximize Vim Window Size in KDE 1489
68.28 Maximize Vim Window on Windows 1490
68.29 Minimize and Restore Vim Windows Quickly 1490
68.30 Move Window Between Tabs Dynamically 1491
68.31 Move window to tab 1492
68.32 Move windows 1492
68.33 Multiple Views of Same Buffer in Different Tabs 1492
68.34 Open GVim Matching Console Window Position 1493
68.35 Open Multiple Files in New Windows 1493
68.36 Open Neovim Matching Console Window Geometry 1495
68.37 Open New Buffers Below Current Window 1496
68.38 Open New Vertical Buffers to the Right 1496
68.39 Open Source Navigator Files in Vim Tabs 1496
68.40 Open Tag in Split Window 1497
68.41 Prevent Accidental Window Closure 1497
68.42 Prevent Cursor Movement When Switching Windows 1498
68.43 Quick Tab Management for Editing Multiple Files 1499
68.44 Quick Window Navigation with Spacebar 1499
68.45 Quick Window Resizing with Keyboard Shortcuts 1500
68.46 Quick Window Resizing with Keymaps 1500
68.47 Quick file explorer 1501
68.48 Quickly Resize Windows in Neovim 1502
68.49 Resize Terminal for Diff View 1503
68.50 Resize Windows with Alt+Arrow Keys 1503
68.51 Resize Windows with Ctrl + Cursor Keys 1504
68.52 Restore Current Window/Buffer After Global Command 1504
68.53 Rolodex Window Mode 1505
68.54 Smart window navigation keymaps 1505
68.55 Special window commands 1506
68.56 Synchronous Scrolling in Multiple Windows 1506
68.57 Tab management 1507
68.58 Toggle Line Numbers Across All Windows 1507
68.59 Toggle Window Maximization Without Losing Splits 1508
68.60 Toggle Window Maximize/Restore 1509
68.61 Use Window-Local Variables for Initialization 1510
68.62 Window Splitting and Sizing Control 1510

68.63 Window Zooming Toggle 1511
68.64 Window closing 1512
68.65 Window commands from Ex mode 1512
68.66 Window navigation basics 1513
68.67 Window position navigation 1513

69 Workflow patterns 1515
69.1 Add lines to multiple files with cfdo 1515
69.2 Backup and recovery workflow 1515
69.3 Build and deployment workflow 1516
69.4 Code quality and standards workflow 1516
69.5 Code review and annotation workflow 1517
69.6 Code review and collaboration workflow 1518
69.7 Configuration management workflow 1518
69.8 Documentation workflow 1519
69.9 Error handling and debugging patterns 1519
69.10 Focus and distraction management 1520
69.11 Git workflow integration 1521
69.12 Knowledge management workflow 1521
69.13 Learning and experimentation workflow 1522
69.14 Multi-file editing workflow 1522
69.15 Performance profiling workflow 1523
69.16 Project workspace initialization 1523
69.17 Refactoring workflow patterns 1524
69.18 Resolve merge conflicts with git jump 1525
69.19 Search and replace workflow 1525
69.20 Session and workspace persistence 1526
69.21 Testing and debugging workflow 1526

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. The plugin 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.

Daily tip from Neovim Tips plugin

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 functions

1.1 Advanced CSV Column Navigation and Manipulation

Category: advanced_functions
Tags: csv, data-processing, text-manipulation

Provides powerful navigation and manipulation features for CSV files, including column highlighting, searching within columns, and sorting by specific columns

Example: Example (Vim)

```
function! CSVH(colnr)
  if a:colnr > 1
    let n = a:colnr - 1
    execute 'match Keyword /^\([^,]*,\)\{'.n.'}\zs[^,]*'
    execute 'normal! 0'.n.'f,'
  elseif a:colnr == 1
    match Keyword /^[^,]*/
    normal! 0
  else
    match
  endif
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
```

Example: Example (Neovim)

```
function _G.highlight_csv_column(colnr)
  if colnr > 1 then
    local n = colnr - 1
    vim.cmd(string.format('match Keyword /^([^,]*,){%d}\\zs[^,]*/', n))
    vim.cmd(string.format('normal! 0%df,', n))
  elseif colnr == 1 then
    vim.cmd('match Keyword /^[^,]*/')
    vim.cmd('normal! 0')
  else
    vim.cmd('match')
  end
end

vim.api.nvim_create_user_command('Csv', function(opts)
  _G.highlight_csv_column(tonumber(opts.args))
```

```
end, { nargs = 1 })
```

Source: vim.fandom.com

1.2 Advanced PHP Function Documentation Viewer

Category: advanced_functions
Tags: php, documentation, custom-function, preview-window

Create a sophisticated function to fetch and display PHP documentation in a preview window with smart parsing

Example: Example (Vim)

```
function! OpenPhpFunction (keyword)
  let proc_keyword = substitute(a:keyword , '_', '-', 'g')
  exe 'pedit'
  exe 'wincmd P'
  exe 'enew'
  exe "set buftype=nofile"
  exe 'silent r!lynx -dump -nolist http://php.net/'.proc_keyword
  exe 'norm gg'
  exe 'call search("____________________________________")'
  exe 'norm dgg'
  exe 'call search("User Contributed Notes")'
  exe 'norm dGgg'
endfunction
```

Example: Example (Neovim)

```
local function open_php_function(keyword)
  -- Replace underscores with hyphens
  local proc_keyword = keyword:gsub('_', '-')

  -- Open preview window
  vim.cmd('pedit')
  vim.cmd('wincmd P')

  -- Create new buffer
  vim.cmd('enew')
  vim.bo.buftype = 'nofile'

  -- Fetch documentation
  local cmd = string.format('lynx -dump -nolist http://php.net/%s',
  ↪  proc_keyword)
  local output = vim.fn.systemlist(cmd)

  -- Clean up output
  local cleaned_output = {}
  local start_index = nil
  for i, line in ipairs(output) do
```

```
    if line:match('____________________________________') then
      start_index = i
      break
    end
  end

  if start_index then
    for i = start_index + 1, #output do
      if line:match('User Contributed Notes') then
        break
      end
      table.insert(cleaned_output, output[i])
    end
  end

  -- Set buffer contents
  vim.api.nvim_buf_set_lines(0, 0, -1, false, cleaned_output)
  vim.cmd('norm gg')
end

-- Create a command and mapping
vim.api.nvim_create_user_command('PhpDoc', function()
  local word = vim.fn.expand('<cword>')
  open_php_function(word)
end, {})

vim.keymap.set('n', '<leader>pd', ':PhpDoc<CR>', { desc = 'PHP Function
↪  Documentation' })
```

Source: vim.fandom.com

1.3 Automatic Java Import Generation

Category: advanced_functions
Tags: java, import-management, productivity

Quickly insert Java import statements for classes under cursor using a custom function with error handling and preview window management

Example: Example (Vim)

```
function! JavaInsertImport()
  exe "normal mz"
  let cur_class = expand("<cword>")
  try
    if search('^\s*import\s.*\.' . cur_class . '\s*;') > 0
      throw getline('.') . ": import already exist!"
    endif
    wincmd }
    wincmd P
    1
    if search('^\s*public.*\s\%(class\|interface\)\s\+' . cur_class) > 0
```

```
      1
      if search('^\s*package\s') > 0
        yank y
      else
        throw "Package definition not found!"
      endif
    else
      throw cur_class . ": class not found!"
    endif
    wincmd p
    normal! G
    if search('^\s*import\s', 'b') > 0
      put y
    else
      1
      put! y
    endif
    substitute/^\s*package/import/g
    substitute/\s\+/ /g
    exe "normal! 2ER." . cur_class . ";\<Esc>lD"
  catch /.*/
    echoerr v:exception
  finally
    silent! wincmd P
    if &previewwindow
      bwipeout
    endif
    exe "normal! `z"
  endtry
endfunction

noremap <F5> :call JavaInsertImport()<CR>
```

Example: Example (Neovim)

```
function _G.java_insert_import()
  local cur_class = vim.fn.expand('<cword>')

  local ok, result = pcall(function()
    -- Check if import already exists
    if vim.fn.search('^\s*import\s.*\.' .. cur_class .. '\s*;') > 0 then
      error('Import already exists')
    end

    -- Open preview window with tag
    vim.cmd('wincmd }')
    vim.cmd('wincmd P')

    -- Find package and class/interface definition
    vim.cmd('normal! 1G')
    if vim.fn.search('^\s*public.*\s\%(class\|interface\)\s\+' .. cur_class)
    ↪  == 0 then
      error('Class/interface not found')
    end
```

```
    local package_line = vim.fn.search('^\s*package\s')
    if package_line == 0 then
      error('Package definition not found')
    end

    -- Copy package line and convert to import
    local package = vim.fn.getline(package_line)
    local import_line = package:gsub('^%s*package', 'import'):gsub(';$', '')
    ↪  .. '.' .. cur_class .. ';'

    -- Return to original window and insert import
    vim.cmd('wincmd p')
    vim.cmd('normal! G')

    local import_pos = vim.fn.search('^\s*import\s', 'b')
    if import_pos > 0 then
      vim.fn.append(import_pos, import_line)
    else
      vim.fn.append(0, import_line)
    end
  end)

  if not ok then
    print('Error: ' .. result)
  end

  -- Clean up preview window
  vim.cmd('silent! wincmd P')
  if vim.wo.previewwindow then
    vim.cmd('bwipeout')
  end
end

vim.keymap.set('n', '<F5>', _G.java_insert_import, { desc = 'Insert Java
↪  Import' })
```

Source: vim.fandom.com

1.4 Custom Function to Open File in Visual Studio

Category: advanced_functions
Tags: integration, windows, external-tools

Create a Vim function to open the current file and line number in Visual Studio using a custom script (Windows-specific)

Example: Example (Vim)

```
function! DevEnvDDE()
  let cmd = '!devenvdden.pl %:p ' . line(".")
  silent execute cmd
```

```
endfunction
```

Example: Example (Neovim)

```
function _G.open_file_in_visual_studio()
  local file_path = vim.fn.expand('%:p')
  local line_number = vim.fn.line('.')
  local cmd = string.format('!devenvdden.pl %s %d', file_path, line_number)
  vim.cmd(cmd)
end

-- Optional: Map to a keybinding
vim.keymap.set('n', '<leader>vs', _G.open_file_in_visual_studio, { desc =
↪  'Open current file in Visual Studio' })
```

Source: vim.fandom.com

1.5 Dynamic Footnote Insertion with Vim Function

Category: advanced_functions
Tags: text-editing, markdown, functions, footnotes

A flexible Vim function to insert and manage footnotes dynamically, with numbering and positioning near signatures

Example: Example (Vim)

```
function! VimFootnotes()
  if exists("b:vimfootnotenumber")
    let b:vimfootnotenumber = b:vimfootnotenumber + 1
    let cr = ""
  else
    let b:vimfootnotenumber = 0
    let cr = "\<CR>"
  endif
  let b:pos = line('.') . ' | normal! ' . virtcol('.') . '|' . '4l'
  exe "normal a[" . b:vimfootnotenumber . "]\<Esc>G"
  if search("-- $", "b")
    exe "normal O" . cr . "[" . b:vimfootnotenumber . "] "
  else
    exe "normal o" . cr . "[" . b:vimfootnotenumber . "] "
  endif
  startinsert!
endfunction
```

Example: Example (Neovim)

```
function _G.vim_footnotes()
  local footnote_number = vim.b.vimfootnotenumber or 0
  local cr = footnote_number == 0 and '\n' or ''
```

```
  footnote_number = footnote_number + 1
  vim.b.vimfootnotenumber = footnote_number

  -- Insert footnote marker
  vim.api.nvim_feedkeys(string.format('[%d]', footnote_number), 'n', false)
  vim.api.nvim_command('normal! G')

  -- Check for signature and place footnote appropriately
  if vim.fn.search('-- $', 'b') > 0 then
    vim.api.nvim_command('normal! O' .. cr .. string.format('[%d] ',
    ↪  footnote_number))
  else
    vim.api.nvim_command('normal! o' .. cr .. string.format('[%d] ',
    ↪  footnote_number))
  end

  vim.cmd('startinsert!')
end

-- Map to a key (example)
vim.keymap.set('i', '<C-f>', _G.vim_footnotes, { desc = 'Insert Footnote' })
```

Source: vim.fandom.com

1.6 Dynamically Display Current Function Name

Category: advanced_functions
Tags: function-detection, code-navigation, cursor-context

A function to detect and display the name of the current function, even in nested contexts like control statements or nested functions

Example: Example (Vim)

```
function WhatFunctionAreWeIn()
  let strList = ["while", "foreach", "ifelse", "if else", "for", "if",
  ↪  "else", "try", "catch", "case", "switch"]
  let foundcontrol = 1
  let position = ""
  let pos=getpos(".")
  let view=winsaveview()
  while (foundcontrol)
    let foundcontrol = 0
    normal [{
    call search('\S','bW')
    let tempchar = getline(".")[col(".") - 1]
    if (match(tempchar, ")") >=0 )
      normal %
      call search('\S','bW')
    endif
    let tempstring = getline(".")
```

```
    for item in strList
      if( match(tempstring,item) >= 0 )
        let position = item . " - " . position
        let foundcontrol = 1
        break
      endif
    endfor
    if(foundcontrol == 0)
      call cursor(pos)
      call winrestview(view)
      return tempstring.position
    endif
  endwhile
  call cursor(pos)
  call winrestview(view)
  return tempstring.position
endfunction
```

Example: Example (Neovim)

```
function _G.WhatFunctionAreWeIn()
  local strList = {"while", "foreach", "ifelse", "if else", "for", "if",
  ↪  "else", "try", "catch", "case", "switch"}
  local foundcontrol = true
  local position = ""
  local pos = vim.fn.getpos(".")
  local view = vim.fn.winsaveview()

  while foundcontrol do
    foundcontrol = false
    vim.cmd('normal! [')
    vim.fn.search('\S', 'bW')
    local tempchar = vim.fn.getline(".")sub(vim.fn.col("."),
    ↪  vim.fn.col("."))

    if tempchar:match(")") then
      vim.cmd('normal! %')
      vim.fn.search('\S', 'bW')
    end

    local tempstring = vim.fn.getline(".")
    for _, item in ipairs(strList) do
      if tempstring:find(item) then
        position = item .. " - " .. position
        foundcontrol = true
        break
      end
    end

    if not foundcontrol then
      vim.fn.cursor(pos[2], pos[3])
      vim.fn.winrestview(view)
      return tempstring .. position
    end
```

```
  end

  vim.fn.cursor(pos[2], pos[3])
  vim.fn.winrestview(view)
  return tempstring .. position
end

-- Optional: Create a command to call the function
vim.api.nvim_create_user_command('ShowCurrentFunction', function()
  print(_G.WhatFunctionAreWeIn())
end, {})
```

Source: vim.fandom.com

1.7 Extract Text from HTML Files

Category: advanced_functions
Tags: html, text-extraction, utility

Use a text-based browser to extract readable text from HTML files or URLs

Example: Example (Vim)

```
function! ViewHtmlText(url)
  if !empty(a:url)
    new
    setlocal buftype=nofile bufhidden=hide noswapfile
    execute 'r !elinks ' . a:url . ' -dump -dump-width ' . winwidth(0)
    1d
  endif
endfunction

" Mappings for different preview scenarios
nnoremap <Leader>H :update<Bar>call ViewHtmlText(expand('%:p'))<CR>
vnoremap <Leader>h y:call ViewHtmlText(@@)<CR>
nnoremap <Leader>h :call ViewHtmlText(@+)<CR>
```

Example: Example (Neovim)

```
function _G.view_html_text(url)
  if url and url ~= '' then
    vim.cmd('new')
    vim.api.nvim_buf_set_option(0, 'buftype', 'nofile')
    vim.api.nvim_buf_set_option(0, 'bufhidden', 'hide')
    vim.api.nvim_buf_set_option(0, 'swapfile', false)

    local output = vim.fn.systemlist(string.format('elinks %s -dump
    ↪  -dump-width %d', url, vim.o.columns))
    vim.api.nvim_buf_set_lines(0, 0, -1, false, output)
    vim.cmd('normal! gg')
  end
```

```
end

-- Mappings
vim.keymap.set('n', '<Leader>H', function()
  vim.cmd.update()
  _G.view_html_text(vim.fn.expand('%:p'))
end, { desc = 'Preview current HTML file text' })

vim.keymap.set('v', '<Leader>h', function()
  local selected_text = vim.fn.getreg('@')
  _G.view_html_text(selected_text)
end, { desc = 'Preview selected HTML/URL' })

vim.keymap.set('n', '<Leader>h', function()
  _G.view_html_text(vim.fn.getreg('+'))
end, { desc = 'Preview clipboard HTML/URL' })
```

Source: vim.fandom.com

1.8 Flexible Compilation Function

Category: advanced_functions
Tags: compilation, file-management, automation

Create a flexible function to compile files in the current directory, with automatic directory handling

Example: Example (Vim)

```
function! Make()
  let curr_dir = expand('%:h')
  if curr_dir == ''
    let curr_dir = '.'
  endif
  execute 'lcd ' . curr_dir
  execute 'make %:r.o'
  execute 'lcd -'
endfunction

nnoremap <F7> :update<CR>:call Make()<CR>
```

Example: Example (Neovim)

```
function _G.compile_current_file()
  local curr_dir = vim.fn.expand('%:h')
  if curr_dir == '' then curr_dir = '.' end

  vim.cmd.lcd(curr_dir)
  vim.cmd.make(vim.fn.expand('%:r') .. '.o')
  vim.cmd.lcd('-')
end
```

```
vim.keymap.set('n', '<F7>', function()
  vim.cmd.write()
  _G.compile_current_file()
end, { desc = 'Compile current file in project directory' })
```

Source: vim.fandom.com

1.9 Flexible Heading Underline Function

Category: advanced_functions
Tags: text-formatting, custom-function, user-command

Create a flexible function to underline text with various characters, adaptable to different document styles

Example: Example (Vim)

```
function! s:Underline(chars)
  let chars = empty(a:chars) ? '-' : a:chars
  let nr_columns = virtcol('$') - 1
  let uline = repeat(chars, (nr_columns / len(chars)) + 1)
  put =strpart(uline, 0, nr_columns)
endfunction
command! -nargs=? Underline call s:Underline(<q-args>)
```

Example: Example (Neovim)

```
-- Flexible underline function in Lua
local function underline(chars)
  chars = chars or '-'
  local line = vim.api.nvim_get_current_line()
  local line_length = vim.fn.virtcol('$') - 1

  -- Create underline repeating the character(s)
  local uline = string.rep(chars, math.floor(line_length / #chars) + 1)
  uline = uline:sub(1, line_length)

  -- Insert underline after current line
  vim.api.nvim_buf_set_lines(0, vim.fn.line('.'), vim.fn.line('.'), false,
  ↪ {line, uline})
end

-- Create user command
vim.api.nvim_create_user_command('Underline', function(opts)
  underline(opts.args ~= '' and opts.args or nil)
end, { nargs = '?' })
```

Source: vim.fandom.com

1.10 Flexible Underline Command

Category: advanced_functions
Tags: text-formatting, user-command, customization

Create a flexible user command to underline text with custom characters

Example: Example (Vim)

```
function! s:Underline(chars)
  let chars = empty(a:chars) ? '-' : a:chars
  let nr_columns = virtcol('$') - 1
  let uline = repeat(chars, (nr_columns / len(chars)) + 1)
  put =strpart(uline, 0, nr_columns)
endfunction
command! -nargs=? Underline call s:Underline(<q-args>)
```

Example: Example (Neovim)

```
local function underline(chars)
  chars = chars == '' and '-' or chars
  local columns = vim.fn.virtcol('$') - 1
  local uline = chars:rep(math.floor(columns / #chars) + 1)
  uline = uline:sub(1, columns)

  -- Insert the underline
  vim.api.nvim_put({uline}, 'l', true, false)
end

-- Create user command
vim.api.nvim_create_user_command('Underline', function(opts)
  underline(opts.args)
end, { nargs = '?' })
```

Source: vim.fandom.com

1.11 Flexible Underline Function

Category: advanced_functions
Tags: text-formatting, user-command, customization

Create a custom command to underline text with various characters, adapting to the current line's length

Example: Example (Vim)

```
function! s:Underline(chars)
  let chars = empty(a:chars) ? '-' : a:chars
  let nr_columns = virtcol('$') - 1
  let uline = repeat(chars, (nr_columns / len(chars)) + 1)
```

```
  put =strpart(uline, 0, nr_columns)
endfunction
command! -nargs=? Underline call s:Underline(<q-args>)
```

Example: Example (Neovim)

```
function _G.underline_text(chars)
  chars = chars or '-'
  local nr_columns = vim.fn.virtcol('$') - 1
  local uline = chars:rep(math.floor(nr_columns / #chars) + 1)
  uline = uline:sub(1, nr_columns)
  vim.api.nvim_put({uline}, 'l', true, false)
end

vim.api.nvim_create_user_command('Underline', function(opts)
  _G.underline_text(opts.args ~= '' and opts.args or nil)
end, { nargs = '?' })
```

Source: vim.fandom.com

1.12 Generate Java Bean Getters and Setters

Category: advanced_functions
Tags: java, code-generation, text-manipulation

Quickly generate Java bean getter and setter methods for a variable by prompting for name and type

Example: Example (Vim)

```
function! s:AddBean()
  let line = line('.')
  let name = inputdialog('Enter the name of the variable: ')
  let type = inputdialog('Enter the type of the variable: ')
  let upperName = substitute(name, '^(\w)(.*)
$', '\u\1\2', '')
  call append(line, "\t}")
  call append(line, "\t\tthis.".name." = ".name.";")
  call append(line, "\tpublic void set".upperName."(".type." ".name.") {")
  call append(line, "")
  call append(line, "\t}")
  call append(line, "\t\treturn (this.".name.");")
  call append(line, "\tpublic ".type." get""".upperName."() { ")
  call append(line, "")
  call append(line, "\tprivate ".type." ".name.";")
  call append(line, "\t//".name)
  return line
endfunction

nnoremap <buffer> <silent>dc :call <SID>AddBean()<CR>
```

Example: Example (Neovim)

```
local function add_bean()
  local line = vim.fn.line('.')
  local name = vim.fn.input('Enter the name of the variable: ')
  local type = vim.fn.input('Enter the type of the variable: ')
  local upper_name = name:gsub('^%l', string.upper)

  local bean_lines = {
    string.format('        private %s %s;', type, name),
    string.format('        //%s', name),
    string.format('        public %s get%s() {', type, upper_name),
    '                return this.' .. name .. ';',
    '        }',
    string.format('        public void set%s(%s %s) {', upper_name, type,
    ↪  name),
    '                this.' .. name .. ' = ' .. name .. ';',
    '        }'
  }

  vim.api.nvim_buf_set_lines(0, line, line, false, bean_lines)
end

vim.keymap.set('n', 'dc', add_bean, { buffer = true, silent = true })
```

Source: vim.fandom.com

1.13 Generate Web Forum Compatible HTML Snippet

Category: advanced_functions
Tags: html, code-export, custom-function

Create a function to generate HTML code snippets that work across different web platforms

Example: Example (Vim)

```
function! MyToHtml(line1, line2)
  let g:html_use_css = 0
  exec a:line1.',.a:line2.'TOhtml'
  %g/<body/normal k$dgg
  %s/<body\s*\(bgcolor="[^"]*"\)\s*text=\("[^"]*"\)\s*>/<table \1
  ↪  cellPadding=0><tr><td><font color=\2>/
  %s#</body>\(.|\n\)*</html>#\='</font></td></tr></table>'#i
endfunction
command! -range=% MyToHtml :call MyToHtml(<line1>,<line2>)
```

Example: Example (Neovim)

```
function _G.my_to_html(line1, line2)
  vim.g.html_use_css = 0
  vim.cmd(string.format('%d,%dTOhtml', line1, line2))
```

```
  -- Additional HTML manipulation logic would be translated here
end

vim.api.nvim_create_user_command('MyToHtml', function(opts)
  _G.my_to_html(opts.line1, opts.line2)
end, { range = true })
```

Source: vim.fandom.com

1.14 Move Characters to Specific Column Range

Category: advanced_functions
Tags: text-manipulation, range-commands, advanced-editing

Custom Vim function to move specified characters to a specific column within a line range, useful for text alignment and formatting

Example: Example (Vim)

```
function! MCCB (num_occr, mv_char, col_num) range
    if (a:firstline <= a:lastline)
        nmap s 80i <ESC>
        let line_num = a:firstline
        while line_num <= a:lastline
            execute "normal " . line_num . "G0" . a:num_occr . "f" .
            ↪  a:mv_char . "s" . a:col_num . "|dw"
            let line_num = line_num + 1
        endwhile
        nunmap s
    else
        execute printf('ERROR : Start line %d is higher than End line %d',
        ↪  a:firstline, a:lastline)
    endif
endfunction
```

Example: Example (Neovim)

```
function _G.move_chars_to_column(first_line, last_line, occurrence,
↪  move_char, col_num)
    if first_line <= last_line then
        for line_num = first_line, last_line do
            vim.cmd(string.format('normal! %dG0%df%ss%d|dw', line_num,
            ↪  occurrence, move_char, col_num))
        end
    else
        print(string.format('ERROR: Start line %d is higher than End line
        ↪  %d', first_line, last_line))
    end
end

-- Usage example
```

```
-- :lua move_chars_to_column(5, 11, 1, '=', 8)
```

Source: vim.fandom.com

1.15 Screen-Aware Command Execution

Category: advanced_functions
Tags: terminal, command-execution, screen

Dynamically modify command execution based on whether Screen is running

Example: Example (Vim)

```
if match($TERM, "screen")!=-1
  let g:GNU_Screen_used = 1
else
  let g:GNU_Screen_used = 0
endif

function InScreen(command)
  return g:GNU_Screen_used ? 'screen '.a:command : a:command
endfunction
```

Example: Example (Neovim)

```
local function is_screen_running()
  return vim.env.TERM:match("screen") ~= nil
end

vim.g.gnu_screen_used = is_screen_running()

local function in_screen(command)
  return vim.g.gnu_screen_used and "screen " .. command or command
end
```

Source: vim.fandom.com

1.16 Smart Preprocessor Directive Commenter

Category: advanced_functions
Tags: code-commenting, preprocessor, c-language

Automatically adds contextual comments to preprocessor directives like #endif and #else, improving code readability by explaining the purpose of conditional compilation blocks

Example: Example (Vim)

```
function! SmartPreProcCommenter()
  mark y
  let saved_wrapscan=&wrapscan
  set nowrapscan
  let elsecomment=""
  let endcomment=""
  try
    $?^\s*#if
    while 1
      let content=getline('.')
      let elsecomment=BuildElseComment(content,elsecomment)
      let endcomment=BuildEndComment(content,endcomment)
      s/^\s*\zs#/##
      /^\s*#\(elif\|else\|endif\)
      let content=getline('.')
      if match(content,'^\s*#elif') != -1
        continue
      elseif match(content,'^\s*#else') != -1
        call setline('.',ReplaceComment(content,elsecomment))
        s/^\s*\zs#/##
        /^\s*#endif
      endif
      call setline('.',ReplaceComment(getline('.'),endcomment))
      s/^\s*\zs#/##
      ?^\s*#if
    endwhile
  catch /search hit TOP/
    silent! %s/^\s*\zs##/#
  endtry
  let &wrapscan=saved_wrapscan
  normal `y
endfunc
```

Example: Example (Neovim)

```
function SmartPreProcCommenter()
  vim.cmd('mark y')
  local saved_wrapscan = vim.o.wrapscan
  vim.o.wrapscan = false
  local elsecomment = ""
  local endcomment = ""

  local ok, err = pcall(function()
    vim.cmd('$?^\s*#if')
    while true do
      local content = vim.fn.getline('.')
      elsecomment = BuildElseComment(content, elsecomment)
      endcomment = BuildEndComment(content, endcomment)

      vim.cmd('s/^\s*\zs#/##')
      vim.cmd('/^\s*#\(elif\|else\|endif\)')

      content = vim.fn.getline('.')
```

```
      if content:match('^\s*#elif') then
        -- continue
      elseif content:match('^\s*#else') then
        vim.fn.setline('.', ReplaceComment(content, elsecomment))
        vim.cmd('s/^\s*\zs#/##')
        vim.cmd('/^\s*#endif')
      end

      vim.fn.setline('.', ReplaceComment(vim.fn.getline('.'), endcomment))
      vim.cmd('s/^\s*\zs#/##')
      vim.cmd('?^\s*#if')
    end
  end)

  if not ok and err:match('search hit TOP') then
    vim.cmd('silent! %s/^\s*\zs##/#')
  end

  vim.o.wrapscan = saved_wrapscan
  vim.cmd('normal `y')
end
```

Source: vim.fandom.com

1.17 Text Extraction from HTML/URL

Category: advanced_functions
Tags: text-extraction, web-preview, custom-function

Extract text from HTML files or URLs using a text-based browser like elinks, creating a new buffer for preview

Example: Example (Vim)

```
function! ViewHtmlText(url)
  if !empty(a:url)
    new
    setlocal buftype=nofile bufhidden=hide noswapfile
    execute 'r !elinks ' . a:url . ' -dump -dump-width ' . winwidth(0)
    1d
  endif
endfunction

nnoremap <Leader>H :update<Bar>call ViewHtmlText(expand('%:p'))<CR>
```

Example: Example (Neovim)

```
function _G.view_html_text(url)
  if url ~= '' then
    vim.cmd('new')
    vim.api.nvim_buf_set_option(0, 'buftype', 'nofile')
```

```
    vim.api.nvim_buf_set_option(0, 'bufhidden', 'hide')
    vim.api.nvim_buf_set_option(0, 'swapfile', false)

    local output = vim.fn.system('elinks ' .. url .. ' -dump -dump-width '
    ↪  .. vim.fn.winwidth(0))
    vim.api.nvim_buf_set_lines(0, 0, 1, false, vim.split(output, '\n'))
  end
end

vim.keymap.set('n', '<Leader>H', function()
  vim.cmd.update()
  _G.view_html_text(vim.fn.expand('%:p'))
end)
```

Source: vim.fandom.com

CHAPTER 2

Advanced mappings

2.1 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('ia', 'teh', 'the')
vim.keymap.set('ia', '@@', 'your.email@domain.com')
vim.keymap.set('ia', 'dts', function()
    return os.date('%Y-%m-%d')
end, { expr = true })
```

Source: Community contributed

2.2 Access Python Documentation in Vim

Category: advanced_mappings
Tags: python, documentation, key-mapping

Quickly access Python library documentation using pydoc directly from Vim, with support for single words and module methods

Example: Example (Vim)

```
nnoremap <buffer> K :<C-u>let save_isk = &iskeyword |
    \ set iskeyword+=. |
    \ execute "!pydoc " . expand("<cword>") |
    \ let &iskeyword = save_isk<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'K', function()
  local save_isk = vim.o.iskeyword
  vim.o.iskeyword = vim.o.iskeyword .. '.'
  vim.cmd('!pydoc ' .. vim.fn.expand('<cword>'))
  vim.o.iskeyword = save_isk
end, { buffer = true })
```

Source: vim.fandom.com

2.3 Add Blank Line Without Moving Cursor

Category: advanced_mappings
Tags: key-mapping, cursor-position, navigation

Insert a blank line above or below current line while keeping cursor in place

Example: Example (Vim)

```
nnoremap <C-J> m`o<Esc>``
nnoremap <C-K> m`O<Esc>``
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-J>', 'm`o<Esc>``', { desc = 'Insert line below, keep
↪  cursor' })
vim.keymap.set('n', '<C-K>', 'm`O<Esc>``', { desc = 'Insert line above, keep
↪  cursor' })
```

Source: vim.fandom.com

2.4 Advanced CapsLock Key Behavior

Category: advanced_mappings
Tags: key-mapping, custom-behavior, productivity

Create a custom CapsLock key that acts as Escape when tapped, and Ctrl when held with another key

Example: Example (Vim)

```
" No direct Vim implementation
```

Example: Example (Neovim)

```
-- Conceptual Lua implementation would require external utility like
↪  AutoHotkey
-- Demonstrates advanced key remapping potential
```

Source: vim.fandom.com

2.5 Advanced Command and Function Mapping Techniques

Category: advanced_mappings
Tags: command-line, function-mapping, scripting

Demonstrates how to create flexible Vim commands and mappings that can handle ranges, registers, and optional arguments

Example: Example (Vim)

```
fun! TestOne(reg,bang,args) range
  call Dfunc("TestOne(reg<".a:reg.">bang=".a:bang." q-args=".a:args.")
  ↪  firstline=".a:firstline." lastline=".a:lastline)
  call Dret("TestOne")
endfun

com! -range -register -bang TestOne <line1>,<line2>call
↪  TestOne("<reg>",<bang>0,<q-args>)
nnoremap \aa :TestOne<CR>
```

Example: Example (Neovim)

```
function _G.test_one(reg, bang, args, first_line, last_line)
  -- Equivalent logic for processing command arguments
  vim.pretty_print({
    reg = reg,
    bang = bang,
    args = args,
    first_line = first_line,
    last_line = last_line
  })
end

-- Create command using Lua API
vim.api.nvim_create_user_command('TestOne', function(opts)
  _G.test_one(opts.reg or '', opts.bang, opts.args, opts.line1, opts.line2)
end, {
  range = true,
```

```
  register = true,
  bang = true
})

-- Create mapping
vim.keymap.set('n', '\\aa', ':TestOne<CR>')
```

Source: vim.fandom.com

2.6 Advanced HTML Tag Completion

Category: advanced_mappings
Tags: html, auto-insert, mapping

Sophisticated HTML tag completion with auto-indentation and flexible insertion

Example: Example (Vim)

```
function s:CompleteTags()
  inoremap <buffer> > ></<C-x><C-o><Esc>:startinsert!<CR><C-O>?</<CR>
  inoremap <buffer> ><CR>
  ↪  ></<C-x><C-o><Esc>:startinsert!<CR><C-O>?</<CR><CR><Tab><CR><Up><C-O>$
endfunction
autocmd BufRead,BufNewFile *.html,*.js,*.xml call s:CompleteTags()
```

Example: Example (Neovim)

```
local function complete_tags()
  vim.keymap.set('i', '>', function()
    return '>' .. '</' .. vim.fn['complete'](vim.fn.col('.'),
    ↪  vim.fn['compl#omnifunc']())
  end, { buffer = true, expr = true })

  vim.keymap.set('i', '><CR>', function()
    return '>' .. '</' .. vim.fn['complete'](vim.fn.col('.'),
    ↪  vim.fn['compl#omnifunc']()) .. '\n\t\n'
  end, { buffer = true, expr = true })
end

vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = {'*.html', '*.js', '*.xml'},
  callback = complete_tags
})
```

Source: vim.fandom.com

2.7 Advanced Pasted Text Selection with Preserve Mode

Category: advanced_mappings
Tags: text-selection, registers, visual-mode

Dynamically select pasted text while preserving the original visual mode (character, line, or block)

Example: Example (Vim)

```
nnoremap <expr> gp '`[' . strpart(getregtype(), 0, 1) . '`]'
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'gp', function()
  return '`[' .. vim.fn.strpart(vim.fn.getregtype(), 0, 1) .. '`]'
end, { expr = true, desc = 'Select pasted text with original mode' })
```

Source: vim.fandom.com

2.8 Advanced Single Character Insertion with Repeat

Category: advanced_mappings
Tags: key-mapping, editing, repeat

Insert multiple single characters with count support and repeatability

Example: Example (Vim)

```
function! RepeatChar(char, count)
  return repeat(a:char, a:count)
endfunction

nnoremap s :<C-U>exec "normal i".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
nnoremap S :<C-U>exec "normal a".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 's', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.cmd('normal! i' .. string.rep(char, count))
end)

vim.keymap.set('n', 'S', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
```

```
  vim.cmd('normal! a' .. string.rep(char, count))
end)
```

Source: vim.fandom.com

2.9 Advanced Word Swapping Mapping

Category: advanced_mappings
Tags: text-manipulation, key-mapping

Custom mapping to swap words without changing cursor position, supporting complex word exchanges

Example: Example (Vim)

```
nnoremap <silent> gw
↪  "_yiw:s/\(%#\w\+\)\(\W\+\)\(\w\+\)/\3\2\1/<CR><c-o><c-l>:nohlsearch<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'gw', function()
  local current_word = vim.fn.expand('<cword>')
  local next_word = vim.fn.expand('<cword>')
  -- Implement word swapping logic
  -- This is a simplified placeholder
end, { desc = 'Swap current word with next' })
```

Source: vim.fandom.com

2.10 Align #endif with Corresponding #if Directive

Category: advanced_mappings
Tags: preprocessor, code-alignment, mapping

Quickly insert and align #endif with its corresponding #if directive in C/C++ code

Example: Example (Vim)

```
inoremap <buffer> #en X<BS><Esc>?#if<CR>"zy0^Og0"zpDa#endif<CR>X<BS><Esc>?#
↪  end?-1<CR>^"zy0^O0"zpDa
```

Example: Example (Neovim)

```
vim.keymap.set('i', '#en', function()
  -- Simplified implementation of finding and aligning #endif
  local line = vim.api.nvim_get_current_line()
```

```
  local cursor = vim.api.nvim_win_get_cursor(0)

  -- Insert #endif and attempt to align with corresponding #if
  vim.api.nvim_put({'#endif'}, 'l', true, true)
end, { buffer = true })
```

Source: vim.fandom.com

2.11 Alternative Abbreviation Expansion Method

Category: advanced_mappings
Tags: abbreviation, text-manipulation, key-mapping

A more robust method for expanding abbreviations, especially when the abbreviation is at the end of a line

Example: Example (Vim)

```
nno <C-X><C-X> ciw@<Esc>"_s<C-R>"<Esc>b
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-X><C-X>', function()
  -- Change inner word, then paste register content
  vim.cmd('normal! ciw@')
  vim.cmd('normal! "+p')
end, { desc = 'Expand abbreviation at line end' })
```

Source: vim.fandom.com

2.12 Alternative Cursor Position Preservation Mapping

Category: advanced_mappings
Tags: cursor-movement, key-mapping

An alternative method to preserve cursor position after repeating a command, using a different mark technique

Example: Example (Vim)

```
noremap . .'^
```

Example: Example (Neovim)

```
vim.keymap.set('n', '.', function()
  vim.cmd('normal! .')
  vim.cmd('normal! `[')
```

```
end, { desc = 'Repeat last edit with minimal cursor movement' })
```

Source: vim.fandom.com

2.13 Alternative Escape Key Mappings

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Quick alternative ways to exit insert mode without reaching for the Escape key

Example: Example (Vim)

```
" Quick exit insert mode mappings
:imap jj <Esc>
:imap jk <Esc>
:imap kj <Esc>
:imap ;; <Esc>
```

Example: Example (Neovim)

```
-- Alternative escape key mappings in Neovim
vim.keymap.set('i', 'jj', '<Esc>', { noremap = true })
vim.keymap.set('i', 'jk', '<Esc>', { noremap = true })
vim.keymap.set('i', 'kj', '<Esc>', { noremap = true })
vim.keymap.set('i', ';;', '<Esc>', { noremap = true })
```

Source: vim.fandom.com

2.14 Auto-Close Brackets & Navigate Out

Category: advanced_mappings
Tags: insert-mode, auto-completion, brackets

Automatically insert closing brackets and provide easy navigation out of them in insert mode

Example: Example (Vim)

```
inoremap ( ()<Esc>i
inoremap [ []<Esc>i
inoremap { {<CR>}<Esc>O

" Jump out of brackets
inoremap <C-j> <Esc>:call search(BC_GetChar(), "W")<CR>a
```

Example: Example (Neovim)

```
-- Auto-close brackets and navigate
vim.keymap.set('i', '(', '()<Esc>i', { desc = 'Auto-close parentheses' })
vim.keymap.set('i', '[', '[]<Esc>i', { desc = 'Auto-close square brackets'
↪ })
vim.keymap.set('i', '{', '{<CR>}<Esc>O', { desc = 'Auto-close curly braces'
↪ })

-- Navigation out of brackets
vim.keymap.set('i', '<C-j>', '<Esc>/', { desc = 'Navigate out of brackets'
↪ })
```

Source: vim.fandom.com

2.15 Auto-Close Brackets and Braces

Category: advanced_mappings
Tags: auto-completion, editing

Quickly insert matching brackets, parentheses, and braces with cursor positioned inside

Example: Example (Vim)

```
imap () ()<LEFT>
imap [] []<LEFT>
imap {} {}<LEFT>
imap <> <><LEFT>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '(', '()<Left>', { desc = 'Auto-close parentheses' })
vim.keymap.set('i', '[', '[]<Left>', { desc = 'Auto-close square brackets'
↪ })
vim.keymap.set('i', '{', '{}<Left>', { desc = 'Auto-close curly braces' })
vim.keymap.set('i', '<', '<><Left>', { desc = 'Auto-close angle brackets' })
```

Source: vim.fandom.com

2.16 Auto-Close Quotes in HTML/XML Attributes

Category: advanced_mappings
Tags: auto-completion, editing, html, xml

Automatically insert matching quotes and position cursor between them when typing HTML/XML attribute values

Example: Example (Vim)

```
inoremap " ""<LEFT>
inoremap ' ''<LEFT>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '"', '""<Left>', { desc = 'Auto-close double quotes' })
vim.keymap.set('i', "'", "''<Left>", { desc = 'Auto-close single quotes' })
```

Source: vim.fandom.com

2.17 Auto-Generate Function Headers Quickly

Category: advanced_mappings
Tags: key-mapping, text-insertion, productivity

Create a custom function to automatically insert standardized function headers with a single keystroke, saving time and maintaining consistent documentation

2.18 Auto-Insert Closing Characters with Smart Behavior

Category: advanced_mappings
Tags: insert-mode, autocomplete, key-mapping

Automatically insert closing characters like parentheses, braces, and quotes with intelligent handling of different scenarios

Example: Example (Vim)

```
inoremap {      {}<Left>
inoremap {<CR>  {<CR>}<Esc>O
inoremap {{     {
inoremap {}     {}

inoremap (  ()<Left>
inoremap <expr> )  strpart(getline('.'), col('.')-1, 1) == ")" ? "\<Right>"
↪  : ")"
```

Example: Example (Neovim)

```
-- Lua equivalent for closing characters
vim.keymap.set('i', '{', '{}<Left>', { desc = 'Auto-insert closing brace' })
vim.keymap.set('i', '{<CR>', '{<CR>}<Esc>O', { desc = 'Auto-insert closing
↪  brace on new line' })
vim.keymap.set('i', '{{', '{', { desc = 'Prevent double brace insertion' })
vim.keymap.set('i', '{}', '{}', { desc = 'Allow empty braces' })
```

```
vim.keymap.set('i', '(', '()<Left>', { desc = 'Auto-insert closing
↪  parenthesis' })
vim.keymap.set('i', ')', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  return line:sub(col, col) == ')' and '<Right>' or ')'
end, { expr = true, desc = 'Smart closing parenthesis handling' })
```

Source: vim.fandom.com

2.19 Auto-Insert Function Headers

Category: advanced_mappings
Tags: code-generation, documentation, automation

Quickly generate standardized function header comments with a keystroke

Example: Example (Vim)

```
" Shortcut to insert function header template
map <F2> o/**<CR> * Function description<CR> *<CR> * @param <CR> * @return
↪  <CR>*/<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F2>', function()
  local header = {
    '/**',
    ' * Function description',
    ' *',
    ' * @param ',
    ' * @return ',
    ' */',
    ''
  }
  vim.api.nvim_put(header, 'l', true, true)
end, { desc = 'Insert function header' })
```

Source: vim.fandom.com

2.20 Auto-close Brackets and Braces in Insert Mode

Category: advanced_mappings
Tags: auto-completion, insert-mode, key-mapping

Automatically insert closing characters like (), {}, [] with smart behavior for different scenarios

Example: Example (Vim)

```
inoremap {      {}<Left>
inoremap {<CR>  {<CR>}<Esc>O
inoremap {{     {
inoremap {}     {}

inoremap        (  ()<Left>
inoremap <expr> )  strpart(getline('.'), col('.')-1, 1) == ")" ? "\<Right>"
↪  : ")"
```

Example: Example (Neovim)

```
vim.keymap.set('i', '{', '{}<Left>', { desc = 'Auto-insert closing brace' })
vim.keymap.set('i', '{<CR>', '{<CR>}<Esc>O', { desc = 'Auto-insert closing
↪  brace on new line' })

vim.keymap.set('i', '(', '()<Left>', { desc = 'Auto-insert closing
↪  parenthesis' })
vim.keymap.set('i', ')', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  return line:sub(col, col) == ')' and '<Right>' or ')'
end, { expr = true, desc = 'Smart closing parenthesis' })
```

Source: vim.fandom.com

2.21 Auto-continue Python module omnicompletion

Category: advanced_mappings
Tags: python, completion, key-mapping

Automatically trigger omnicompletion after inserting a dot in Python files, improving module completion workflow

Example: Example (Vim)

```
imap <silent> <expr> <buffer> <CR> pumvisible() ?
↪  "<CR><C-R>=(col('.')-1&&match(getline(line('.')), '\.',
      \ col('.')-2) == col('.')-2)?"\<lt>C-X>\<lt>C-O>":""<CR>"
      \ : "<CR>"
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<CR>', function()
  if vim.fn.pumvisible() == 1 then
    local line = vim.fn.getline('.')
    local col = vim.fn.col('.') - 1
    if col > 0 and line:sub(col, col) == '.' then
      return vim.api.nvim_replace_termcodes('<C-X><C-O>', true, false, true)
```

```
    end
  end
  return vim.api.nvim_replace_termcodes('<CR>', true, false, true)
  end,
  { expr = true, buffer = true })
```

Source: vim.fandom.com

2.22 Auto-pair mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('i', '(', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  local char = line:sub(col - 1, col - 1)

  if char:match('%w') then
    return '('
  else
    return '()<Left>'
  end
end, { expr = true })

vim.keymap.set('i', '{', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  local char = line:sub(col - 1, col - 1)

  if char:match('%w') then
    return '{'
  else
    return '{}<Left>'
  end
end, { expr = true })

vim.keymap.set('i', '[', '[]<Left>', { expr = true })
vim.keymap.set('i', '"', '""<Left>', { expr = true })
```

Source: Community contributed

2.23 Automate Java Getter/Setter Generation

Category: advanced_mappings
Tags: java, code-generation, mapping

Create a custom mapping to quickly generate Java getter and setter methods for class variables

Example: Example (Vim)

```
map jgs mawv/ <CR>"ty/ <CR>wvwh"ny/getters<CR>$a<CR><CR><Esc>xxapublic
<Esc>"tpa<Esc>"npbiget<Esc>l~ea()<CR>{<CR><Tab>return
<Esc>"npa;<CR>}<Esc>=<CR><Esc>/setters<CR>$a<CR><CR><Esc>xxapublic void
<Esc>"npbiset<Esc>l~ea(<Esc>"tpa <Esc>"npa)<CR>{<CR><Tab>this.<Esc>"npa=
<Esc>"npa;<CR>}<Esc>=<CR>`ak
```

Example: Example (Neovim)

```
-- Lua equivalent requires more complex implementation
-- Recommended to use a dedicated plugin like 'java_getset.vim'
function generate_getter_setter()
  -- Example implementation would parse current line
  -- Extract variable type and name
  -- Generate getter and setter methods
end

-- Potential key mapping
vim.keymap.set('n', '<leader>jgs', generate_getter_setter, { desc =
↪  'Generate Java Getter/Setter' })
```

Source: vim.fandom.com

2.24 Automatic Closing Characters for Brackets and Quotes

Category: advanced_mappings
Tags: auto-completion, insert-mode, key-mapping

Automatically insert and skip closing characters like parentheses, brackets, and quotes in insert mode, making coding more efficient

Example: Example (Vim)

```
inoremap ( ()<Left>
inoremap <expr> ) strpart(getline('.'), col('.')-1, 1) == ")" ? "\<Right>" :
↪  ")"
```

Example: Example (Neovim)

```
vim.keymap.set('i', '(', '()<Left>', { noremap = true })
vim.keymap.set('i', ')', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  return line:sub(col, col) == ')' and '<Right>' or ')'
end, { expr = true })
```

Source: vim.fandom.com

2.25 Automatic Closing Characters with Smart Insertion

Category: advanced_mappings
Tags: insert-mode, auto-completion, key-mapping

Automatically insert closing brackets, braces, and parentheses with intelligent behavior for different scenarios

Example: Example (Vim)

```
inoremap {      {}<Left>
inoremap {<CR>  {<CR>}<Esc>O
inoremap {{     {
inoremap {}     {}

inoremap        (  ()<Left>
inoremap <expr> )  strpart(getline('.'), col('.')-1, 1) == ")" ? "\<Right>"
 ↪  : ")"
```

Example: Example (Neovim)

```
vim.keymap.set('i', '{', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  vim.fn.feedkeys('{}\27hi', 'n')
end)

vim.keymap.set('i', '(', function()
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  vim.fn.feedkeys('()\27hi', 'n')
end)
```

Source: vim.fandom.com

2.26 Automatic Code Block Commenting

Category: advanced_mappings
Tags: autocomplete, code-generation, mapping

Automatically adds end-of-block comments for code blocks, reducing manual commenting for large functions

Example: Example (Vim)

```
function CurlyBracket()
  let l:startline = line(".")
  let l:result1 = searchpair('{', '', '}', 'bW')
  if (result1 > 0)
    let l:linenum = line(".")
    let l:string1 = substitute(getline(l:linenum), '^\s*\(.*\)\s*$', '\1',
    ↪  "")
    let l:my_string = substitute(l:string1, '\s*{[^{]*$', '', "")
    let l:my_strlen = strlen(l:my_string)
    if (l:my_strlen > 30)
      let l:my_string = strpart(l:my_string,0,30)."..."
    endif

    if ((l:startline - l:linenum) > 10)
      exe "normal a /* " . l:my_string . " */"
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.CurlyBracket()
  local startline = vim.fn.line('.')
  local result1 = vim.fn.searchpair('{', '', '}', 'bW')
  if result1 > 0 then
    local linenum = vim.fn.line('.')
    local string1 = vim.fn.substitute(vim.fn.getline(linenum),
    ↪  '^\\s*\\(.*\\)\\s*$', '\\1', '')
    local my_string = vim.fn.substitute(string1, '\\s*{[^{]*$', '', '')
    local my_strlen = #my_string

    if my_strlen > 30 then
      my_string = string.sub(my_string, 1, 30) .. '...'
    end

    if (startline - linenum) > 10 then
      vim.cmd('normal a /* ' .. my_string .. ' */')
    end
  end
end

-- Add autocmd for specific file types
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = {'*.c', '*.cc', '*.h', '*.cpp', '*.java'},
  callback = function()
    vim.keymap.set('i', '}<CR>', '<Esc>:lua CurlyBracket()<CR>a', { buffer =
    ↪  true })
  end
})
```

Source: vim.fandom.com

2.27 Automatic Comment Mapping for Different Filetypes

Category: advanced_mappings
Tags: comments, filetype, autocmd, mapping

Dynamically set comment/uncomment mappings based on current file type, allowing easy commenting across multiple languages

Example: Example (Vim)

```
function CommentIt()
  if &filetype == "vim"
    vmap +# :s/^/"/<CR>
    vmap -# :s/^"//
  elseif &filetype == "c"
    vmap +# I/*<Esc>gv<End><Esc>a*/<Esc>
    vmap -# I<Esc>2xgv$<Esc>h2x<Esc>
  endif
endfunction

autocmd BufEnter * call CommentIt()
```

Example: Example (Neovim)

```
function _G.comment_it()
  local ft = vim.bo.filetype
  if ft == 'vim' then
    vim.keymap.set('v', '+#', ':s/^/"/<CR>', { buffer = true })
    vim.keymap.set('v', '-#', ':s/^"//
', { buffer = true })
  elseif ft == 'c' then
    vim.keymap.set('v', '+#', 'I/*<Esc>gv<End><Esc>a*/<Esc>', { buffer =
    ↪  true })
    vim.keymap.set('v', '-#', 'I<Esc>2xgv$<Esc>h2x<Esc>', { buffer = true })
  end
end

vim.api.nvim_create_autocmd('BufEnter', {
  callback = _G.comment_it
})
```

Source: vim.fandom.com

2.28 Automatically Create and Update Cscope Database

Category: advanced_mappings
Tags: cscope, code-navigation, project-management

Quickly generate and update cscope database for C/C++ projects, enabling efficient code navigation and symbol lookup

Example: Example (Vim)

```
nmap <F11> :!find . -iname '*.c' -o -iname '*.cpp' -o -iname '*.h' -o -iname
↪  '*.hpp' > cscope.files ;
  \:!cscope -b -i cscope.files -f cscope.out<CR>
  \:cs reset<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F11>', function()
  -- Generate list of source files
  vim.fn.system('find . -iname "*.c" -o -iname "*.cpp" -o -iname "*.h" -o
  ↪  -iname "*.hpp" > cscope.files')

  -- Build cscope database
  vim.fn.system('cscope -b -i cscope.files -f cscope.out')

  -- Reset cscope connection
  vim.cmd('cs reset')
end, { desc = 'Update Cscope Database' })
```

Source: vim.fandom.com

2.29 Avoid Mapping Frequently Used Navigation Keys

Category: advanced_mappings
Tags: key-mapping, performance, navigation

Prevent mapping keys like 'j', 'k', 'l', 'h' at the start of custom mappings to avoid input delays

Example: Example (Vim)

```
" Bad example (avoid)
:nmap jx :somecommand<CR>
" Good practice: use less common prefix
:nmap <leader>jx :somecommand<CR>
```

Example: Example (Neovim)

```
-- Avoid mapping frequently used navigation keys
-- Use a leader key or less common prefix
vim.keymap.set('n', '<leader>jx', ':somecommand<CR>')
```

Source: vim.fandom.com)

2.30 Break Undo Chain with Modifier Keys

Category: advanced_mappings
Tags: undo, key-mapping, editing

Use <C-g>u to create undo breakpoints for more granular undo operations in insert mode

Example: Example (Vim)

```
inoremap <BS> <c-g>u<BS>
inoremap <CR> <c-g>u<CR>
inoremap <del> <c-g>u<del>
inoremap <c-w> <c-g>u<c-w>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<BS>', '<C-g>u<BS>', { desc = 'Undo break on backspace'
↪  })
vim.keymap.set('i', '<CR>', '<C-g>u<CR>', { desc = 'Undo break on enter' })
vim.keymap.set('i', '<Del>', '<C-g>u<Del>', { desc = 'Undo break on delete'
↪  })
vim.keymap.set('i', '<C-w>', '<C-g>u<C-w>', { desc = 'Undo break on word
↪  delete' })
```

Source: vim.fandom.com

2.31 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: Example (Neovim)

```
-- 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,
})
```

Source: Community contributed

2.32 Camel/Snake Case Conversion Functions

Category: advanced_mappings
Tags: text-transformation, custom-function, visual-mode

Create reusable functions to convert case styles that work line-wise and in visual selections

Example: Example (Vim)

```
function! s:Camelize(range) abort
  if a:range == 0
    s#\(\%\(\<\l\+\)\%(_\)\@=\)\|_\(\l\)#\u\1\2#g
  else
    s#\%V\(\%\(\<\l\+\)\%(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g
  endif
endfunction

command! -range CamelCase silent! call <SID>Camelize(<range>)
```

Example: Example (Neovim)

```
local function camelize(range)
  local cmd = range == 0 and
    's#\(\%\(\<\l\+\)\%(_\)\@=\)\|_\(\l\)#\u\1\2#g' or
    's#\%V\(\%\(\<\l\+\)\%(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g'

  vim.cmd(cmd)
end

vim.api.nvim_create_user_command('CamelCase', function(opts)
  camelize(opts.range)
end, { range = true })
```

Source: vim.fandom.com

2.33 Capitalize Words Without Moving Cursor

Category: advanced_mappings
Tags: case-modification, text-editing, mapping

Quick mapping to capitalize words in-place during insert or normal mode without cursor movement

Example: Example (Vim)

```
"Normal mode mappings:
nmap <F7> mzg~iw`z
nmap <F8> mzgUiw`z

"Insert mode mappings:
imap <F8> _<Esc>mzwbgUiw`zi<Del>
```

Example: Example (Neovim)

```
-- Normal mode mappings
vim.keymap.set('n', '<F7>', 'mzg~iw`z', { desc = 'Toggle word case' })
vim.keymap.set('n', '<F8>', 'mzgUiw`z', { desc = 'Capitalize word' })

-- Insert mode mappings
vim.keymap.set('i', '<F8>', '_<Esc>mzwbgUiw`zi<Del>', { desc = 'Capitalize
↪  word in insert mode' })
```

Source: vim.fandom.com

2.34 Clipboard Keymappings for macOS

Category: advanced_mappings
Tags: clipboard, key-mapping, productivity

Create convenient keymappings for copying and pasting using system clipboard

Example: Example (Vim)

```
map <F2> :.w !pbcopy<CR><CR>
map <F3> :r !pbpaste<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F2>', ':.w !pbcopy<CR><CR>', { desc = 'Copy current
↪  line' })
vim.keymap.set('n', '<F3>', ':r !pbpaste<CR>', { desc = 'Paste from
↪  clipboard' })
```

Source: vim.fandom.com

2.35 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('c', '<C-a>', '<Home>')
vim.keymap.set('c', '<C-e>', '<End>')
vim.keymap.set('c', '<C-b>', '<Left>')
vim.keymap.set('c', '<C-f>', '<Right>')
vim.keymap.set('c', '<C-d>', '<Delete>')
```

Source: Community contributed

2.36 Compile File with F7 Keybinding

Category: advanced_mappings
Tags: compilation, key-mapping, build-automation

Add a function to save and compile the current file using F7, with automatic directory handling

Example: Example (Vim)

```
function! Make()
  let curr_dir = expand('%:h')
  if curr_dir == ''
    let curr_dir = '.'
  endif
  execute 'lcd ' . curr_dir
  execute 'make %:r.o'
  execute 'lcd -'
endfunction
nnoremap <F7> :update<CR>:call Make()<CR>
```

Example: Example (Neovim)

```
function _G.make_current_file()
  local curr_dir = vim.fn.expand('%:h')
  if curr_dir == '' then curr_dir = '.' end
  vim.cmd('lcd ' .. curr_dir)
  vim.cmd('make %:r.o')
  vim.cmd('lcd -')
end

vim.keymap.set('n', '<F7>', function()
  vim.cmd('update')
  _G.make_current_file()
end, { desc = 'Save and compile current file' })
```

Source: vim.fandom.com

2.37 Conditional Abbreviation Expansion with Prompt

Category: advanced_mappings
Tags: abbreviation, interactive-expansion, user-prompt

Create an abbreviation that asks for confirmation before expanding, giving users control over automatic text expansion

Example: Example (Vim)

```
function! s:Ask(abbr,expansion,defprompt)
  let answer = confirm("Expand '" . a:abbr . "'?", "&Yes\n&No", a:defprompt)
  return answer == 1 ? a:expansion : a:abbr
endfunction

iabbrev <expr> for <SID>Ask('for', "for () {\n}", 2)
```

Example: Example (Neovim)

```
local function ask_expand(abbr, expansion, defprompt)
  local choice = vim.fn.confirm(string.format("Expand '%s'?", abbr),
  ↪  "&Yes\n&No", defprompt)
  return choice == 1 and expansion or abbr
end

vim.cmd.iabbrev{
  args = {'<expr>', 'for', string.format('<cmd>lua return
  ↪  require"user.utils".ask_expand("for", "for () {\n}", 2)<CR>')}
}
```

Source: vim.fandom.com

2.38 Conditional Closing Brace Insertion

Category: advanced_mappings
Tags: auto-completion, conditional-mapping, insert-mode

Insert closing braces only when the cursor is at the end of the line, preventing unwanted auto-completions

Example: Example (Vim)

```
function! ConditionalPairMap(open, close)
  let line = getline('.')
  let col = col('.')
  if col < col('$') || stridx(line, a:close, col + 1) != -1
    return a:open
  else
    return a:open . a:close . repeat("\<left>", len(a:close))
  endif
endf
inoremap <expr> ( ConditionalPairMap('(', ')')
inoremap <expr> { ConditionalPairMap('{', '}')
```

Example: Example (Neovim)

```
local function conditional_pair_map(open, close)
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  if col < vim.fn.col('$') or line:find(close, col + 1) then
    return open
  else
    return open .. close .. string.rep('<Left>', #close)
  end
end

vim.keymap.set('i', '(', function() return conditional_pair_map('(', ')')
↪  end, { expr = true })
vim.keymap.set('i', '{', function() return conditional_pair_map('{', '}')
↪  end, { expr = true })
```

Source: vim.fandom.com

2.39 Conditional Closing Character Insertion

Category: advanced_mappings
Tags: insert-mode, conditional-mapping, text-editing

Insert closing characters only when cursor is at end of line, preventing unnecessary auto-completion

Example: Example (Vim)

```
function! ConditionalPairMap(open, close)
  let line = getline('.')
  let col = col('.')
  if col < col('$') || stridx(line, a:close, col + 1) != -1
    return a:open
  else
    return a:open . a:close . repeat("\<left>", len(a:close))
  endif
endf
inoremap <expr> ( ConditionalPairMap('(', ')')
inoremap <expr> { ConditionalPairMap('{', '}')
inoremap <expr> [ ConditionalPairMap('[', ']')
```

Example: Example (Neovim)

```
local function conditional_pair_map(open, close)
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  if col < vim.fn.col('$') or line:find(close, col + 1) then
    return open
  else
    return open .. close .. string.rep('\27h', #close)
  end
end

vim.keymap.set('i', '(', function() return conditional_pair_map('(', ')')
↪  end, { expr = true })
vim.keymap.set('i', '{', function() return conditional_pair_map('{', '}')
↪  end, { expr = true })
```

Source: vim.fandom.com

2.40 Conditional Date Abbreviation Expansion

Category: advanced_mappings
Tags: abbreviation, insert-mode, custom-mapping

Create a flexible abbreviation for inserting dates that can be selectively expanded using a custom shortcut key

Example: Example (Vim)

```
iabbrev date^A <c-r>=strftime("%F")<CR>
inoremap <c-b> <c-v><c-a><c-]>
```

Example: Example (Neovim)

```
vim.cmd('iabbrev date^A <c-r>=strftime("%F")<CR>')
vim.keymap.set('i', '<c-b>', '<c-v><c-a><c-]>', { desc = 'Expand date
↪  abbreviation' })
```

Source: vim.fandom.com

2.41 Conditional Key Remapping with AutoHotkey

Category: advanced_mappings
Tags: key-mapping, workflow, conditional-mapping

Create application-specific key mappings that only activate in certain contexts like Vim or VSCode

Example: Example (Vim)

```
" No native Vim equivalent
```

Example: Example (Neovim)

```
-- While Neovim doesn't directly support this, you can use autocmds
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'vim',
  callback = function()
    vim.keymap.set('n', '<CapsLock>', '<Esc>', { buffer = true })
  end
})
```

Source: vim.fandom.com

2.42 Conditional Pair Mapping for Closing Characters

Category: advanced_mappings
Tags: auto-completion, insert-mode, smart-mapping

Insert closing characters only when cursor is at end of line to avoid unnecessary interruptions

Example: Example (Vim)

```
function! ConditionalPairMap(open, close)
  let line = getline('.')
  let col = col('.')
  if col < col('$') || stridx(line, a:close, col + 1) != -1
    return a:open
  else
    return a:open . a:close . repeat("\<left>", len(a:close))
```

```
  endif
endf
inoremap <expr> ( ConditionalPairMap('(', ')')
inoremap <expr> { ConditionalPairMap('{', '}')
inoremap <expr> [ ConditionalPairMap('[', ']')
```

Example: Example (Neovim)

```
local function conditional_pair_map(open, close)
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  if col < vim.fn.col('$') or line:find(close, col + 1) then
    return open
  else
    return open .. close .. string.rep('<Left>', #close)
  end
end

vim.keymap.set('i', '(', function() return conditional_pair_map('(', ')')
↪  end, { expr = true })
vim.keymap.set('i', '{', function() return conditional_pair_map('{', '}')
↪  end, { expr = true })
vim.keymap.set('i', '[', function() return conditional_pair_map('[', ']')
↪  end, { expr = true })
```

Source: vim.fandom.com

2.43 Configure Shift-Tab Key Behavior in Vim

Category: advanced_mappings
Tags: key-mapping, terminal-compatibility

Solve issues with Shift-Tab key mapping across different terminals and systems, enabling consistent key behavior

Example: Example (Vim)

```
" Map Shift-Tab to work correctly
map <Esc>[Z <s-tab>
ounmap <Esc>[Z

" Alternatively, set terminal key code
exe 'set t_kB=' . nr2char(27) . '[Z'
```

Example: Example (Neovim)

```
-- Map Shift-Tab in Neovim
vim.keymap.set({'n', 'i', 'v'}, '<S-Tab>', '<C-d>', { desc = 'Unindent or
↪  navigate back' })
```

```
-- Set terminal key code for compatibility
vim.o.t_kB = vim.fn.nr2char(27) .. '[Z'
```

Source: vim.fandom.com

2.44 Context-Specific Key Remapping with AutoHotkey

Category: advanced_mappings
Tags: key-mapping, conditional-mapping, workflow

Create application-specific key mappings using AutoHotkey, allowing different behaviors in different contexts

Example: Example (Neovim)

```
-- Neovim equivalent using lua conditional mapping
-- Map CapsLock to Escape only in Neovim/terminal modes
vim.keymap.set({'n', 't'}, '<CapsLock>', '<Esc>', {
  desc = 'Escape in normal and terminal modes',
  buffer = true  -- Makes mapping buffer-local
})
```

Source: vim.fandom.com

2.45 Context-aware mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
    pattern = 'python',
    callback = function()
        vim.keymap.set('n', '<F5>', '<Cmd>python %<CR>', { buffer = true })
    end
})
vim.api.nvim_create_autocmd('FileType', {
```

```
    pattern = 'javascript',
    callback = function()
        vim.keymap.set('n', '<F5>', '<Cmd>node %<CR>', { buffer = true })
    end
})
vim.api.nvim_create_autocmd('FileType', {
    pattern = 'sh',
    callback = function()
        vim.keymap.set('n', '<F5>', '<Cmd>bash %<CR>', { buffer = true })
    end
})
```

Source: Community contributed

2.46 Contextual F1 Help with Smart Behavior

Category: advanced_mappings
Tags: key-mapping, help, context-aware

Intelligent F1 mapping that opens help for word under cursor or closes help buffer

Example: Example (Vim)

```
function! MapF1()
  if &buftype == "help"
    exec 'quit'
  else
    exec 'help'
  endif
endfunction
noremap <F1> :call MapF1()<CR>
```

Example: Example (Neovim)

```
function _G.smart_help()
  if vim.bo.buftype == "help" then
    vim.cmd('quit')
  else
    vim.cmd('help')
  end
end

vim.keymap.set('n', '<F1>', _G.smart_help)
```

Source: vim.fandom.com

2.47 Convenient Macro Playback Mapping

Category: advanced_mappings
Tags: key-mapping, macros, productivity

Map space key to quickly replay a macro from the q register

Example: Example (Vim)

```
" Map space to replay macro in q register
:nnoremap <Space> @q
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Space>', '@q', { desc = 'Replay macro from q register'
↪  })
```

Source: vim.fandom.com

2.48 Convert Quoted Printable Email to Plain Text

Category: advanced_mappings
Tags: email, text-conversion, key-mapping

Quick mapping to decode quoted printable encoded emails in Vim, converting special encodings to readable text

Example: Example (Vim)

```
nnoremap <Leader>Q :%s/=\(\x\x\<BAR>\n\)/\=submatch(1)=='
'?'':nr2char('0x'.submatch(1))/ge<CR>
vnoremap <Leader>Q :s/=\(\x\x\<BAR>\n\)/\=submatch(1)=='
'?'':nr2char('0x'.submatch(1))/ge<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>Q', function()
  vim.cmd([[%s/=\(\x\x\|\n\)/\=submatch(1) == '\n' ? '' : nr2char('0x' .
  ↪  submatch(1))/ge]])
end, { desc = 'Convert Quoted Printable to Plain Text' })

vim.keymap.set('v', '<Leader>Q', function()
  vim.cmd([[s/=\(\x\x\|\n\)/\=submatch(1) == '\n' ? '' : nr2char('0x' .
  ↪  submatch(1))/ge]])
end, { desc = 'Convert Selected Quoted Printable to Plain Text' })
```

Source: vim.fandom.com

2.49 Copy and Replace Words or Lines Repeatedly

Category: advanced_mappings
Tags: registers, copy-paste, text-manipulation

Efficiently copy and replace text at multiple locations using yanking and dot command

Example: Example (Vim)

```
# Copy word: yiw
# Replace word: ciw<C-R>0<Esc>
# Repeat replacements with '.'
```

Example: Example (Neovim)

```
-- Copy word: 'yiw'
-- Replace word: 'ciw' .. vim.fn.getreg('0') .. '<Esc>'
-- Use '.' to repeat replacements
-- Leverage vim.fn.getreg() to access registers in Lua
```

Source: vim.fandom.com

2.50 Create Aligned Comment Boxes Quickly

Category: advanced_mappings
Tags: comment, mapping, productivity

Quickly insert a formatted comment box that spans 100 columns, useful for section headers or documentation

Example: Example (Vim)

```
map ,co O#====================================================================⌋
 ↪  ==================================<CR>#<CR>#==========================⌋
 ↪  ==========================================================================⌋
 ↪  ===<Esc>100\|Dkk100\|DjA
```

Example: Example (Neovim)

```
vim.keymap.set('n', ',co', function()
  -- Create a 100-column comment box
  vim.api.nvim_put({'#' .. string.rep('=', 100), '#', '#' .. string.rep('=',
   ↪  100)}, 'l', true, true)
  vim.cmd('normal! kk100|D')
end, { desc = 'Insert comment box' })
```

Source: vim.fandom.com

2.51 Create Context-Aware Mappings for GUI and Console

Category: advanced_mappings
Tags: key-mapping, conditional-mapping, cross-platform

Dynamically create mappings that adapt between GUI and console Vim based on the current environment, reducing configuration duplication

Example: Example (Vim)

```
function! ModeMapping(guiLhs, termLhs, rhs, ...)
  let mapCommand='map'
  if (a:0 > 0)
    let mapCommand=a:1
  endif
  if (has("gui_running"))
    echo mapCommand . " " . a:guiLhs . " " . a:rhs
  else
    echo mapCommand . " " . a:termLhs . " " . a:rhs
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.mode_mapping(gui_lhs, term_lhs, rhs, map_command)
  map_command = map_command or 'map'
  if vim.g.gui_running then
    vim.cmd(map_command .. ' ' .. gui_lhs .. ' ' .. rhs)
  else
    vim.cmd(map_command .. ' ' .. term_lhs .. ' ' .. rhs)
  end
end
```

Source: vim.fandom.com

2.52 Create Custom Keymaps for Language Mapping

Category: advanced_mappings
Tags: key-mapping, internationalization, custom-keymap

Create custom keymaps for different languages, allowing translation of keyboard inputs with flexible mapping options

Example: Example (Vim)

```
" Example keymap for Czech with UTF-8
let b:keymap_name = "cz"
highlight lCursor ctermbg=red guibg=red

loadkeymap
```

```
" Key mappings go here
A <Char-0x00C1>  " Map A to Á
```

Example: Example (Neovim)

```
-- Lua equivalent for custom keymap
vim.g.keymap_name = "cz"
vim.cmd.highlight("lCursor ctermbg=red guibg=red")

-- Note: Full keymap loading requires Vim script
```

Source: vim.fandom.com

2.53 Create Custom Language Keymaps

Category: advanced_mappings
Tags: key-mapping, internationalization, language-input

Create custom keymaps for different languages or special character inputs, with support for UTF-8 and multiple encodings

Example: Example (Vim)

```
" Keymap file structure example
let b:keymap_name="cz"
highlight lCursor ctermbg=red guibg=red

loadkeymap
" Mapping examples
sz ß  " German eszet mapping
:a ä   " Umlaut mapping
```

Example: Example (Neovim)

```
-- Lua equivalent for creating custom keymaps
-- Note: Neovim keymap creation is more flexible
vim.g.keymap_name = "cz"
vim.api.nvim_set_hl(0, 'lCursor', { bg = 'red' })

-- Custom language mappings can be set using vim.keymap.set
-- with buffer-local option for specific language inputs
```

Source: vim.fandom.com

2.54 Create Custom Option Toggle Function

Category: advanced_mappings
Tags: key-mapping, function, customization

Create a reusable function to toggle specific options with a key mapping

Example: Example (Vim)

```
function! ToggleMouse()
  if &mouse == 'a'
    set mouse=
    echo "Mouse usage disabled"
  else
    set mouse=a
    echo "Mouse usage enabled"
  endif
endfunction

nnoremap <F12> :call ToggleMouse()<CR>
```

Example: Example (Neovim)

```
function _G.toggle_mouse()
  if vim.o.mouse == 'a' then
    vim.o.mouse = ''
    print('Mouse usage disabled')
  else
    vim.o.mouse = 'a'
    print('Mouse usage enabled')
  end
end

vim.keymap.set('n', '<F12>', _G.toggle_mouse, { desc = 'Toggle mouse
↪  support' })
```

Source: vim.fandom.com

2.55 Create Custom Text Objects in Vim/Neovim

Category: advanced_mappings
Tags: text-objects, custom-mapping, operator-pending

Demonstrates how to create custom text objects for selecting and operating on specific regions like folds or indent levels

Example: Example (Vim)

```
" Visual mode mapping for 'a fold'
vnoremap af :<C-U>silent! normal! [zV]z<CR>

" Operator-pending mode mapping
omap af :normal Vaf<CR>
```

Example: Example (Neovim)

```
-- Visual mode mapping for 'a fold'
vim.keymap.set('v', 'af', function()
  vim.cmd.silent('normal! [zV]z')
end, { desc = 'Select entire fold' })

-- Operator-pending mode mapping
vim.keymap.set('o', 'af', function()
  vim.cmd.normal('Vaf')
end, { desc = 'Operate on entire fold' })
```

Source: vim.fandom.com

2.56 Create Granular Undo Breakpoints in Insert Mode

Category: advanced_mappings
Tags: undo, insert-mode, key-mapping

Add breakpoints in insert mode to allow undoing changes at a more granular level, specifically per line or logical chunk

Example: Example (Vim)

```
inoremap <CR> <C-G>u<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<CR>', '<C-G>u<CR>', { desc = 'Create undo breakpoint
↪  on line break' })
```

Source: vim.fandom.com

2.57 Create Mode-Specific Key Mappings

Category: advanced_mappings
Tags: key-mapping, modes, customization

Create targeted key mappings for specific Vim editing modes, allowing more precise and context-aware shortcuts

Example: Example (Vim)

```
" Normal mode mapping
nnoremap <F2> :lchdir %:p:h<CR>:pwd<CR>

" Insert mode mapping
imap <F3> <C-R>=strftime('%c')<CR>
```

Example: Example (Neovim)

```
-- Normal mode mapping
vim.keymap.set('n', '<F2>', ':lchdir %:p:h<CR>:pwd<CR>', { desc = 'Change to
↪  current file directory' })

-- Insert mode mapping
vim.keymap.set('i', '<F3>', function() return vim.fn.strftime('%c') end, {
↪  expr = true })
```

Source: vim.fandom.com

2.58 Create Mode-Specific Key Mappings in Vim

Category: advanced_mappings
Tags: key-mapping, mode-specific, configuration

Learn how to create key mappings that work only in specific Vim modes like normal, insert, visual, etc. This allows for more precise and targeted key configurations.

Example: Example (Vim)

```
" Normal mode mapping
nnoremap <F2> :lchdir %:p:h<CR>:pwd<CR>

" Insert mode mapping
imap <F3> <C-R>=strftime('%c')<CR>
```

Example: Example (Neovim)

```
-- Normal mode mapping
vim.keymap.set('n', '<F2>', ':lchdir %:p:h<CR>:pwd<CR>', { desc = 'Change to
↪  current file directory' })

-- Insert mode mapping
vim.keymap.set('i', '<F3>', function()
  return vim.fn.strftime('%c')
end, { expr = true, desc = 'Insert current timestamp' })
```

Source: vim.fandom.com

2.59 Create Recursive Vim Mappings for Repetitive Editing

Category: advanced_mappings
Tags: macro, mapping, automation, editing

Recursive mappings allow you to create powerful, self-repeating editing commands that can automatically process multiple lines or perform complex transformations

Example: Example (Vim)

```
" Example 1: Recursive mapping to process file names
:map z Iwc <Esc>lyawA><Esc>pa.log<CR>echo "HelloWorld"<Esc>jz

" Example 2: Recursive mapping to increment numbers
:map z 2^Ajz
```

Example: Example (Neovim)

```
-- Lua implementation requires more explicit handling
-- This demonstrates the concept rather than direct translation
function RecursiveFileProcessing()
  -- Move to start of line
  vim.cmd('normal! Iwc ')
  -- Copy word
  vim.cmd('normal! lyaw')
  -- Append log and echo
  vim.cmd('normal! A> echo "HelloWorld"')
  -- Move to next line
  vim.cmd('normal! j')
end

-- Can be mapped using more modern Neovim mapping
vim.keymap.set('n', 'z', RecursiveFileProcessing)
```

Source: vim.fandom.com

2.60 Create Unique Plugin Mappings with <Plug>

Category: advanced_mappings
Tags: key-mapping, plugins, customization

Use <Plug> to create unique, user-configurable key mappings in plugins that won't conflict with existing mappings

Example: Example (Vim)

```
noremap <unique> <Plug>ScriptFunc :call <SID>VimScriptFn()<CR>
nmap _p <Plug>ScriptFunc
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Plug>ScriptFunc', function() -- implement VimScriptFn
↪  end, { unique = true })
vim.keymap.set('n', '_p', '<Plug>ScriptFunc')
```

Source: vim.fandom.com)

2.61 Create a Function to Toggle Options

Category: advanced_mappings
Tags: key-mapping, function, toggle

Dynamically toggle options like mouse usage with a function and key mapping

Example: Example (Vim)

```
nnoremap <F12> :call ToggleMouse()<CR>
function! ToggleMouse()
  if &mouse == 'a'
    set mouse=
    echo "Mouse usage disabled"
  else
    set mouse=a
    echo "Mouse usage enabled"
  endif
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F12>', function()
  if vim.opt.mouse:get() == 'a' then
    vim.opt.mouse = ''
    print('Mouse usage disabled')
  else
    vim.opt.mouse = 'a'
    print('Mouse usage enabled')
  end
end, { desc = 'Toggle mouse usage' })
```

Source: vim.fandom.com

2.62 Cross-Platform Clipboard Key Mappings

Category: advanced_mappings
Tags: key-mapping, clipboard, productivity

Add familiar clipboard shortcuts for copy, cut, and paste in insert and visual modes

Example: Example (Vim)

```
" Clipboard shortcuts
:inoremap <C-v> <ESC>"+pa
:vnoremap <C-c> "+y
:vnoremap <C-d> "+d
```

Example: Example (Neovim)

```
-- Clipboard shortcuts
vim.keymap.set('i', '<C-v>', '<ESC>"+pa')
vim.keymap.set('v', '<C-c>', '"+y')
vim.keymap.set('v', '<C-d>', '"+d')
```

Source: vim.fandom.com

2.63 Ctrl-S: Smart File Saving Shortcut

Category: advanced_mappings
Tags: key-mapping, file-operations, productivity

Create a custom Ctrl-S mapping that saves existing files or opens a file browser for new files, working in multiple modes

Example: Example (Vim)

```
" Smart save command
command -nargs=0 -bar Update if &modified
    if empty(bufname('%'))
        browse confirm write
    else
        confirm write
    endif
endif

" Normal mode mapping
nnoremap <silent> <C-S> :<C-u>Update<CR>

" Insert mode mapping
inoremap <c-s> <c-o>:Update<CR>
```

Example: Example (Neovim)

```
-- Create Update command
vim.api.nvim_create_user_command('Update', function()
    if vim.bo.modified then
        if vim.api.nvim_buf_get_name(0) == '' then
            vim.cmd('browse confirm write')
        else
            vim.cmd('confirm write')
        end
    end
end, {})

-- Key mappings
vim.keymap.set('n', '<C-S>', '<Cmd>Update<CR>', { silent = true })
vim.keymap.set('i', '<C-S>', '<C-o>:Update<CR>', { silent = true })
```

Source: vim.fandom.com

2.64 Custom Alt Key Mappings for Special Characters

Category: advanced_mappings
Tags: key-mapping, custom-bindings

Create Alt key combinations to quickly insert special characters or symbols that might be hard to type directly

Example: Example (Vim)

```
" Alt key character mappings
" <M-`> 0
" <M-q> \
" <M-w> |
" <M-f> [
" <M-g> ]
```

Example: Example (Neovim)

```
-- Alt key character mappings
vim.keymap.set('n', '<M-`>', '0', { desc = 'Insert 0' })
vim.keymap.set('n', '<M-q>', '\\', { desc = 'Insert backslash' })
vim.keymap.set('n', '<M-w>', '|', { desc = 'Insert pipe' })
vim.keymap.set('n', '<M-f>', '[', { desc = 'Insert left bracket' })
vim.keymap.set('n', '<M-g>', ']', { desc = 'Insert right bracket' })
```

Source: vim.fandom.com

2.65 Custom AutoCAD MTEXT Editing Menu in Vim

Category: advanced_mappings
Tags: integration, custom-menu, external-tools

Create a custom menu for AutoCAD MTEXT editing with special character and formatting insertions

Example: Example (Vim)

```
"AutoCad menu for MTEXT editation
"Menu for inserting special AutoCAD text formatting codes

imenu &AutoCad.Insert.Space \~
vmenu &AutoCad.Insert.Space <Esc>`<i\~<Esc>%

imenu &AutoCad.Colour.Red \C1;
vmenu &AutoCad.Colour.Red <Esc>`>a\C7;<Esc>`<i\C1;<Esc>%
```

Example: Example (Neovim)

```
-- Lua equivalent for custom AutoCAD MTEXT menu
local function create_autocad_menu()
  vim.api.nvim_create_user_command('AutoCadInsertSpace', function()
    vim.cmd('normal! i~')
  end, {})

  vim.api.nvim_create_user_command('AutoCadInsertRedColor', function()
    vim.cmd('normal! i\C1;')
  end, {})
end

-- Call the function to set up the menu
create_autocad_menu()
```

Source: vim.fandom.com

2.66 Custom Backspace Function for Edge Cases

Category: advanced_mappings
Tags: key-mapping, custom-function, editing

Advanced backspace handling that works at line boundaries and preserves editing context

Example: Example (Vim)

```
func Backspace()
  if col('.') == 1
    if line('.') != 1
      return "\<ESC>Dk$p\<S-J>i"
    else
      return ""
    endif
  else
    return "\<Left>\<Del>"
  endif
endfunc

inoremap <BS> <c-r>=Backspace()<CR>
```

Example: Example (Neovim)

```
function _G.custom_backspace()
  local col = vim.fn.col('.')
  local line = vim.fn.line('.')

  if col == 1 then
    if line ~= 1 then
      return vim.api.nvim_replace_termcodes('<ESC>Dk$p<S-J>i', true, false,
      ↪  true)
```

```
    else
      return ''
    end
  else
    return vim.api.nvim_replace_termcodes('<Left><Del>', true, false, true)
  end
end

vim.keymap.set('i', '<BS>', 'v:lua.custom_backspace()', { expr = true })
```

Source: vim.fandom.com

2.67 Custom Folding Key Mappings

Category: advanced_mappings
Tags: folding, key-mapping, navigation

Remap Shift-Left/Right to close and open folds, simplifying fold navigation

Example: Example (Vim)

```
nnoremap <S-Left> zo
inoremap <S-Left> <C-O>zo
nnoremap <S-Right> zc
inoremap <S-Right> <C-O>zc
```

Example: Example (Neovim)

```
vim.keymap.set({'n', 'i'}, '<S-Left>', 'zo', { desc = 'Open fold' })
vim.keymap.set({'n', 'i'}, '<S-Right>', 'zc', { desc = 'Close fold' })
```

Source: vim.fandom.com

2.68 Custom Jump List Navigation Function

Category: advanced_mappings
Tags: navigation, custom-function, jump-list

Create a custom function to interactively select and navigate jump list locations

Example: Example (Vim)

```
function! GotoJump()
  jumps
  let j = input("Please select your jump: ")
  if j != ''
    let pattern = '\v\c^\+'
    if j =~ pattern
```

```
      let j = substitute(j, pattern, '', 'g')
      execute "normal " . j . "\<c-i>"
    else
      execute "normal " . j . "\<c-o>"
    endif
  endif
endfunction

" Optional mapping
nmap <Leader>j :call GotoJump()<CR>
```

Example: Example (Neovim)

```
function _G.goto_jump()
  vim.cmd('jumps')
  local j = vim.fn.input('Please select your jump: ')
  if j ~= '' then
    local pattern = '^+'
    if j:match(pattern) then
      j = j:gsub('^+', '')
      vim.cmd('normal! ' .. j .. '\<c-i>')
    else
      vim.cmd('normal! ' .. j .. '\<c-o>')
    end
  end
end

-- Set up mapping
vim.keymap.set('n', '<Leader>j', _G.goto_jump, { desc = 'Navigate jump list'
 ↪  })
```

Source: vim.fandom.com

2.69 Custom Jump List Selection Function

Category: advanced_mappings
Tags: jump-list, custom-function, navigation

Interactive function to select and jump to specific locations in jump list

Example: Example (Vim)

```
function! GotoJump()
  jumps
  let j = input("Please select your jump: ")
  if j != ''
    let pattern = '\v\c^\+'
    if j =~ pattern
      let j = substitute(j, pattern, '', 'g')
      execute "normal " . j . "\<c-i>"
    else
```

```
      execute "normal " . j . "\<c-o>"
    endif
  endif
endfunction

" Map to leader key
nmap <Leader>j :call GotoJump()<CR>
```

Example: Example (Neovim)

```
function _G.goto_jump()
  vim.cmd('jumps')
  local j = vim.fn.input('Please select your jump: ')
  if j ~= '' then
    if j:match('^+') then
      j = j:gsub('^+', '')
      vim.cmd('normal! ' .. j .. '\<C-i>')
    else
      vim.cmd('normal! ' .. j .. '\<C-o>')
    end
  end
end

-- Map to leader key
vim.keymap.set('n', '<Leader>j', _G.goto_jump, { desc = 'Interactive jump
↪  list navigation' })
```

Source: vim.fandom.com

2.70 Custom Search Text Object for Older Vim

Category: advanced_mappings
Tags: search, mapping, text-objects

Create a custom 's' mapping to easily copy, change, or operate on search matches in Vim versions before 7.4

Example: Example (Vim)

```
" Search text object mappings
vnoremap <silent> s //e<C-r>=&selection=='exclusive'?'+1':''<CR><CR>
    \:<C-u>call histdel('search',-1)<Bar>let @/=histget('search',-1)<CR>gv
omap s :normal vs<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent (requires vim.keymap.set)
vim.keymap.set('v', 's', function()
  -- Implement search text object logic
  -- Note: More complex to replicate exact Vimscript behavior
```

```
  -- Recommend using built-in gn in Neovim 0.7+
end)
```

Source: vim.fandom.com

2.71 Custom Toolbar URL Launcher

Category: advanced_mappings
Tags: toolbar, custom-mapping, windows

Replace existing toolbar icon to launch selected URL in browser using visual mode

Example: Example (Vim)

```
vnoremenu 1.140 ToolBar.New "wy:!start explorer <C-R>w<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '<ToolBar>New', function()
  local selected_text = vim.fn.getreg('"')
  vim.fn.system('start explorer ' .. selected_text)
end)
```

Source: vim.fandom.com

2.72 Cycle Numeric Options Intelligently

Category: advanced_mappings
Tags: configuration, key-mapping, options

Create a function to cycle numeric options between predefined min/max values with a specified increment

Example: Example (Vim)

```
function CycleNum(option,min,inc,max)
  exec ('let tz_value = (((&'.a:option.'-'.a:min.')+'.a:inc.')%(('.a:max.'-
  ↪  '.a:min.')+'.a:inc.'))+'.a:min)
  if (tz_value < a:min) " in case inc<0
    let tz_value = tz_value+a:max
  endif
  exec ('setlocal '.a:option.'='.tz_value)
endfunction

" Cycle tabstop
noremap <silent> <M-t>s :call CycleNum("tabstop",4,4,8)<BAR>set tabstop?<CR>
```

Example: Example (Neovim)

```
function _G.cycle_num(option, min, inc, max)
  local current = vim.api.nvim_get_option(option)
  local next_val = ((current - min + inc) % (max - min + inc)) + min
  if next_val < min then
    next_val = next_val + max
  end
  vim.api.nvim_set_option(option, next_val)
  print(string.format('%s is now %d', option, next_val))
end

-- Cycle tabstop
vim.keymap.set('n', '<M-t>s', function() _G.cycle_num('tabstop', 4, 4, 8)
↪  end)
```

Source: vim.fandom.com

2.73 Debug Code Abbreviations for PHP

Category: advanced_mappings
Tags: debugging, abbreviation, php

Quick insert of debug print statements and variable inspection

Example: Example (Vim)

```
iab phpb exit("<hr>Debug ");
iab phpv echo "<hr><pre>";var_dump($a);exit("debug ");
iab phpallv print_r(get_defined_vars());
```

Example: Example (Neovim)

```
vim.cmd([[iabbrev phpb exit("<hr>Debug ");]])
vim.cmd([[iabbrev phpv echo "<hr><pre>";var_dump($a);exit("debug ");]])
vim.cmd([[iabbrev phpallv print_r(get_defined_vars());]])
```

Source: vim.fandom.com)

2.74 Diagnose and Clear Problematic Key Mappings

Category: advanced_mappings
Tags: key-mapping, debugging, troubleshooting

Check and remove unintended insert mode mappings for backspace key

Example: Example (Vim)

```
" Check backspace mappings
:verbose imap <BS>
:verbose imap ^H

" Clear problematic mapping
:iunmap <BS>
```

Example: Example (Neovim)

```
-- Check backspace mappings
vim.cmd('verbose imap <BS>')
vim.cmd('verbose imap ^H')

-- Clear problematic mapping
vim.keymap.del('i', '<BS>')
```

Source: vim.fandom.com

2.75 Diagnose and Fix Backspace Key Mappings

Category: advanced_mappings
Tags: troubleshooting, key-mapping, insert-mode

Check for existing backspace mappings and remove them if they're causing issues

Example: Example (Vim)

```
" Check backspace mappings
:verbose imap <BS>

" Remove problematic mapping
:iunmap <BS>
```

Example: Example (Neovim)

```
-- Check backspace mappings
vim.cmd('verbose imap <BS>')

-- Remove problematic mapping
vim.keymap.del('i', '<BS>')
```

Source: vim.fandom.com

2.76 Disable Accidentally Triggered F1 Help Key

Category: advanced_mappings
Tags: key-mapping, usability, keyboard-shortcut

Prevent accidental F1 key presses from opening help, which can be disruptive during editing

Example: Example (Vim)

```
" Disable F1 help key in normal and insert modes
:nmap <F1> <nop>
:imap <F1> <Esc>
```

Example: Example (Neovim)

```
-- Disable F1 help key in Neovim
vim.keymap.set('n', '<F1>', '<Nop>')
vim.keymap.set('i', '<F1>', '<Esc>')
```

Source: vim.fandom.com

2.77 Disable Alt Key Menu Shortcuts in Vim

Category: advanced_mappings
Tags: key-mapping, configuration, input-handling

Disable Alt key menu shortcuts to allow custom Alt key mappings, which is useful for creating alternative key bindings

Example: Example (Vim)

```
set winaltkeys=no
```

Example: Example (Neovim)

```
vim.o.winaltkeys = 'no'
```

Source: vim.fandom.com

2.78 Disable Middle Mouse Button Paste

Category: advanced_mappings
Tags: mouse, clipboard, input-prevention

Prevent accidental middle mouse button pastes in Vim/Neovim by mapping the middle mouse button to do nothing

Example: Example (Vim)

```
:map <MiddleMouse> <Nop>
:imap <MiddleMouse> <Nop>

:map <2-MiddleMouse> <Nop>
:imap <2-MiddleMouse> <Nop>
:map <3-MiddleMouse> <Nop>
:imap <3-MiddleMouse> <Nop>
:map <4-MiddleMouse> <Nop>
:imap <4-MiddleMouse> <Nop>
```

Example: Example (Neovim)

```
-- Disable middle mouse button paste in all modes
vim.keymap.set({'n', 'i', 'v', 'x', 's', 'c', 't'}, '<MiddleMouse>',
↪  '<Nop>')
vim.keymap.set({'n', 'i', 'v', 'x', 's', 'c', 't'}, '<2-MiddleMouse>',
↪  '<Nop>')
vim.keymap.set({'n', 'i', 'v', 'x', 's', 'c', 't'}, '<3-MiddleMouse>',
↪  '<Nop>')
vim.keymap.set({'n', 'i', 'v', 'x', 's', 'c', 't'}, '<4-MiddleMouse>',
↪  '<Nop>')
```

Source: vim.fandom.com

2.79 Disable Unwanted Built-in Vim Commands

Category: advanced_mappings
Tags: key-mapping, customization

Easily disable built-in Vim commands that you find annoying or accidentally trigger

Example: Example (Vim)

```
:map K <Nop>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'K', '<Nop>', { desc = 'Disable built-in command' })
```

Source: vim.fandom.com

2.80 Disable Unwanted Vim Key Mappings

Category: advanced_mappings
Tags: key-mapping, configuration, customization

Quickly disable built-in Vim key commands that you find annoying or accidentally trigger

Example: Example (Vim)

```
:map K <Nop>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'K', '<Nop>', { desc = 'Disable built-in K command' })
```

Source: vim.fandom.com

2.81 Dynamic Commenting Operator for Different Filetypes

Category: advanced_mappings
Tags: commenting, text-manipulation, operators

Create a flexible commenting/uncommenting operator that automatically detects comment syntax based on filetype

Example: Example (Vim)

```
function! CommentStr()
  if &ft == 'cpp' || &ft == 'java' || &ft == 'javascript'
    return '//'
  elseif &ft == 'vim'
    return '"'
  elseif &ft == 'python' || &ft == 'perl' || &ft == 'sh' || &ft == 'R'
    return '#'
  elseif &ft == 'lisp'
    return ';'
  endif
  return ''
endfunction

nnoremap <Leader>c <Esc>:set opfunc=DoCommentOp<CR>g@
nnoremap <Leader>C <Esc>:set opfunc=UnCommentOp<CR>g@
```

Example: Example (Neovim)

```
local function get_comment_string()
  local ft = vim.bo.filetype
  local comment_map = {
    cpp = '//', java = '//', javascript = '///',
    vim = '"', python = '#', perl = '#',
    sh = '#', R = '#', lisp = ';'
  }
  return comment_map[ft] or ''
end
```

```
-- Similar mapping logic would be implemented using vim.keymap.set
-- This demonstrates the core concept of dynamic comment detection
```

Source: vim.fandom.com

2.82 Dynamic File Preview in Index/Listing

Category: advanced_mappings
Tags: file-preview, custom-mapping, dynamic-loading

Create a custom mapping to quickly preview files from an index or listing, with version-aware implementation

Example: Example (Vim)

```
nnoremap <buffer> <Space> :exec "let
↪  alist=readfile(expand('%:p:h').'/'.substitute(getline('.'),
↪  '\(^.*\|\s*\)\|\(\s\s*$\)', '', 'g'))\|echo join(alist,\"\n\")"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Space>', function()
  local filename = vim.fn.expand('%:p:h') .. '/' ..
  ↪  vim.fn.substitute(vim.fn.getline('.'), '\(^.*\|\s*\)\|\(\s\s*$\)', '',
  ↪  'g')
  local lines = vim.fn.readfile(filename)
  print(table.concat(lines, '\n'))
end, { buffer = true })
```

Source: vim.fandom.com

2.83 Dynamic Key Handler with Multiple Contexts

Category: advanced_mappings
Tags: key-mapping, function-handler, conditional-execution

Create a flexible key mapping that can execute different handlers based on context, allowing dynamic behavior for a single key press

Example: Example (Vim)

```
function! RunHandlers(handler_list, default_action)
  for handler in a:handler_list
    execute "let handled = " handler."()"
    if handled
      return
    endif
```

```
  endfor
  execute a:default_action
endfunction

" Example handler
function! CloseScratch()
  if &buftype == "nofile" && &bufhidden == "hide" && !&swapfile
    bdelete
    return 1
  endif
  return 0
endfunction

" Setup multiple handlers for Escape key
let g:esc_handler = []
call add(g:esc_handler, "CloseScratch")
nnoremap <Esc> :call RunHandlers(g:esc_handler, "call DefaultEsc()")<CR>
```

Example: Example (Neovim)

```
function RunHandlers(handler_list, default_action)
  for _, handler in ipairs(handler_list) do
    local handled = _G[handler]()
    if handled then
      return
    end
  end
  vim.cmd(default_action)
end

function CloseScratch()
  if vim.bo.buftype == "nofile" and vim.bo.bufhidden == "hide" and not
  ↪  vim.bo.swapfile then
    vim.cmd('bdelete')
    return true
  end
  return false
end

-- Setup multiple handlers for Escape key
local esc_handlers = {"CloseScratch"}
vim.keymap.set('n', '<Esc>', function()
  RunHandlers(esc_handlers, 'normal! \<Esc>')
end)
```

Source: vim.fandom.com

2.84 Dynamic Search Mode Cycling

Category: advanced_mappings
Tags: search, custom-mapping, productivity

Create a custom F4 mapping to cycle through different search result display modes

Example: Example (Vim)

```
function! s:SearchMode()
  if !exists('s:searchmode') || s:searchmode == 0
    nnoremap <silent> n n:call <SID>MaybeMiddle()<CR>
    let s:searchmode = 1
  elseif s:searchmode == 1
    nnoremap n nzz
    let s:searchmode = 2
  else
    nunmap n
    let s:searchmode = 0
  endif
endfunction
```

Example: Example (Neovim)

```
local function search_mode()
  local modes = {
    [0] = function()
      vim.keymap.set('n', 'n', function()
        local winline = vim.fn.winline()
        local winheight = vim.fn.winheight(0)
        if winline == 1 or winline == winheight then
          vim.cmd('normal! zz')
        end
        vim.cmd('normal! n')
      end)
    end,
    [1] = function()
      vim.keymap.set('n', 'n', 'nzz')
    end,
    [2] = function()
      vim.keymap.del('n', 'n')
    end
  }

  local current_mode = vim.g.search_mode or 0
  local next_mode = (current_mode + 1) % 3

  modes[next_mode]()
  vim.g.search_mode = next_mode
end

-- Bind to F4
vim.keymap.set('n', '<F4>', search_mode)
```

Source: vim.fandom.com

2.85 Easily Capitalize Words and Regions

Category: advanced_mappings
Tags: text-manipulation, case-conversion, editing

Provides custom mappings to capitalize words, lines, and visual selections with intuitive gc (capitalize) commands

Example: Example (Vim)

```
if (&tildeop)
  nmap gcw guw~l
  nmap gcW guW~l
  nmap gciw guiw~l
  nmap gciW guiW~l
  nmap gcis guis~l
  nmap gc$ gu$~l
  nmap gcgc guu~l
  nmap gcc guu~l
  vmap gc gu~l
else
  nmap gcw guw~h
  nmap gcW guW~h
  nmap gciw guiw~h
  nmap gciW guiW~h
  nmap gcis guis~h
  nmap gc$ gu$~h
  nmap gcgc guu~h
  nmap gcc guu~h
  vmap gc gu~h
endif
```

Example: Example (Neovim)

```
-- Capitalize every word on current line
vim.keymap.set('n', 'gcc', function()
  vim.cmd('s/\v<(.)((\w*))>/\u\1\L\2/g')
end)

-- Capitalize words in visual selection
vim.keymap.set('x', 'gc', function()
  vim.cmd('s/\%V\v<(.)((\w*))>/\u\1\L\2/g')
end)
```

Source: vim.fandom.com

2.86 Easily Move Lines Up or Down

Category: advanced_mappings
Tags: navigation, editing, key-mapping

Provides intuitive Alt+j/k mappings to move lines up or down in normal, insert, and visual modes, with automatic re-indentation

Example: Example (Vim)

```
nnoremap <A-j> :m .+1<CR>==
nnoremap <A-k> :m .-2<CR>==
inoremap <A-j> <Esc>:m .+1<CR>==gi
inoremap <A-k> <Esc>:m .-2<CR>==gi
vnoremap <A-j> :m '>+1<CR>gv=gv
vnoremap <A-k> :m '<-2<CR>gv=gv
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<A-j>', ':m .+1<CR>==', { noremap = true, silent = true
↪  })
vim.keymap.set('n', '<A-k>', ':m .-2<CR>==', { noremap = true, silent = true
↪  })
vim.keymap.set('i', '<A-j>', '<Esc>:m .+1<CR>==gi', { noremap = true, silent
↪  = true })
vim.keymap.set('i', '<A-k>', '<Esc>:m .-2<CR>==gi', { noremap = true, silent
↪  = true })
vim.keymap.set('v', '<A-j>', ':m ">+1<CR>gv=gv', { noremap = true, silent =
↪  true })
vim.keymap.set('v', '<A-k>', ':m "<-2<CR>gv=gv', { noremap = true, silent =
↪  true })
```

Source: vim.fandom.com

2.87 Easy Block Selection with Alt+Left Mouse

Category: advanced_mappings
Tags: mouse-selection, key-mapping, visual-mode

Enable block selection using Alt+Left Mouse in Vim, similar to other editors

Example: Example (Vim)

```
noremap <M-LeftMouse> <4-LeftMouse>
inoremap <M-LeftMouse> <4-LeftMouse>
onoremap <M-LeftMouse> <C-C><4-LeftMouse>

noremap <M-LeftDrag> <LeftDrag>
inoremap <M-LeftDrag> <LeftDrag>
onoremap <M-LeftDrag> <C-C><LeftDrag>
```

Example: Example (Neovim)

```
-- Enable block selection with Alt+Left Mouse
vim.keymap.set({'n', 'i', 'o'}, '<M-LeftMouse>', '<4-LeftMouse>', { noremap
↪  = true })
vim.keymap.set({'n', 'i', 'o'}, '<M-LeftDrag>', '<LeftDrag>', { noremap =
↪  true })
```

Source: vim.fandom.com

2.88 Easy French Character Input Mappings

Category: advanced_mappings
Tags: internationalization, text-input, character-mapping

Create intuitive mappings for quickly inserting French accented characters in insert and command modes

Example: Example (Vim)

```
" French character mappings
map! ;z à
map! ;a â
map! ;b ä
map! ;c ç
map! ;d è
map! ;e ê
map! ;f ë
map! ;g é
imap ;q «  »<Esc>hi
```

Example: Example (Neovim)

```
-- French character mappings for Neovim
vim.keymap.set({'i', 'c'}, ';z', 'à', { desc = 'Insert à' })
vim.keymap.set({'i', 'c'}, ';a', 'â', { desc = 'Insert â' })
vim.keymap.set({'i', 'c'}, ';c', 'ç', { desc = 'Insert ç' })
vim.keymap.set('i', ';q', '«  »<Esc>hi', { desc = 'Insert French quotation
↪  marks' })
```

Source: vim.fandom.com

2.89 Easy French Character Mappings

Category: advanced_mappings
Tags: internationalization, character-mapping, input-method

Provides a systematic way to input French accented characters using simple key combinations in insert and command modes

Example: Example (Vim)

```
" French character mappings
map! ;z à
map! ;a â
map! ;b ä
map! ;c ç
map! ;d è
map! ;e ê
map! ;f ë
map! ;g é
imap ;q «  »<Esc>hi
```

Example: Example (Neovim)

```
-- French character mappings in Neovim
vim.keymap.set({'i', 'c'}, ';z', 'à', { desc = 'Insert à' })
vim.keymap.set({'i', 'c'}, ';a', 'â', { desc = 'Insert â' })
vim.keymap.set({'i', 'c'}, ';c', 'ç', { desc = 'Insert ç' })
vim.keymap.set('i', ';q', '«  »<Esc>hi', { desc = 'Insert French quotation
↪  marks' })
```

Source: vim.fandom.com

2.90 Enhanced Block Selection with Matchit

Category: advanced_mappings
Tags: selection, text-objects, code-navigation

Use extended matching and text object selection for more flexible code navigation and editing

Example: Example (Vim)

```
" Install matchit.vim
runtime macros/matchit.vim
```

Example: Example (Neovim)

```
-- For Neovim, use a plugin like vim-matchup
-- Enables advanced % key navigation and text object selection
```

Source: vim.fandom.com

2.91 Enhanced Mouse Interaction in Vim

Category: advanced_mappings
Tags: mouse, visual-mode, key-mapping

Improve mouse functionality in Vim, adding block selection and paste behavior similar to other text editors

Example: Example (Vim)

```
"Mouse visual block selection (like MS Word)
set mouse=ra

"For GUI Vim
if has("gui_running")
  nmap <A-LeftMouse> ms<LeftMouse><C-v>`so
  imap <A-LeftMouse> <Esc><C-v>`^ms<Esc>gi<LeftMouse><C-o><C-v>`so
else
  "Paste toggle in console Vim
  nmap <F7> :set paste! paste?<CR>
  set pastetoggle=<F7> mouse=rnv
endif
```

Example: Example (Neovim)

```
-- Enhanced mouse interaction in Neovim
vim.opt.mouse = 'ra'

-- GUI-specific mappings
if vim.g.neovide or vim.g.nvui then
  vim.keymap.set({'n', 'i'}, '<A-LeftMouse>', function()
    vim.cmd('normal! ms`[v`]')
  end)
end

-- Paste toggle
vim.keymap.set({'n', 'i', 'v'}, '<F7>', function()
  vim.o.paste = not vim.o.paste
  print('Paste mode: ' .. (vim.o.paste and 'ON' or 'OFF'))
end)
```

Source: vim.fandom.com

2.92 Execute External Commands with Key Mapping

Category: advanced_mappings
Tags: key-mapping, external-commands, workflow

Map a key to run external commands quickly, with automatic execution by adding <CR>

Example: Example (Vim)

```
map <F6> :!p4 edit %<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F6>', function() vim.cmd('!p4 edit %') end, { desc =
↪  'Run external command on current file' })
```

Source: vim.fandom.com

2.93 Execute Files and URLs from Buffer Text

Category: advanced_mappings
Tags: windows, file-operations, key-mapping

Quickly open files, URLs, or execute commands directly from the current buffer text on Windows

Example: Example (Vim)

```
" Execute current file
nmap \xf :silent !start rundll32 url.dll,FileProtocolHandler %:p <CR>

" Execute string under cursor
nmap \x :silent !start rundll32 url.dll,FileProtocolHandler <cWORD> <CR>

" Execute string with current file's path
nmap \xl :silent !start rundll32 url.dll,FileProtocolHandler %:p:h/<cWORD>
↪  <CR>
```

Example: Example (Neovim)

```
-- Execute current file
vim.keymap.set('n', '<leader>xf', function()
  vim.fn.system('rundll32 url.dll,FileProtocolHandler ' ..
  ↪  vim.fn.expand('%:p'))
end, { silent = true })

-- Execute string under cursor
vim.keymap.set('n', '<leader>x', function()
  vim.fn.system('rundll32 url.dll,FileProtocolHandler ' ..
  ↪  vim.fn.expand('<cWORD>'))
end, { silent = true })

-- Execute string with current file's path
vim.keymap.set('n', '<leader>xl', function()
  vim.fn.system('rundll32 url.dll,FileProtocolHandler ' ..
  ↪  vim.fn.expand('%:p:h') .. '/' .. vim.fn.expand('<cWORD>'))
end, { silent = true })
```

Source: vim.fandom.com

2.94 Execute Files and URLs from Buffer Text in Windows

Category: advanced_mappings
Tags: windows, file-execution, key-mapping

Create mappings to quickly open files, URLs, or execute commands directly from buffer text in Vim/Neovim on Windows

Example: Example (Vim)

```
" eXecute File being edited
nmap \xf :silent !start rundll32 url.dll,FileProtocolHandler %:p <CR>

"eXecute string below cursor
nmap \x :silent !start rundll32 url.dll,FileProtocolHandler <cWORD> <CR>

" eXecute string below cursor after prepending it with path to file
nmap \xl :silent !start rundll32 url.dll,FileProtocolHandler %:p:h/<cWORD>
 ↪  <CR>
```

Example: Example (Neovim)

```
-- Execute current file
vim.keymap.set('n', '\xf', function()
  vim.cmd.silent('!start rundll32 url.dll,FileProtocolHandler ' ..
   ↪  vim.fn.expand('%:p'))
end, { desc = 'Open current file' })

-- Execute word under cursor
vim.keymap.set('n', '\x', function()
  vim.cmd.silent('!start rundll32 url.dll,FileProtocolHandler ' ..
   ↪  vim.fn.expand('<cWORD>'))
end, { desc = 'Execute word under cursor' })

-- Execute word with current file path
vim.keymap.set('n', '\xl', function()
  vim.cmd.silent('!start rundll32 url.dll,FileProtocolHandler ' ..
   ↪  vim.fn.expand('%:p:h') .. '/' .. vim.fn.expand('<cWORD>'))
end, { desc = 'Execute word with file path' })
```

Source: vim.fandom.com

2.95 Execute Normal Mode Commands from Insert Mode

Category: advanced_mappings
Tags: insert-mode, key-mapping, command-execution

Quickly execute a normal mode command accidentally typed in insert mode by using a custom mapping that switches to normal mode, undoes the last insertion, and executes the command

Example: Example (Vim)

```
inoremap <somekey> <Esc>u@.

autocmd CursorHoldI * call feedkeys("\<C-G>u", 'tn')
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<somekey>', '<Esc>u@.', { desc = 'Execute normal mode
↪  command from insert mode' })

vim.api.nvim_create_autocmd('CursorHoldI', {
  callback = function()
    vim.fn.feedkeys('\<C-G>u', 'tn')
  end
})
```

Source: vim.fandom.com

2.96 Exit Insert Mode Without Moving Cursor

Category: advanced_mappings
Tags: insert-mode, cursor-position, key-mapping

Provides a way to exit insert mode without the default cursor movement behavior, using a quick key sequence

Example: Example (Vim)

```
set timeoutlen=300
inoremap fj <Esc>l
inoremap jf <Esc>l
```

Example: Example (Neovim)

```
vim.opt.timeoutlen = 300
vim.keymap.set('i', 'fj', '<Esc>l')
vim.keymap.set('i', 'jf', '<Esc>l')
```

Source: vim.fandom.com

2.97 Expand Existing Abbreviation Quickly

Category: advanced_mappings
Tags: abbreviation, key-mapping, text-expansion

Provides a quick way to expand existing abbreviations in Vim, allowing for dynamic text replacement

Example: Example (Vim)

```
map <C-X><C-X> diw:exe "normal i".@"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-X><C-X>', function()
  -- Delete current word and replace with its expanded version
  vim.cmd('normal! diw""P')
end, { desc = 'Expand existing abbreviation' })
```

Source: vim.fandom.com

2.98 Explicit Undo Chain Break with <End>

Category: advanced_mappings
Tags: undo, key-mapping, editing

Map <End> key to explicitly break the undo chain in insert mode

Example: Example (Vim)

```
inoremap <End> <C-g>u<End>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<End>', '<C-g>u<End>', { desc = 'Break undo chain at
↪  end of line' })
```

Source: vim.fandom.com

2.99 Expression mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<Tab>', function()
    return vim.fn.pumvisible() == 1 and '<C-n>' or '<Tab>'
end, { expr = true })
vim.keymap.set('i', '<CR>', function()
    return vim.fn.pumvisible() == 1 and '<C-y>' or '<CR>'
end, { expr = true })
vim.keymap.set('n', 'n', function()
    return vim.v.searchforward == 1 and 'n' or 'N'
end, { expr = true })
```

Source: Community contributed

2.100 Fast Keycode Mapping in Terminal Vim

Category: advanced_mappings
Tags: key-mapping, terminal, performance

Efficiently map complex key combinations in terminal Vim by using timeout settings and unused function keys to resolve terminal keycode ambiguity

Example: Example (Vim)

```
set timeout timeoutlen=1000 ttimeoutlen=100
if !has("gui_running")
  if &term == "xterm"
    set <F13>=^[[1;6B
    map <F13> <C-S-Down>
    map! <F13> <C-S-Down>
endif
```

Example: Example (Neovim)

```
vim.o.timeout = true
vim.o.timeoutlen = 1000
vim.o.ttimeoutlen = 100

if not vim.g.neovide and vim.fn.has('gui_running') == 0 then
  if vim.o.term == 'xterm' then
    vim.keymap.set({'n', 'i'}, '<F13>', '<C-S-Down>', { noremap = true })
  end
end
```

Source: vim.fandom.com

2.101 Find Unused Keys for Custom Mappings

Category: advanced_mappings
Tags: key-mapping, configuration, vim-tips

Learn how to identify and use unused key sequences for custom mappings without conflicting with existing Vim functionality

Example: Example (Vim)

```
" Check existing maps
:map
:map!
" Check verbose mapping details
:verbose map ,
```

Example: Example (Neovim)

```
-- Check existing maps
vim.cmd('map')
vim.cmd('map!')
-- Check verbose mapping details
vim.cmd('verbose map ,')
```

Source: vim.fandom.com)

2.102 Fix Meta-Keys in Terminal Vim

Category: advanced_mappings
Tags: key-mapping, terminal-config, meta-keys

Resolves issues with Meta (Alt) keys breaking out of Insert mode in terminal Vim by configuring escape sequences

Example: Example (Vim)

```
" fix meta-keys which generate <Esc>a .. <Esc>z
let c='a'
while c <= 'z'
  exec "set <M-".toupper(c).">=\e".c
  exec "imap \e".c." <M-".toupper(c).">"
  let c = nr2char(1+char2nr(c))
endw
```

Example: Example (Neovim)

```
-- Fix Meta-keys in Neovim
for i = string.byte('a'), string.byte('z') do
  local char = string.char(i)
  vim.keymap.set({'n', 'i'}, string.format('<M-%s>', string.upper(char)),
  ↪  function()
    -- Custom meta-key handling
  end)
end
```

Source: vim.fandom.com

2.103 Fix x Command in Virtual Edit Mode

Category: advanced_mappings
Tags: key-mapping, virtual-edit, editing

Modify the x command to jump to end of line when in virtual edit mode, ensuring consistent character deletion behavior

Example: Example (Vim)

```
" redefine x for virtualEdit so that past end of line, it jumps left to
↪  end-of-line
function! Redefine_x_ForVirtualEdit()
  if &ve != "" && col('.') >= col('$')
    normal $
  endif
endfu!
silent! unmap x
:nnoremap <silent>x x:call Redefine_x_ForVirtualEdit()<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('RedefineXForVirtualEdit', function()
  if vim.o.virtualedit ~= '' and vim.fn.col('.') >= vim.fn.col('$') then
    vim.cmd('normal! $')
  end
end, {})

vim.keymap.set('n', 'x', function()
  vim.cmd('normal! x')
  vim.cmd.RedefineXForVirtualEdit()
end, { silent = true })
```

Source: vim.fandom.com

2.104 Flexible Caps Lock Mapping with AutoHotkey

Category: advanced_mappings
Tags: key-mapping, customization

Advanced key mapping to use Caps Lock as Ctrl when held, and Escape when tapped

Example: Example (Vim)

```
" No direct Vimscript implementation
```

Example: Example (Neovim)

```
-- Conceptual implementation
-- Requires external tool like AutoHotkey
-- Provides context-aware key remapping
```

Source: vim.fandom.com

2.105 Flexible Case Conversion Commands

Category: advanced_mappings
Tags: text-manipulation, custom-commands, regex

Create custom commands for converting case within entire buffer or visual selection

Example: Example (Vim)

```
function! s:Camelize(range) abort
  if a:range == 0
    s#\(\%(\<\l\+\)\%(_\)\@=\)\|_\(\l\)#\u\1\2#g
  else
    s#\%V\(\%(\<\l\+\)\%(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g
  endif
endfunction

command! -range CamelCase silent! call <SID>Camelize(<range>)
```

Example: Example (Neovim)

```
local function camelize(range)
  local cmd = range == 0 and
    's#\(\%(\<\l\+\)\%(_\)\@=\)\|_\(\l\)#\u\1\2#g' or
    's#\%V\(\%(\<\l\+\)\%(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g'

  vim.cmd(cmd)
end

vim.api.nvim_create_user_command('CamelCase', function(opts)
  camelize(opts.range)
end, { range = true })
```

Source: vim.fandom.com

2.106 Flexible Insert Mode Command Execution

Category: advanced_mappings
Tags: key-mapping, command-execution, productivity

Multiple approaches to running commands in insert mode: multiple Ctrl-O, using | for multiple commands, or defining a custom function

Example: Example (Vim)

```
:map <F5> :set number!<CR>
:imap <F5> <c-o><F5>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', ':set number!<CR>')
vim.keymap.set('i', '<F5>', '<c-o><F5>', { desc = 'Execute normal mode F5
↪  from insert mode' })
```

Source: vim.fandom.com

2.107 Flexible Mappings for GUI and Console Vim

Category: advanced_mappings
Tags: key-mapping, conditional-mapping, cross-environment

Create a function that dynamically sets different key mappings for GUI and console Vim, reducing configuration complexity

Example: Example (Vim)

```
function! ModeMapping(guiLhs, termLhs, rhs, ...)
  let mapCommand='map'
  if (a:0 > 0)
    let mapCommand=a:1
  endif
  if (has("gui_running"))
    execute mapCommand . " " . a:guiLhs . " " . a:rhs
  else
    execute mapCommand . " " . a:termLhs . " " . a:rhs
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.ModeMapping(guiLhs, termLhs, rhs, mapCommand)
  mapCommand = mapCommand or 'map'
  if vim.g.gui_running then
    vim.cmd(mapCommand .. ' ' .. guiLhs .. ' ' .. rhs)
  else
    vim.cmd(mapCommand .. ' ' .. termLhs .. ' ' .. rhs)
  end
end
```

Source: vim.fandom.com

2.108 Generate Code Templates with Vim

Category: advanced_mappings
Tags: code-generation, text-templates, productivity

Create powerful text templates with variable substitution using a custom Perl script to generate repetitive code structures quickly

Example: Example (Vim)

```
vnoremap <F6> :!perl E:\Devtools\vim\vimfiles\template\truler.pl<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '<F6>', function()
  -- Example implementation would require adapting the Perl script
  -- or creating a similar Lua/Neovim function for template processing
  vim.cmd('!perl E:\\Devtools\\vim\\vimfiles\\template\\truler.pl')
end, { desc = 'Process code template' })
```

Source: vim.fandom.com

2.109 Generate PHP Accessor and Setter Methods

Category: advanced_mappings
Tags: php, code-generation, mapping

Quickly generate getter and setter methods for class variables using Vim mappings

Example: Example (Vim)

```
" Public Accessors
map <F3> :s/\(\(\w\)\(\w\+\)\).*/public function get\u\2\3(){
        return \$this->\1;
}/<CR>
" Public Setters
map <S-F3> :s/\(\(\w\)\(\w\+\)\).*/public function
 ↪  set\u\2\3(\$\1){\r        \$this->\1 = \$\1;\r}/<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for generating PHP accessors/setters
local function generate_getter()
  vim.cmd('s/\(\(\w\)\(\w\+\)\).*/public function get\u\2\3(){
        return $this->' .. vim.fn.expand('<cword>') .. ';
}/')
end

local function generate_setter()
```

```
  vim.cmd('s/\(\(\w\)\(\w\+\)\).*/public function set\u\2\3($' ..
  ↪  vim.fn.expand('<cword>') .. '){
        $this->' .. vim.fn.expand('<cword>') .. ' = $' ..
        ↪  vim.fn.expand('<cword>') .. ';
}/')
end

-- Set keymaps
vim.keymap.set('n', '<F3>', generate_getter, { desc = 'Generate PHP Getter'
↪  })
vim.keymap.set('n', '<S-F3>', generate_setter, { desc = 'Generate PHP
↪  Setter' })
```

Source: vim.fandom.com

2.110 Handle Numeric Keypad Escape Sequences

Category: advanced_mappings
Tags: key-mapping, terminal, input-handling

Map escape sequences from numeric keypad to corresponding numbers and operations

Example: Example (Vim)

```
" Mapping escape sequences to numeric keypad keys
:inoremap <Esc>Oq 1
:inoremap <Esc>Or 2
:inoremap <Esc>Os 3
:inoremap <Esc>Ot 4
:inoremap <Esc>Ou 5
:inoremap <Esc>Ov 6
:inoremap <Esc>Ow 7
:inoremap <Esc>Ox 8
:inoremap <Esc>Oy 9
:inoremap <Esc>Op 0
```

Example: Example (Neovim)

```
-- Lua equivalent for handling numeric keypad escape sequences
local function map_keypad_sequences()
  vim.keymap.set('i', '<Esc>Oq', '1', { noremap = true })
  vim.keymap.set('i', '<Esc>Or', '2', { noremap = true })
  vim.keymap.set('i', '<Esc>Os', '3', { noremap = true })
  vim.keymap.set('i', '<Esc>Ot', '4', { noremap = true })
  vim.keymap.set('i', '<Esc>Ou', '5', { noremap = true })
  vim.keymap.set('i', '<Esc>Ov', '6', { noremap = true })
  vim.keymap.set('i', '<Esc>Ow', '7', { noremap = true })
  vim.keymap.set('i', '<Esc>Ox', '8', { noremap = true })
  vim.keymap.set('i', '<Esc>Oy', '9', { noremap = true })
  vim.keymap.set('i', '<Esc>Op', '0', { noremap = true })
end
```

```
-- Call the function to set up mappings
map_keypad_sequences()
```

Source: vim.fandom.com

2.111 Handle Shift-Tab for Unindent and Completion

Category: advanced_mappings
Tags: key-mapping, input, terminal-compatibility

Configure Shift-Tab to work correctly across different terminals and use it for unindenting or tab completion

Example: Example (Vim)

```
" Map Shift-Tab to unindent or tab completion
:map <Esc>[Z <s-tab>
:ounmap <Esc>[Z

" More robust method
:exe 'set t_kB=' . nr2char(27) . '[Z'
```

Example: Example (Neovim)

```
-- Lua equivalent for handling Shift-Tab
vim.keymap.set({'n', 'i', 'v'}, '<S-Tab>', function()
  -- Implement unindent or tab completion logic
  vim.cmd('normal! <<')
end, { desc = 'Unindent or tab completion' })
```

Source: vim.fandom.com

2.112 Improve Completion Popup Menu Navigation

Category: advanced_mappings
Tags: completion, key-mapping, insert-mode

Enhanced completion menu navigation with intuitive key mappings for selecting and accepting completions

Example: Example (Vim)

```
inoremap <expr> <Esc>      pumvisible() ? "\<C-e>" : "\<Esc>"
inoremap <expr> <CR>       pumvisible() ? "\<C-y>" : "\<CR>"
inoremap <expr> <Down>     pumvisible() ? "\<C-n>" : "\<Down>"
inoremap <expr> <Up>       pumvisible() ? "\<C-p>" : "\<Up>"
```

```
inoremap <expr> <PageDown> pumvisible() ? "\<PageDown>\<C-p>\<C-n>" :
 ↪  "\<PageDown>"
inoremap <expr> <PageUp>   pumvisible() ? "\<PageUp>\<C-p>\<C-n>" :
 ↪  "\<PageUp>"
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<Esc>', function()
  return vim.fn.pumvisible() == 1 and '<C-e>' or '<Esc>'
end, { expr = true })

vim.keymap.set('i', '<CR>', function()
  return vim.fn.pumvisible() == 1 and '<C-y>' or '<CR>'
end, { expr = true })

vim.keymap.set('i', '<Down>', function()
  return vim.fn.pumvisible() == 1 and '<C-n>' or '<Down>'
end, { expr = true })

vim.keymap.set('i', '<Up>', function()
  return vim.fn.pumvisible() == 1 and '<C-p>' or '<Up>'
end, { expr = true })

vim.keymap.set('i', '<PageDown>', function()
  return vim.fn.pumvisible() == 1 and '<PageDown><C-p><C-n>' or '<PageDown>'
end, { expr = true })

vim.keymap.set('i', '<PageUp>', function()
  return vim.fn.pumvisible() == 1 and '<PageUp><C-p><C-n>' or '<PageUp>'
end, { expr = true })
```

Source: vim.fandom.com

2.113 Insert Empty Line in Normal Mode

Category: advanced_mappings
Tags: normal-mode, line-manipulation, productivity

Quickly insert an empty line above or below the current line without entering insert mode

Example: Example (Vim)

```
" Insert line below
nmap ,o o<Esc>k
" Insert line above
nmap ,O O<Esc>j
```

Example: Example (Neovim)

```
vim.keymap.set('n', ',o', 'o<Esc>k', { desc = 'Insert line below' })
vim.keymap.set('n', ',O', 'O<Esc>j', { desc = 'Insert line above' })
```

Source: vim.fandom.com

2.114 Insert Mode Alt Key Mappings for Quick Editing

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

A set of Alt key mappings in insert mode to quickly perform normal mode actions without leaving insert mode, reducing mode switching overhead

Example: Example (Vim)

```
" Basic motion mappings in insert mode

imap <A-h> <Left>
imap <A-j> <Down>
imap <A-k> <Up>
imap <A-l> <Right>

" Other useful insert mode mappings
imap <A-o> <C-o>
imap <A-w> <C-o>:w<CR>
```

Example: Example (Neovim)

```
-- Lua equivalents for Alt key mappings
vim.keymap.set('i', '<A-h>', '<Left>', { desc = 'Move left in insert mode'
 ↪  })
vim.keymap.set('i', '<A-j>', '<Down>', { desc = 'Move down in insert mode'
 ↪  })
vim.keymap.set('i', '<A-k>', '<Up>', { desc = 'Move up in insert mode' })
vim.keymap.set('i', '<A-l>', '<Right>', { desc = 'Move right in insert mode'
 ↪  })

vim.keymap.set('i', '<A-o>', '<C-o>', { desc = 'Execute single normal mode
 ↪  command' })
vim.keymap.set('i', '<A-w>', '<C-o>:w<CR>', { desc = 'Save file from insert
 ↪  mode' })
```

Source: vim.fandom.com

2.115 Insert Multiple Blank Lines Easily

Category: advanced_mappings
Tags: key-mapping, editing, productivity

Quickly insert multiple blank lines with a customizable mapping that ensures at least one blank line before and after

Example: Example (Vim)

```
" Open multiple lines (insert empty lines) before or after current line
function! OpenLines(nrlines, dir)
  let nrlines = a:nrlines < 3 ? 3 : a:nrlines
  let start = line('.') + a:dir
  call append(start, repeat([''], nrlines))
  if a:dir < 0
    normal! 2k
  else
    normal! 2j
  endif
endfunction

" Mappings to open multiple lines and enter insert mode
nnoremap <Leader>o :<C-u>call OpenLines(v:count, 0)<CR>S
nnoremap <Leader>O :<C-u>call OpenLines(v:count, -1)<CR>S
```

Example: Example (Neovim)

```
-- Function to insert multiple blank lines
function _G.open_lines(nrlines, dir)
  nrlines = nrlines < 3 and 3 or nrlines
  local start = vim.fn.line('.') + dir

  -- Insert blank lines
  local blank_lines = vim.fn['repeat']({''}. nrlines)
  vim.fn.append(start, blank_lines)

  -- Adjust cursor position
  if dir < 0 then
    vim.cmd('normal! 2k')
  else
    vim.cmd('normal! 2j')
  end
end

-- Key mappings
vim.keymap.set('n', '<Leader>o', function()
  local count = vim.v.count > 0 and vim.v.count or 1
  _G.open_lines(count, 0)
  vim.cmd('startinsert')
end, { desc = 'Insert blank lines below' })

vim.keymap.set('n', '<Leader>O', function()
  local count = vim.v.count > 0 and vim.v.count or 1
  _G.open_lines(count, -1)
  vim.cmd('startinsert')
end, { desc = 'Insert blank lines above' })
```

Source: vim.fandom.com

2.116 Insert Space Without Moving Cursor

Category: advanced_mappings
Tags: text-alignment, normal-mode, cursor-preservation

Insert a space at the current cursor position without moving the cursor, useful for manual text alignment

Example: Example (Vim)

```
nnoremap <Space> i<Space><Esc>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Space>', 'i<Space><Esc>', { desc = 'Insert space at
↪  cursor' })
```

Source: vim.fandom.com

2.117 Insert-Mode Only Caps Lock Toggle

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Create a Caps Lock functionality that only works in insert mode, toggled by Ctrl-^

Example: Example (Vim)

```
" Toggle insert-mode only Caps Lock
for c in range(char2nr('A'), char2nr('Z'))
  execute 'lnoremap ' . nr2char(c+32) . ' ' . nr2char(c)
  execute 'lnoremap ' . nr2char(c) . ' ' . nr2char(c+32)
endfor

" Automatically turn off when leaving insert mode
autocmd InsertLeave * set iminsert=0
```

Example: Example (Neovim)

```
-- Toggle insert-mode only Caps Lock
for i = string.byte('a'), string.byte('z') do
  local lower = string.char(i)
  local upper = string.char(i - 32)
  vim.keymap.set('l', lower, upper, { noremap = true })
  vim.keymap.set('l', upper, lower, { noremap = true })
end

-- Automatically turn off when leaving insert mode
vim.api.nvim_create_autocmd('InsertLeave', {
```

```
  callback = function()
    vim.o.iminsert = 0
  end
})
```

Source: vim.fandom.com

2.118 Invert Number Row Keys for Faster Typing

Category: advanced_mappings
Tags: key-mapping, productivity, typing

Swap number row keys to type symbols more easily without using Shift key, potentially improving typing speed and comfort

Example: Example (Vim)

```
" map each number to its shift-key character
" Insert mode mappings
inoremap 1 !
inoremap 2 @
inoremap 3 #
inoremap 4 $
inoremap 5 %
inoremap 6 ^
inoremap 7 &
inoremap 8 *
inoremap 9 (
inoremap 0 )
inoremap - _

" Reverse mappings
inoremap ! 1
inoremap @ 2
inoremap # 3
inoremap $ 4
inoremap % 5
inoremap ^ 6
inoremap & 7
inoremap * 8
inoremap ( 9
inoremap ) 0
inoremap _ -
```

Example: Example (Neovim)

```
-- Lua equivalent for number row key inversion
local function setup_key_inversion()
  local maps = {
    ['1'] = '!', ['2'] = '@', ['3'] = '#', ['4'] = '$', ['5'] = '%',
    ['6'] = '^', ['7'] = '&', ['8'] = '*', ['9'] = '(', ['0'] = ')',
```

```
    ['-'] = '_'
  }

  for k, v in pairs(maps) do
    vim.keymap.set('i', k, v)
    vim.keymap.set('i', v, k)
  end
end

setup_key_inversion()
```

Source: vim.fandom.com

2.119 Invoke Function with Count Prefix

Category: advanced_mappings
Tags: function-mapping, custom-commands, count-prefix

Create a mapping that allows invoking a function with a numeric count prefix, similar to built-in Vim commands

Example: Example (Vim)

```
function! Foo(count)
  echo 'FOO: ' . a:count
endfunction

command! -nargs=1 FooCmd call Foo(<args>)
map ,a :<C-U>FooCmd(v:count)<CR>
```

Example: Example (Neovim)

```
function _G.foo(count)
  print(string.format('FOO: %d', count))
end

vim.api.nvim_create_user_command('FooCmd', function(opts)
  _G.foo(tonumber(opts.args))
end, { nargs = 1 })

vim.keymap.set('n', ',a', function()
  local count = vim.v.count > 0 and vim.v.count or 1
  vim.cmd('FooCmd ' .. count)
end)
```

Source: vim.fandom.com

2.120 Launch and View Remote Batch Job Results

Category: advanced_mappings
Tags: remote-jobs, key-mapping, file-operations

Quickly launch remote batch jobs and view their most recent output file with custom key mappings

Example: Example (Vim)

```
" F7: Execute current file after saving and making executable
" F12: Edit most recent file matching a pattern
map <F7> :w<Bar>:!(chmod +x %; %)<CR>
map <F12> :exec EditMostRecentFile()<CR>

function! EditMostRecentFile()
  let g:pattern = input("EditMostRecentFile. Pattern of files ?
  ↪  (".g:recent." )")
  if g:pattern != ""
    let g:recent = g:pattern
  endif
  let shell_cmd = "ls -t ".g:recent."| head -1"
  exec "e ".system(shell_cmd)
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F7>', function()
  vim.cmd('w')
  os.execute('chmod +x ' .. vim.fn.expand('%') .. ' && ' ..
  ↪  vim.fn.expand('%'))
end, { desc = 'Save and execute current file' })

local function edit_most_recent_file()
  local pattern = vim.fn.input('EditMostRecentFile. Pattern of files? ('..
  ↪  (vim.g.recent or '*') .. '): ')
  if pattern ~= '' then
    vim.g.recent = pattern
  end
  local shell_cmd = 'ls -t ' .. (vim.g.recent or '*') .. '| head -1'
  local recent_file = vim.fn.system(shell_cmd):gsub('%s+', '')
  vim.cmd('edit ' .. recent_file)
end

vim.keymap.set('n', '<F12>', edit_most_recent_file, { desc = 'Edit most
↪  recent file' })
```

Source: vim.fandom.com

2.121 Macro Playback with Function Keys

Category: advanced_mappings
Tags: macro, key-mapping, productivity

Map function keys to specific macro registers for quick and easy macro replay

Example: Example (Vim)

```
nmap <F2> @a
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F2>', '@a', { desc = 'Play macro from a register' })
```

Source: vim.fandom.com

2.122 Manual XML Formatting Mapping

Category: advanced_mappings
Tags: xml, formatting, key-mapping

Create a custom key mapping to format XML files on demand using xmllint

Example: Example (Vim)

```
map @@x !%xmllint --format --recover -^M
```

Example: Example (Neovim)

```
vim.keymap.set('n', '@@x', ':%!xmllint --format --recover -<CR>', { desc =
↪  'Format XML document' })
```

Source: vim.fandom.com

2.123 Map Ctrl-Backspace to Delete Previous Word

Category: advanced_mappings
Tags: key-mapping, insert-mode, editing

Allows using Ctrl-Backspace to delete the previous word in insert mode, similar to terminal/readline behavior

Example: Example (Vim)

```
" Map Ctrl-Backspace to delete the previous word in insert mode
imap <C-BS> <C-W>

" For terminal compatibility
noremap! <C-BS> <C-w>
noremap! <C-h> <C-w>
```

Example: Example (Neovim)

```
-- Map Ctrl-Backspace to delete previous word
vim.keymap.set('i', '<C-BS>', '<C-W>', { desc = 'Delete previous word' })

-- More readline-like behavior
vim.keymap.set('i', '<C-w>', '<C-\><C-o>dB', { desc = 'Delete previous WORD'
↪  })
vim.keymap.set('i', '<C-BS>', '<C-\><C-o>db', { desc = 'Delete previous
↪  word' })
```

Source: vim.fandom.com

2.124 Map Semicolon to Command Mode

Category: advanced_mappings
Tags: key-mapping, productivity, command-line

Quickly enter command mode by mapping semicolon to colon, reducing the need to press shift

Example: Example (Vim)

```
" Map semicolon to enter command mode
nnoremap ; :
vnoremap ; :
```

Example: Example (Neovim)

```
-- Map semicolon to enter command mode
vim.keymap.set('n', ';', ':', { desc = 'Enter command mode quickly' })
vim.keymap.set('v', ';', ':', { desc = 'Enter command mode in visual mode'
↪  })
```

Source: vim.fandom.com

2.125 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>"', 'ciw"<C-r>""<Esc>')
vim.keymap.set('n', "<leader>'", "ciw'<C-r>\"'<Esc>")
vim.keymap.set('n', '<leader>\\', '<Cmd>nohlsearch<CR>')
```

Source: Community contributed

2.126 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
local function delete_lines(count)
    vim.cmd('normal! ' .. count .. 'dd')
end

vim.keymap.set('n', '<leader>d' function()
    delete_lines(vim.v.count1)
end)

-- Or even like this
vim.keymap.set('n', '<leader>d' function()
    vim.cmd('normal! ' .. vim.v.count1 .. 'dd')
end, { silent = true })
```

Source: Community contributed

2.127 Move Function Parameters with Macros

Category: advanced_mappings
Tags: macro, editing, function-parameters

Quickly move function parameters within a function call using custom key mappings

Example: Example (Vim)

```
"Move after next comma
nmap ,mc "zdiWxf,a <Esc>"zp

"Move until next parenthesis
nmap ,mp "zdiWxf)hi, <Esc>"zpx
```

Example: Example (Neovim)

```
-- Move parameter after next comma
vim.keymap.set('n', ',mc', function()
  vim.cmd('normal! "zdiWxf,a ')
  vim.cmd('normal! "zp')
end, { desc = 'Move parameter after comma' })

-- Move parameter to end of parameters
vim.keymap.set('n', ',mp', function()
  vim.cmd('normal! "zdiWxf)hi, ')
  vim.cmd('normal! "zpx')
end, { desc = 'Move parameter to end' })
```

Source: vim.fandom.com

2.128 Move Lines Up/Down with Alt Keys

Category: advanced_mappings
Tags: key-mapping, line-manipulation, productivity

Easily move entire lines or selected blocks up or down in normal, insert, and visual modes using Alt+j/k

Example: Example (Vim)

```
nnoremap <A-j> :m .+1<CR>==
nnoremap <A-k> :m .-2<CR>==
inoremap <A-j> <Esc>:m .+1<CR>==gi
inoremap <A-k> <Esc>:m .-2<CR>==gi
vnoremap <A-j> :m '>+1<CR>gv=gv
vnoremap <A-k> :m '<-2<CR>gv=gv
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<A-j>', ':m .+1<CR>==', { noremap = true, silent = true
↪  })
vim.keymap.set('n', '<A-k>', ':m .-2<CR>==', { noremap = true, silent = true
↪  })
vim.keymap.set('i', '<A-j>', '<Esc>:m .+1<CR>==gi', { noremap = true, silent
↪  = true })
vim.keymap.set('i', '<A-k>', '<Esc>:m .-2<CR>==gi', { noremap = true, silent
↪  = true })
vim.keymap.set('v', '<A-j>', ':m ">+1<CR>gv=gv', { noremap = true, silent =
↪  true })
vim.keymap.set('v', '<A-k>', ':m "<-2<CR>gv=gv', { noremap = true, silent =
↪  true })
```

Source: vim.fandom.com

2.129 Multi-Language Comment/Uncomment Mappings

Category: advanced_mappings
Tags: comments, language-specific, text-editing

Flexible mappings to comment/uncomment lines across different programming languages using single-key shortcuts

Example: Example (Vim)

```
" Language-specific comment mappings
map ,# :s/^/#/<CR>
map ,/ :s/^/\/\//<CR>
map ,> :s/^/> /<CR>
map ,' :s/^/'/<CR>
map ,c :s/^/\/\/\|^--\|^> \|^[#"%!;]//<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent using Neovim keymap API
local function comment_line(comment_char)
  vim.api.nvim_command(string.format('s/^/%s/', comment_char))
end

vim.keymap.set('n', ',#', function() comment_line('#') end)
vim.keymap.set('n', ',/', function() comment_line('//') end)
vim.keymap.set('n', ',>', function() comment_line('> ') end)
```

Source: vim.fandom.comcommenting_out_of_source_code)

2.130 Multiple key mappings

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

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

Example: Example (Vim)

```
: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>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>fs', '<Cmd>w<CR>')
vim.keymap.set('n', '<leader>ff', '<Cmd>find<Space>')
vim.keymap.set('n', '<leader>fb', '<Cmd>buffer<Space>')
vim.keymap.set('n', '<C-s>', '<Cmd>w<Space>')
vim.keymap.set('i', '<C-s>', '<Esc><Cmd>w<CR>a')
```

Source: Community contributed

2.131 Numeric Keypad Escape Sequence Mapping

Category: advanced_mappings
Tags: key-mapping, terminal, input

Map PuTTY numeric keypad escape sequences to actual numbers in Vim

Example: Example (Vim)

```
" Map numeric keypad escape sequences
:inoremap <Esc>Oq 1
:inoremap <Esc>Or 2
:inoremap <Esc>Os 3
:inoremap <Esc>Ot 4
:inoremap <Esc>Ou 5
:inoremap <Esc>Ov 6
:inoremap <Esc>Ow 7
:inoremap <Esc>Ox 8
:inoremap <Esc>Oy 9
:inoremap <Esc>Op 0
```

Example: Example (Neovim)

```
-- Map numeric keypad escape sequences
vim.keymap.set('i', '<Esc>Oq', '1')
vim.keymap.set('i', '<Esc>Or', '2')
vim.keymap.set('i', '<Esc>Os', '3')
vim.keymap.set('i', '<Esc>Ot', '4')
vim.keymap.set('i', '<Esc>Ou', '5')
vim.keymap.set('i', '<Esc>Ov', '6')
vim.keymap.set('i', '<Esc>Ow', '7')
vim.keymap.set('i', '<Esc>Ox', '8')
vim.keymap.set('i', '<Esc>Oy', '9')
vim.keymap.set('i', '<Esc>Op', '0')
```

Source: vim.fandom.com

2.132 One-Key Header/Source File Toggle

Category: advanced_mappings
Tags: key-mapping, file-switching

Quick single-key mapping to toggle between .cpp and .h files in the same directory

Example: Example (Vim)

```
map <F4> :e %:p:s,.h$,.X123X,:s,.cpp$,.h,:s,.X123X$,.cpp,<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', function()
  local current_file = vim.fn.expand('%:p')
  local new_file = current_file:gsub('\.h$', '.X123X'):gsub('\.cpp$',
  ↪  '.h'):gsub('\.X123X$', '.cpp')
  vim.cmd('edit ' .. new_file)
end, { desc = 'Toggle between source and header' })
```

Source: vim.fandom.com

2.133 Open Online Documentation for Word Under Cursor

Category: advanced_mappings
Tags: documentation, browser-integration, key-mapping

Quickly open online documentation for the word under the cursor, supporting multiple programming languages and documentation sources

Example: Example (Vim)

```
function! OnlineDoc()
  if &ft =~ "cpp"
    let s:urlTemplate = "http://doc.trolltech.com/4.1/%.html"
  elseif &ft =~ "ruby"
    let s:urlTemplate = "http://www.ruby-doc.org/core/classes/%.html"
  elseif &ft =~ "perl"
    let s:urlTemplate = "http://perldoc.perl.org/functions/%.html"
  else
    return
  endif
  let s:browser = "firefox"
  let s:wordUnderCursor = expand("<cword>")
  let s:url = substitute(s:urlTemplate, "%", s:wordUnderCursor, "g")
  let s:cmd = "silent !" . s:browser . " " . s:url
  execute s:cmd
endfunction

map <M-d> :call OnlineDoc()<CR>
```

Example: Example (Neovim)

```
function OnlineDoc()
  local ft = vim.bo.filetype
  local urlTemplate = nil

  if ft:match("cpp") then
    urlTemplate = "http://doc.trolltech.com/4.1/%.html"
  elseif ft:match("ruby") then
    urlTemplate = "http://www.ruby-doc.org/core/classes/%.html"
  elseif ft:match("perl") then
    urlTemplate = "http://perldoc.perl.org/functions/%.html"
  else
    return
  end

  local browser = "firefox"
  local wordUnderCursor = vim.fn.expand("<cword>")
  local url = urlTemplate:gsub("%%", wordUnderCursor)
  local cmd = string.format("silent !%s %s", browser, url)

  vim.cmd(cmd)
end

vim.keymap.set('n', '<M-d>', OnlineDoc, { desc = 'Open online documentation'
↪  })
```

Source: vim.fandom.com

2.134 Open Web Links Under Cursor

Category: advanced_mappings
Tags: key-mapping, url-handling, web

Quickly open URLs or web links directly from Vim

Example: Example (Vim)

```
" Open URL under cursor
nmap gF viWgF
vmap <silent> gF y:sil! !start C:/progra~1/intern~1/iexplore.exe
↪  <C-R>=escape(@0,"#%")<CR><CR>
```

Example: Example (Neovim)

```
-- Open URL under cursor
vim.keymap.set('n', 'gF', function()
  local url = vim.fn.expand('<cfile>')
  if url ~= '' then
    vim.fn.system(string.format('start %s', url))
  end
end, { desc = 'Open URL under cursor' })

-- Visual mode version
vim.keymap.set('v', 'gF', function()
  local selected_text = vim.fn.getvisualtext()
  if selected_text ~= '' then
    vim.fn.system(string.format('start %s', selected_text))
  end
end, { desc = 'Open selected URL' })
```

Source: vim.fandom.com

2.135 Open Windows Help Files with Custom Keymapping

Category: advanced_mappings
Tags: windows, help, key-mapping

Create custom keymappings to quickly open Windows HTML help files for specific topics, using either current word or visual selection

Example: Example (Vim)

```
nnoremap <C-F1> :silent! !cmd.exe /c start keyhh.exe -#klink "<C-R><C-W>"
↪  "$ASANY9\docs\dbmaen9.chm"<CR><CR>
vnoremap <C-F1> :<C-U>let old_reg=@"<CR>gvy:silent!!cmd.exe /cstart
↪  keyhh.exe -#klink "<C-R><C-R>"" "$ASANY9\docs\dbmaen9.chm"<CR><CR>:let
↪  @"=old_reg<CR>:echo ""<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-F1>', function()
  vim.cmd.silent('!cmd.exe /c start keyhh.exe -#klink "' ..
  ↪ vim.fn.expand('<cword>') .. '" "$ASANY9\\docs\\dbmaen9.chm"')
end)

vim.keymap.set('v', '<C-F1>', function()
  local old_reg = vim.fn.getreg('"')
  vim.cmd.normal{'gvy'}
  local selected_text = vim.fn.getreg('"')
  vim.cmd.silent('!cmd.exe /c start keyhh.exe -#klink "' .. selected_text ..
  ↪ '" "$ASANY9\\docs\\dbmaen9.chm"')
  vim.fn.setreg('"', old_reg)
end)
```

Source: vim.fandom.com

2.136 Open Windows Help Files with Vim Keymaps

Category: advanced_mappings
Tags: windows, key-mapping, help

Create custom keymaps to open Windows HTML Help files with the word under cursor or selected text

Example: Example (Vim)

```
nnoremap <C-F1> :silent! !cmd.exe /c start keyhh.exe -#klink "<C-R><C-W>"
↪ "$ASANY9\docs\dbmaen9.chm"<CR><CR>
vnoremap <C-F1> :<C-U>let old_reg=@"<CR>gvy:silent!!cmd.exe /cstart
↪ keyhh.exe -#klink "<C-R><C-R>"" "$ASANY9\docs\dbmaen9.chm"<CR><CR>:let
↪ @"=old_reg<CR>:echo ""<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-F1>', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.system(string.format('cmd.exe /c start keyhh.exe -#klink "%s"
  ↪ "$ASANY9\\docs\\dbmaen9.chm"', word))
end, { silent = true })

vim.keymap.set('v', '<C-F1>', function()
  local old_reg = vim.fn.getreg('"')
  vim.cmd('normal! gvy')
  local selected = vim.fn.getreg('"')
  vim.fn.system(string.format('cmd.exe /c start keyhh.exe -#klink "%s"
  ↪ "$ASANY9\\docs\\dbmaen9.chm"', selected))
  vim.fn.setreg('"', old_reg)
end, { silent = true })
```

Source: vim.fandom.com

2.137 Operator-pending mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('o', 'in(', '<Cmd><C-u>normal! f(vi(<CR>')
vim.keymap.set('o', 'an(', '<Cmd><C-u>normal! f(va(<CR>')
vim.keymap.set('o', 'in{', '<Cmd><C-u>normal! f{va{<CR>')
```

Source: Community contributed

2.138 Postfix Abbreviations for Quick Code Snippets

Category: advanced_mappings
Tags: abbreviations, productivity, code-generation

Create dynamic code snippet expansions using postfix abbreviations, allowing quick insertion of boilerplate code by typing the array/variable name followed by an abbreviation

Example: Example (Vim)

```
:ab ff <Esc>^d$ifor(int i=0;i<<Esc>pi.length;i++){<CR><CR>}//end for loop
↪  over array <Esc>pi[i]<Esc>==k==k==ji<Tab>
```

Example: Example (Neovim)

```
-- Example of creating an abbreviation in Lua
vim.cmd.abbreviate('ff', '<Esc>^d$ifor(int
↪  i=0;i<<Esc>pi.length;i++){<CR><CR>}//end for loop over array
↪  <Esc>pi[i]<Esc>==k==k==ji<Tab>')
```

Source: vim.fandom.com

2.139 Preserve Original Semicolon Functionality

Category: advanced_mappings
Tags: key-mapping, workaround

Remap double semicolon to preserve original semicolon functionality for repeat f/t commands

Example: Example (Vim)

```
" Double semicolon to get original semicolon behavior
noremap ;; ;
```

Example: Example (Neovim)

```
-- Double semicolon to get original semicolon behavior
vim.keymap.set('n', ';;', ';', { desc = 'Repeat last f/t motion' })
```

Source: vim.fandom.com

2.140 Prevent Accidental Command Typos

Category: advanced_mappings
Tags: key-mapping, error-prevention, productivity

Correct common command-line typos and prevent unintended actions by remapping frequently mistyped commands

Example: Example (Vim)

```
" Correct common command-line typos
cabbrev Q quit
cabbrev W write

" Prevent accidentally holding Shift
cabbrev q!@ q!
cabbrev wq!@ wq!
```

Example: Example (Neovim)

```
-- Lua equivalent for command typo correction
vim.cmd('cabbrev Q quit')
vim.cmd('cabbrev W write')
vim.cmd('cabbrev q!@ q!')
vim.cmd('cabbrev wq!@ wq!')
```

Source: vim.fandom.com

2.141 Prevent Accidental Key Presses

Category: advanced_mappings
Tags: key-mapping, error-prevention, usability

Remap problematic keys to prevent unintended actions or disable them entirely

Example: Example (Vim)

```
" Prevent F1 key from interrupting in insert mode
inoremap <F1> <Nop>

" Normalize Shift+Arrow key behavior
nnoremap <S-Up> V
nnoremap <S-Down> V
```

Example: Example (Neovim)

```
-- Prevent F1 key from interrupting in insert mode
vim.keymap.set('i', '<F1>', '<Nop>')

-- Normalize Shift+Arrow key behavior
vim.keymap.set('n', '<S-Up>', 'V')
vim.keymap.set('n', '<S-Down>', 'V')
```

Source: vim.fandom.com

2.142 Prevent Accidental Text Deletion in Insert Mode

Category: advanced_mappings
Tags: insert-mode, undo, key-mapping

Add an undo breakpoint before Ctrl-U and Ctrl-W deletions to make them undoable

Example: Example (Vim)

```
inoremap <c-u> <c-g>u<c-u>
inoremap <c-w> <c-g>u<c-w>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<C-u>', '<C-g>u<C-u>', { desc = 'Undo-breakpoint before
↪  line deletion' })
vim.keymap.set('i', '<C-w>', '<C-g>u<C-w>', { desc = 'Undo-breakpoint before
↪  word deletion' })
```

Source: vim.fandom.com

2.143 Quick Abbreviation Expansion Mapping

Category: advanced_mappings
Tags: abbreviation, mapping, productivity

Provides a quick way to expand existing abbreviations with a custom keybinding

Example: Example (Vim)

```
map <C-X><C-X> diw:exe "normal i".@"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-X><C-X>', function()
  -- Alternative implementation that works better
  vim.cmd('ciw@')
  vim.cmd('normal! "_s\<C-R>"')
  vim.cmd('normal! b')
end, { desc = 'Expand existing abbreviation' })
```

Source: vim.fandom.com

2.144 Quick Braces Insertion in Insert Mode

Category: advanced_mappings
Tags: key-mapping, programming, productivity

Quickly insert opening and closing braces with automatic indentation, saving keystrokes when programming

Example: Example (Vim)

```
" Opening and closing braces
imap <C-F> {<CR>}<C-O>O
```

Example: Example (Neovim)

```
-- Quick braces insertion
vim.keymap.set('i', '<C-f>', '{<CR>}<C-o>O', { desc = 'Insert braces with
↪  auto-indent' })
```

Source: vim.fandom.com

2.145 Quick C Debugging Print Statement Macro

Category: advanced_mappings
Tags: debugging, key-mapping

Quickly insert fprintf debugging statements with file and line information for easier tracking and removal

Example: Example (Vim)

```
nmap _if ofprintf(0<C-d>stderr, "{%s} {%d} - \n", __FILE__,
↪  __LINE__);<Esc>F\i
```

Example: Example (Neovim)

```
vim.keymap.set('n', '_if', function()
  local debug_stmt = string.format('fprintf(0stderr, "{%%s} {%%d} - \n",
  ↪  __FILE__, __LINE__);
')
  vim.api.nvim_put({ debug_stmt }, 'l', true, true)
end)
```

Source: vim.fandom.com

2.146 Quick Comment/Uncomment for Web Languages

Category: advanced_mappings
Tags: key-mapping, commenting, filetype-specific

Provides language-specific comment/uncomment mappings for HTML, JS, and CSS using localleader key

Example: Example (Vim)

```
" HTML
autocmd BufNewFile,BufRead *.html nnoremap <buffer> <localleader>c I<!--
↪  <esc>A --><esc>
autocmd BufNewFile,BufRead *.html nnoremap <buffer> <localleader>uc 0/-
↪  <cr>lv0d$v3hd
" JS
autocmd FileType javascript nnoremap <buffer> <localleader>c I// <esc>
autocmd FileType javascript nnoremap <buffer> <localleader>uc 0v2ld
" CSS
autocmd BufNewFile,BufRead *.css nnoremap <buffer> <localleader>c I/* <esc>A
↪  */<esc>
autocmd BufNewFile,BufRead *.css nnoremap <buffer> <localleader>uc
↪  0v2ld$v2hd
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.html',
  callback = function()
```

```
    vim.keymap.set('n', '<localleader>c', 'I<!-- <esc>A --><esc>', {buffer =
    ↪ true})
    vim.keymap.set('n', '<localleader>uc', '0/- <cr>lv0d$v3hd', {buffer =
    ↪ true})
  end
})

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'javascript',
  callback = function()
    vim.keymap.set('n', '<localleader>c', 'I// <esc>', {buffer = true})
    vim.keymap.set('n', '<localleader>uc', '0v2ld', {buffer = true})
  end
})

vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.css',
  callback = function()
    vim.keymap.set('n', '<localleader>c', 'I/* <esc>A */<esc>', {buffer =
    ↪ true})
    vim.keymap.set('n', '<localleader>uc', '0v2ld$v2hd', {buffer = true})
  end
})
```

Source: vim.fandom.com

2.147 Quick Compile Mapping with F7

Category: advanced_mappings
Tags: compile, key-mapping, build-tools

Add a convenient key mapping to save and compile the current buffer in its directory

Example: Example (Vim)

```
function! Make()
  let curr_dir = expand('%:h')
  if curr_dir == ''
    let curr_dir = '.'
  endif
  execute 'lcd ' . curr_dir
  execute 'make %:r.o'
  execute 'lcd -'
endfunction
nnoremap <F7> :update<CR>:call Make()<CR>
```

Example: Example (Neovim)

```
function _G.compile_current_buffer()
  local curr_dir = vim.fn.expand('%:h')
  if curr_dir == '' then curr_dir = '.' end
```

```
  vim.cmd('lcd ' .. curr_dir)
  vim.cmd('make %:r.o')
  vim.cmd('lcd -')
end

vim.keymap.set('n', '<F7>', function()
  vim.cmd('update')
  _G.compile_current_buffer()
end)
```

Source: vim.fandom.com

2.148 Quick Dictionary Lookup in Vim

Category: advanced_mappings
Tags: key-mapping, external-tool, dictionary

Use F12 to quickly look up words in a dictionary file using agrep, with support for whole word and partial matches

Example: Example (Vim)

```
" Whole word lookup (exact match)
map <F12> b"*yw<Esc>:! c:/bin/agrep -wih <C-R>* "c:/dict/an-cs.txt"<CR>

" Partial pattern lookup
map <S-F12> b"*yw<Esc>:! c:/bin/agrep -ih <C-R>* "c:/dict/an-cs.txt"<CR>
```

Example: Example (Neovim)

```
-- Whole word lookup (exact match)
vim.keymap.set('n', '<F12>', function()
  local word = vim.fn.expand('<cword>')
  vim.cmd('!c:/bin/agrep -wih ' .. word .. ' "c:/dict/an-cs.txt"')
end)

-- Partial pattern lookup
vim.keymap.set('n', '<S-F12>', function()
  local word = vim.fn.expand('<cword>')
  vim.cmd('!c:/bin/agrep -ih ' .. word .. ' "c:/dict/an-cs.txt"')
end)
```

Source: vim.fandom.com

2.149 Quick Directory Change Mapping

Category: advanced_mappings
Tags: key-mapping, navigation, productivity

Create a quick mapping to change directory to the current file's location and optionally display the new path

Example: Example (Vim)

```
" Quick directory change mapping
nnoremap <leader>cd :cd %:p:h<CR>

" With path display
nnoremap <leader>cd :cd %:p:h<CR>:pwd<CR>
```

Example: Example (Neovim)

```
-- Quick directory change mapping
vim.keymap.set('n', '<leader>cd', ':cd %:p:h<CR>', { desc = 'Change to
↪  current file directory' })

-- With path display
vim.keymap.set('n', '<leader>cd', function()
  vim.cmd('cd %:p:h')
  vim.cmd('pwd')
end, { desc = 'Change to current file directory and show path' })
```

Source: vim.fandom.com

2.150 Quick Escape Key Alternatives in Insert Mode

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Multiple ways to quickly exit insert mode without reaching for the Escape key, improving typing efficiency

Example: Example (Vim)

```
" Alternative escape mappings
:imap jj <Esc>
:imap jk <Esc>
:imap kj <Esc>
:imap ;; <Esc>
```

Example: Example (Neovim)

```
-- Alternative escape mappings
vim.keymap.set('i', 'jj', '<Esc>', { noremap = true })
vim.keymap.set('i', 'jk', '<Esc>', { noremap = true })
vim.keymap.set('i', 'kj', '<Esc>', { noremap = true })
vim.keymap.set('i', ';;', '<Esc>', { noremap = true })
```

Source: vim.fandom.com

2.151 Quick Escape from Insert Mode

Category: advanced_mappings
Tags: insert-mode, key-mapping, productivity

Use alternative key combinations to exit insert mode quickly without reaching for Escape key

Example: Example (Vim)

```
" Quick escape mappings
:imap jk <Esc>
:imap kj <Esc>
:imap ;; <Esc>
```

Example: Example (Neovim)

```
-- Quick escape mappings
vim.keymap.set('i', 'jk', '<Esc>', { noremap = true })
vim.keymap.set('i', 'kj', '<Esc>', { noremap = true })
vim.keymap.set('i', ';;', '<Esc>', { noremap = true })
```

Source: vim.fandom.com

2.152 Quick File Save Mapping

Category: advanced_mappings
Tags: key-mapping, file-operations, productivity

Create a quick leader key mapping to save files only when changes have been made, avoiding unnecessary writes

Example: Example (Vim)

```
noremap <Leader>s :update<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>s', ':update<CR>', { desc = 'Save file if
↪  modified' })
```

Source: vim.fandom.com

2.153 Quick File Upload with FTPeel Integration

Category: advanced_mappings
Tags: ftp, file-upload, external-integration

Create a custom mapping to quickly upload the current file to FTPeel on macOS using AppleScript

Example: Example (Vim)

```
" FTPeel MagicMirror support
fun! MagicMirrorIt()
  let path = substitute(expand("%:p"), '/', ":", "g")
  let nice_path = substitute(path, "^:", "", "")
  execute('!osascript -e "tell application \"FTPeel\" to open \"' .
  ↪ nice_path . '\"')
endfun
map <C-S> :call MagicMirrorIt()<CR>
```

Example: Example (Neovim)

```
function MagicMirrorIt()
  local path = vim.fn.expand('%:p'):gsub('/', ':')
  local nice_path = path:gsub('^:', '')
  vim.cmd(string.format('!osascript -e "tell application \"FTPeel\" to open
  ↪ \"%s\"', nice_path))
end

vim.keymap.set('n', '<C-S>', MagicMirrorIt, { desc = 'Upload file to FTPeel'
↪ })
```

Source: vim.fandom.com

2.154 Quick Global Substitution Mapping

Category: advanced_mappings
Tags: substitution, key-mapping, productivity

Create a fast mapping to initiate a global search and replace with confirmation

Example: Example (Vim)

```
:map <F4> :%s///gc<Left><Left><Left>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', ':%s///gc<Left><Left><Left>', { desc = 'Global
↪ substitution with confirmation' })
```

Source: vim.fandom.com

2.155 Quick HTML Anchor Creation from Two Lines

Category: advanced_mappings
Tags: key-mapping, html, text-transformation

Convert two lines (URL and title) into a single HTML anchor tag using a simple F7 mapping

Example: Example (Vim)

```
" Convert two lines (URL then TITLE) to one line: <a href=URL>TITLE</a>
map <F7> <Esc>I<a href="<Esc>A"><Esc>gJA</a><Esc>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F7>', function()
  -- Get the current line and next line
  local url_line = vim.fn.getline('.')
  local title_line = vim.fn.getline(vim.fn.line('.') + 1)

  -- Create HTML anchor tag
  local link = string.format('<a href="%s">%s</a>',
  ↪  url_line:match('^%s*(.-)%s*$'), title_line:match('^%s*(.-)%s*$'))

  -- Replace lines with anchor tag
  vim.fn.setline('.', link)
  vim.fn.deleteline(vim.fn.line('.') + 1)
end, { desc = 'Convert two lines to HTML anchor tag' })
```

Source: vim.fandom.com

2.156 Quick HTML/URL Preview in Browser

Category: advanced_mappings
Tags: browser-preview, file-operations, web-development

Easily open the current HTML file or URL in a browser directly from Vim/Neovim using flexible key mappings

Example: Example (Vim)

```
" Open current file in default browser
nnoremap <F5> :update<Bar>silent !xdg-open %:p &<CR>

" Open URL under cursor in browser
nnoremap <F8> :silent !xdg-open <cfile> &<CR>
```

Example: Example (Neovim)

```
-- Open current file in default browser
vim.keymap.set('n', '<F5>', function()
  vim.cmd.update()
  vim.fn.system('xdg-open ' .. vim.fn.expand('%:p') .. ' &')
end, { desc = 'Open current file in browser' })

-- Open URL under cursor in browser
vim.keymap.set('n', '<F8>', function()
  vim.fn.system('xdg-open ' .. vim.fn.expand('<cfile>') .. ' &')
end, { desc = 'Open URL under cursor in browser' })
```

Source: vim.fandom.com

2.157 Quick Insert Character at End of Line

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Quickly add a character (like semicolon or quote) at the end of the current line without moving the cursor

Example: Example (Vim)

```
imap <silent><F2> <Esc>v`^me<Esc>gi<C-o>:call Ender()<CR>

function Ender()
  let endchar = nr2char(getchar())
  execute "normal \<End>a".endchar
  normal `e
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<F2>', function()
  -- Record current position
  local current_line = vim.fn.line('.')
  local current_col = vim.fn.col('.')

  -- Move to end of line and prompt for character
  local endchar = vim.fn.nr2char(vim.fn.getchar())
  vim.cmd('normal! A' .. endchar)

  -- Return to original position
  vim.fn.cursor(current_line, current_col)
end, { silent = true })
```

Source: vim.fandom.com

2.158 Quick Insert Mode Commands via Double Letters

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Map double uppercase letters to perform quick normal mode actions without leaving insert mode, improving editing efficiency

Example: Example (Vim)

```
" Quick movements
" Go to first non-blank text
inoremap II <Esc>I
" Go to end of line
inoremap AA <Esc>A
" Open new line above
inoremap OO <Esc>O

" Line modifications
" Change right of cursor
inoremap CC <Esc>C
" Change whole line
inoremap SS <Esc>S
" Delete current line
inoremap DD <Esc>dd
" Undo
inoremap UU <Esc>u
```

Example: Example (Neovim)

```
-- Quick movements and line modifications
local function setup_insert_mode_mappings()
  vim.keymap.set('i', 'II', '<Esc>I', { desc = 'Go to first non-blank text'
  ↪  })
  vim.keymap.set('i', 'AA', '<Esc>A', { desc = 'Go to end of line' })
  vim.keymap.set('i', 'OO', '<Esc>O', { desc = 'Open new line above' })
  vim.keymap.set('i', 'CC', '<Esc>C', { desc = 'Change right of cursor' })
  vim.keymap.set('i', 'SS', '<Esc>S', { desc = 'Change whole line' })
  vim.keymap.set('i', 'DD', '<Esc>dd', { desc = 'Delete current line' })
  vim.keymap.set('i', 'UU', '<Esc>u', { desc = 'Undo' })
end

setup_insert_mode_mappings()
```

Source: vim.fandom.com

2.159 Quick Internet Search for Word Under Cursor

Category: advanced_mappings
Tags: key-mapping, web-search, productivity

Quickly search the internet for the word under the cursor using a custom key mapping

Example: Example (Vim)

```
" Search Google for word under cursor
nmap gF vviWgF
vmap <silent> gF y:sil! !start C:/progra~1/intern~1/iexplore.exe -nohome
↪  http://www.google.com/search?hl=en&q=<C-R>0<CR>
```

Example: Example (Neovim)

```
-- Search Google for word under cursor
vim.keymap.set('n', 'gF', function()
  local word = vim.fn.expand('<cword>')
  local url = string.format('http://www.google.com/search?hl=en&q=%s', word)
  vim.fn.system(string.format('start %s', url))
end, { desc = 'Search Google for word under cursor' })

-- Visual mode version
vim.keymap.set('v', 'gF', function()
  local selected_text = vim.fn.getvisualtext()
  local url = string.format('http://www.google.com/search?hl=en&q=%s',
  ↪  selected_text)
  vim.fn.system(string.format('start %s', url))
end, { desc = 'Search Google for selected text' })
```

Source: vim.fandom.com

2.160 Quick Java Compilation and Execution Mappings

Category: advanced_mappings
Tags: java, compilation, key-mapping

Set up function key mappings to quickly compile and run Java files, with support for both current and alternate files

Example: Example (Vim)

```
" Compile current file with F9, run with F10
" Compile alternate file with F11, run with F12
map <F9> :set makeprg=javac\ %<CR>:make<CR>
map <F10> :!echo %|awk -F. '{print $1}'|xargs java<CR>
map <F11> :set makeprg=javac\ #<CR>:make<CR>
map <F12> :!echo #|awk -F. '{print $1}'|xargs java<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F9>', function()
  vim.o.makeprg = 'javac %'
  vim.cmd('make')
```

```
end, { desc = 'Compile current Java file' })

vim.keymap.set('n', '<F10>', function()
  local filename = vim.fn.expand('%:r')
  vim.cmd('!java ' .. filename)
end, { desc = 'Run current Java file' })

-- Similar mappings can be added for F11 and F12 for alternate file
```

Source: vim.fandom.com

2.161 Quick MSDN Documentation Lookup

Category: advanced_mappings
Tags: key-mapping, web-search, documentation

Quickly open MSDN documentation for a word under cursor using Google's 'I'm Feeling Lucky' search

Example: Example (Vim)

```
nmap <F1> :silent ! start iexplore "http://www.google.com/search?hl=en&btnI
↪  =I%27m+Feeling+Lucky&q=site%3Amsdn.microsoft.com%20<cWORD>"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F1>', function()
  local word = vim.fn.expand('<cWORD>')
  local url = string.format("http://www.google.com/search?hl=en&btnI=I'm+Fe
  ↪  eling+Lucky&q=site:msdn.microsoft.com %s",
  ↪  word)
  vim.fn.system(string.format("start iexplore '%s'", url))
end, { desc = 'Quick MSDN Documentation Lookup' })
```

Source: vim.fandom.com

2.162 Quick Macro Playback with Q

Category: advanced_mappings
Tags: macro, key-mapping, productivity

Remap Q to quickly play back the macro recorded in the q register, making macro replay more convenient

Example: Example (Vim)

```
noremap Q @q
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'Q', '@q', { desc = 'Play macro from q register' })
```

Source: vim.fandom.com

2.163 Quick Markdown Preview with External Tools

Category: advanced_mappings
Tags: markdown, preview, external-tools

Create a key mapping to quickly convert Markdown to HTML and open in a browser for instant preview

Example: Example (Vim)

```
map ^P :w!<CR>:w!/home/user/tmp/vim-markdown.md<CR>:!pandoc -s -f markdown
↪  -t html -o /home/user/tmp/vim-markdown.html
↪  /home/user/tmp/vim-markdown.md<CR>:!dillo
↪  /home/user/tmp/vim-markdown.html > /dev/null 2> /dev/null&<CR><CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-p>', function()
  -- Save current file
  vim.cmd('write')

  -- Define temporary paths
  local tmp_md = '/home/user/tmp/vim-markdown.md'
  local tmp_html = '/home/user/tmp/vim-markdown.html'

  -- Write current buffer to temp markdown file
  vim.fn.writefile(vim.api.nvim_buf_get_lines(0, 0, -1, false), tmp_md)

  -- Convert markdown to HTML using pandoc
  os.execute(string.format('pandoc -s -f markdown -t html -o %s %s',
  ↪  tmp_html, tmp_md))

  -- Open in browser (can replace 'dillo' with preferred browser)
  os.execute(string.format('dillo %s > /dev/null 2> /dev/null &', tmp_html))
end, { desc = 'Quick Markdown Preview' })
```

Source: vim.fandom.com

2.164 Quick Number Conversion Under Cursor

Category: advanced_mappings
Tags: mapping, number-conversion, utility

Add a custom mapping to quickly convert and display the number under the cursor be-

tween decimal and hexadecimal formats

Example: Example (Vim)

```
nnoremap gn :call DecAndHex(expand("<cWORD>"))<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'gn', function()
  local word = vim.fn.expand('<cWORD>')
  local dec = tonumber(word)
  local hex = tonumber(word, 16)

  if dec then
    print(string.format('%d -> 0x%X', dec, dec))
  elseif hex then
    print(string.format('0x%X -> %d', hex, hex))
  else
    print('Not a valid number')
  end
end, { desc = 'Convert number under cursor' })
```

Source: vim.fandom.com

2.165 Quick Omni and User Completion

Category: advanced_mappings
Tags: completion, key-mapping, productivity

Add mappings to quickly open omni and user completion menus with smart behavior

Example: Example (Vim)

```
" Open omni completion menu
inoremap <expr> <C-Space> (pumvisible() ? (col('.') > 1 ? '<Esc>i<Right>' :
↪  '<Esc>i') : '') .
            \ '<C-x><C-o><C-r>=pumvisible() ? "\<lt>C-n>\<lt>C-p>\<lt>Down>"
            ↪  : ""<CR>'
" Open user completion menu
inoremap <expr> <S-Space> (pumvisible() ? (col('.') > 1 ? '<Esc>i<Right>' :
↪  '<Esc>i') : '') .
            \ '<C-x><C-u><C-r>=pumvisible() ? "\<lt>C-n>\<lt>C-p>\<lt>Down>"
            ↪  : ""<CR>'
```

Example: Example (Neovim)

```
-- Open omni completion menu
vim.keymap.set('i', '<C-Space>', function()
  local col = vim.fn.col('.')
  local prefix = (vim.fn.pumvisible() == 1) and
```

```
    ((col > 1) and '<Esc>i<Right>' or '<Esc>i') or ''

  return prefix .. '<C-x><C-o><C-r>=pumvisible() ?
  ↪  "\<lt>C-n>\<lt>C-p>\<lt>Down>" : ""<CR>'
  end, { expr = true })

-- Open user completion menu
vim.keymap.set('i', '<S-Space>', function()
  local col = vim.fn.col('.')
  local prefix = (vim.fn.pumvisible() == 1) and
    ((col > 1) and '<Esc>i<Right>' or '<Esc>i') or ''

  return prefix .. '<C-x><C-u><C-r>=pumvisible() ?
  ↪  "\<lt>C-n>\<lt>C-p>\<lt>Down>" : ""<CR>'
  end, { expr = true })
```

Source: vim.fandom.com

2.166 Quick One-Key Session Command Mapping

Category: advanced_mappings
Tags: key-mapping, command-execution, productivity

Dynamically map the last executed command to a function key for quick repeated execution

Example: Example (Vim)

```
function MapLastCommandToKeys(keysToMapTo)
  exe "unmap ".a:keysToMapTo
  exe "map ".a:keysToMapTo." :".histget("cmd")."<CR>"
endfunction

function PrepareMap(keysToMapTo)
  exe "map ".a:keysToMapTo." :call
  ↪  MapLastCommandToKeys('".a:keysToMapTo."')<CR>"
endfunction

" Map F1-F12 for quick command repetition
let i=1
while i<13
  call PrepareMap('<F'.i.'>')
  let i=i+1
endwhile
```

Example: Example (Neovim)

```
local function map_last_command_to_keys(keys_to_map_to)
  vim.cmd('unmap ' .. keys_to_map_to)
  vim.cmd('map ' .. keys_to_map_to .. ' :' .. vim.fn.histget('cmd') ..
  ↪  '<CR>')
```

```
end

local function prepare_map(keys_to_map_to)
  vim.cmd('map ' .. keys_to_map_to .. ' :lua map_last_command_to_keys("' ..
  ↪  keys_to_map_to .. '")<CR>')
end

-- Map F1-F12 for quick command repetition
for i = 1, 12 do
  prepare_map('<F' .. i .. '>')
end
```

Source: vim.fandom.com

2.167 Quick PHP Function Inline Help

Category: advanced_mappings
Tags: mapping, help, php

Create a custom mapping to fetch PHP function documentation in a new window using lynx

Example: Example (Vim)

```
map <C-F1> "vyiw:new<CR>:execute "r!lynx -dump
↪  http://localhost/phpman/function.".@v.".html"<CR>:1<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-F1>', function()
  -- Yank the word under cursor
  local word = vim.fn.expand('<cword>')

  -- Open new window and fetch PHP function documentation
  vim.cmd('new')
  vim.fn.system('lynx -dump http://localhost/phpman/function.' .. word ..
  ↪  '.html')
  vim.cmd('1')
end, { desc = 'Get PHP function inline help' })
```

Source: vim.fandom.com

2.168 Quick Python Documentation in Vim

Category: advanced_mappings
Tags: python, documentation, buffer-management

Easily access Python documentation for words/modules under cursor using pydoc, with

flexible display options

Example: Example (Vim)

```
nnoremap <buffer> K :<C-u>let save_isk = &iskeyword |
    \ set iskeyword+=. |
    \ execute "!pydoc " . expand("<cword>") |
    \ let &iskeyword = save_isk<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'K', function()
  local save_iskeyword = vim.opt.iskeyword:get()
  vim.opt.iskeyword:append('.')
  vim.fn.system('pydoc ' .. vim.fn.expand('<cword>'))
  vim.opt.iskeyword = save_iskeyword
end, { buffer = true, desc = 'Show Python documentation' })
```

Source: vim.fandom.com

2.169 Quick Single Character Insertion

Category: advanced_mappings
Tags: key-mapping, editing, efficiency

Quickly insert a single character without switching to insert mode, with support for character repetition

Example: Example (Vim)

```
function! RepeatChar(char, count)
  return repeat(a:char, a:count)
endfunction
nnoremap s :<C-U>exec "normal i".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
nnoremap S :<C-U>exec "normal a".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
```

Example: Example (Neovim)

```
function RepeatChar(char, count)
  return string.rep(char, count)
end

vim.keymap.set('n', 's', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.cmd('normal! i' .. string.rep(char, count))
end)
```

```
vim.keymap.set('n', 'S', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.cmd('normal! a' .. string.rep(char, count))
end)
```

Source: vim.fandom.com

2.170 Quick Single Word Insertion Mapping

Category: advanced_mappings
Tags: key-mapping, insert-mode, workflow

Create a custom mapping to quickly insert a single word and automatically exit insert mode

Example: Example (Vim)

```
nmap <buffer> <silent> ,w :exec ":imap \<space\> \<space\>\<esc\>,BB"<CR>i
nmap <buffer> <silent> ,BB :exec ":iunmap \<space\>"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', ',w', function()
  -- Temporarily map space to exit insert mode
  vim.keymap.set('i', ' ', function()
    vim.cmd('stopinsert')
    vim.keymap.del('i', ' ')
  end, { buffer = true })
  vim.cmd('startinsert')
end, { desc = 'Quick word insertion' })
```

Source: vim.fandom.com

2.171 Quick Spellcheck Mappings

Category: advanced_mappings
Tags: spelling, autocorrect, navigation

Create convenient mappings for spell checking and navigating spelling errors in Vim/Neovim

Example: Example (Vim)

```
map <F5> :setlocal spell! spelllang=en_us<CR>
" Jump to next misspelled word
map <F6> ]s
" Jump to previous misspelled word
map <F7> [s
```

```
" Suggest corrections
map <F8> z=
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', function()
  vim.opt.spell = not vim.opt.spell
  vim.opt.spelllang = 'en_us'
end, { desc = 'Toggle Spell Check' })

vim.keymap.set('n', '<F6>', ']s', { desc = 'Next Spelling Error' })
vim.keymap.set('n', '<F7>', '[s', { desc = 'Previous Spelling Error' })
vim.keymap.set('n', '<F8>', 'z=', { desc = 'Spelling Suggestions' })
```

Source: vim.fandom.com

2.172 Quick Syntax Highlighting Reset Mapping

Category: advanced_mappings
Tags: mapping, syntax, productivity

Add a quick keybinding to reset syntax highlighting when it breaks

Example: Example (Vim)

```
noremap <F12> <Esc>:syntax sync fromstart<CR>
inoremap <F12> <C-o>:syntax sync fromstart<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F12>', ':syntax sync fromstart<CR>', { noremap = true
↪  })
vim.keymap.set('i', '<F12>', '<C-o>:syntax sync fromstart<CR>', { noremap =
↪  true })
```

Source: vim.fandom.com

2.173 Quick System Clipboard Keymaps

Category: advanced_mappings
Tags: clipboard, key-mapping, productivity

Add convenient keymaps to interact with system clipboard in different modes

Example: Example (Vim)

```
" Paste in insert mode
:inoremap <C-v> <ESC>"+pa
" Copy in visual mode
:vnoremap <C-c> "+y
" Cut in visual mode
:vnoremap <C-d> "+d
```

Example: Example (Neovim)

```
-- Paste in insert mode
vim.keymap.set('i', '<C-v>', '<ESC>"+pa', { noremap = true })

-- Copy in visual mode
vim.keymap.set('v', '<C-c>', '"+y', { noremap = true })

-- Cut in visual mode
vim.keymap.set('v', '<C-d>', '"+d', { noremap = true })
```

Source: vim.fandom.com

2.174 Quick Text Replacement Workflow

Category: advanced_mappings
Tags: text-objects, productivity, replace

Efficient method to replace multiple words or text objects using change and register recall

Example: Example (Vim)

```
" Replace words consistently with repeatability
" yiw (yank word)
" ciw<C-r>0 (change inner word, paste from register 0)
" . (repeat last change)
```

Example: Example (Neovim)

```
-- Note: This workflow is inherently Vim-like and works similarly in Neovim
-- 1. yank word with yiw
-- 2. Move to target word
-- 3. Use ciw<C-r>0 to replace
-- 4. Use . to repeat
```

Source: vim.fandom.com

2.175 Quick Toggle for Vim Options

Category: advanced_mappings
Tags: key-mapping, configuration, options

Provides a single-key interface to toggle various Vim options quickly, reducing the number of mappings and simplifying option management

Example: Example (Vim)

```
" Map leader+o to open toggle options menu
map <Leader>o :ToggleOptions<CR>

function! ToggleOptionsMenu()
  let l:char = s:Char_menu( "{w}rap, {W}rapscan, {l}ist, {b}inary,
  ↪  {m}odifiable, {d}iff, {v}irtualedit, uhe{x}" )
  if l:char ==# 'w'
    set wrap!
    set wrap?
  elseif l:char ==# 'd'
    if &diff
      set nodiff foldcolumn=0
    else
      diffthis
    endif
    set diff?
  endif
endfunction
```

Example: Example (Neovim)

```
-- Lua equivalent of toggle options menu
local function toggle_options()
  local options = {
    w = function() vim.wo.wrap = not vim.wo.wrap end,
    d = function()
      if vim.wo.diff then
        vim.wo.diff = false
        vim.wo.foldcolumn = '0'
      else
        vim.cmd('diffthis')
      end
    end
  }

  local char = vim.fn.getchar()
  local option_func = options[vim.fn.nr2char(char)]
  if option_func then option_func() end
end

-- Map leader+o to toggle options
vim.keymap.set('n', '<Leader>o', toggle_options, { desc = 'Toggle Vim
↪  Options' })
```

Source: vim.fandom.com

2.176 Quick Uuencode/Decode Text in Vim

Category: advanced_mappings
Tags: encoding, text-processing, key-mapping

Provides convenient key mappings to uuencode and uudecode text in normal and visual modes, useful for email attachments and text encoding

Example: Example (Vim)

```
nnoremap <silent> <Leader>ue :%!uuencode -m /dev/stdout<CR>
nnoremap <silent> <Leader>ud :%!uudecode -o /dev/stdout<CR>
vnoremap <silent> <Leader>ue !uuencode -m /dev/stdout<CR>
vnoremap <silent> <Leader>ud !uudecode -o /dev/stdout<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>ue', ':%!uuencode -m /dev/stdout<CR>', { silent
↪  = true })
vim.keymap.set('n', '<Leader>ud', ':%!uudecode -o /dev/stdout<CR>', { silent
↪  = true })
vim.keymap.set('v', '<Leader>ue', '!uuencode -m /dev/stdout<CR>', { silent =
↪  true })
vim.keymap.set('v', '<Leader>ud', '!uudecode -o /dev/stdout<CR>', { silent =
↪  true })
```

Source: vim.fandom.com

2.177 Quick Web Search from Vim

Category: advanced_mappings
Tags: search, web-integration, key-mapping

Easily search the web for the word under cursor or selected text using different search engines

Example: Example (Vim)

```
" Search Google for current word
nmap gF vviWgF
vmap <silent> gF y:sil! !start C:/progra~1/intern~1/iexplore.exe -nohome
↪  http://www.google.com/search?hl=en&q=<C-R>0<CR>
```

Example: Example (Neovim)

```
-- Search Google for current word
vim.keymap.set('n', 'gF', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.system('xdg-open "https://www.google.com/search?q=' .. word .. '"')
end, { desc = 'Search Google for word under cursor' })
```

Source: vim.fandom.com

2.178 Quick Word 'Stamping' Across Document

Category: advanced_mappings
Tags: editing, registers, key-mapping

Replace words quickly with last yanked text using a custom mapping

Example: Example (Vim)

```
" Replace current word with last yanked text
nnoremap S diw"0P
```

Example: Example (Neovim)

```
-- Lua equivalent for word stamping
vim.keymap.set('n', 'S', function()
  vim.cmd('normal! diw"0P')
end, { noremap = true })
```

Source: vim.fandom.com

2.179 Quick Word Replacement Mapping

Category: advanced_mappings
Tags: text-objects, editing, replacement

Create a quick mapping to replace the current word with the last yanked text, making repetitive text replacement easy.

Example: Example (Vim)

```
nnoremap S diw"0P
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'S', 'diw"0P', { desc = 'Replace word with last yanked
↪  text' })
```

Source: vim.fandom.com

2.180 Quick Word Replacement Without Typing

Category: advanced_mappings
Tags: word-replace, registers, key-mapping

Efficiently replace words using registers without manual retyping, allowing fast word substitution across multiple occurrences

Example: Example (Vim)

```
" Copy word to register b
noremap gy lb"bye

" Replace word with register b (with space)
noremap go lb"bPldwi <Esc>hbye

" Replace word with register b (without space)
noremap gp lb"bPldwhbyw
```

Example: Example (Neovim)

```
-- Copy word to register b
vim.keymap.set('n', 'gy', 'lb"bye', { desc = 'Yank word to register b' })

-- Replace word with register b (with space)
vim.keymap.set('n', 'go', 'lb"bPldwi <Esc>hbye', { desc = 'Replace word with
↪  register b and add space' })

-- Replace word with register b (without space)
vim.keymap.set('n', 'gp', 'lb"bPldwhbyw', { desc = 'Replace word with
↪  register b' })
```

Source: vim.fandom.com

2.181 Quick Word Replacement with Last Yanked Text

Category: advanced_mappings
Tags: text-objects, registers, editing

Use 'S' to quickly replace the current word with the last yanked text, allowing rapid word replacement across a file

Example: Example (Vim)

```
nnoremap S diw"0P
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'S', function()
  vim.cmd('normal! diw"0P')
end)
```

Source: vim.fandom.com

2.182 Quick Word Search Across Files

Category: advanced_mappings
Tags: mapping, search, quickfix

Create a mapping to search for word under cursor in all files and open quickfix window

Example: Example (Vim)

```
map <F4> :execute "vimgrep /" . expand("<cword>") . "/j **" <Bar> cw<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', function()
  vim.cmd('execute "vimgrep /' .. vim.fn.expand('<cword>') .. '/j **" | cw')
end, { noremap = true, silent = true })
```

Source: vim.fandom.com

2.183 Quick Word Search Across Project

Category: advanced_mappings
Tags: mapping, search, productivity

Create a mapping to search for the word under cursor across all files in the current project

Example: Example (Vim)

```
map <F4> :execute "vimgrep /" . expand("<cword>") . "/j **" <Bar> cw<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', function()
  vim.cmd('vimgrep /' .. vim.fn.expand('<cword>') .. '/j **')
  vim.cmd('cw')
end)
```

Source: vim.fandom.com

2.184 Quick Word/Line Change with Auto-Exit

Category: advanced_mappings
Tags: key-mapping, editing-workflow, custom-mapping

Custom mappings to quickly change text and exit insert mode with Space or Enter

Example: Example (Vim)

```
nmap pw :inoremap <Space> <Space><Esc>:iunmap <Space><CR><CR> cw
nmap p$ :inoremap <CR> <CR><Esc>:iunmap <CR><CR><CR> c$
```

Example: Example (Neovim)

```
-- Lua equivalent with modern Neovim mapping
vim.keymap.set('n', 'pw', function()
  vim.api.nvim_buf_set_keymap(0, 'i', '<Space>', '<Space><Esc>', {noremap =
  ↪  true, silent = true})
  vim.cmd('normal! cw')
end, { desc = 'Change word and exit with Space' })

vim.keymap.set('n', 'p$', function()
  vim.api.nvim_buf_set_keymap(0, 'i', '<CR>', '<CR><Esc>', {noremap = true,
  ↪  silent = true})
  vim.cmd('normal! c$')
end, { desc = 'Change to line end and exit with Enter' })
```

Source: vim.fandom.com

2.185 Quick Word/Selection Replacement Mappings

Category: advanced_mappings
Tags: clipboard, key-mapping, visual-mode

Convenient key mappings for copying and replacing words or selections using clipboard

Example: Example (Vim)

```
" Copy current word/selection to clipboard
nnoremap <F4> "+yiw
vnoremap <F4> "+y

" Replace with clipboard contents
nnoremap <F5> viw"+p
vnoremap <F5> "+p
```

Example: Example (Neovim)

```
-- Copy current word/selection to clipboard
vim.keymap.set('n', '<F4>', '"+yiw', { desc = 'Copy word to clipboard' })
vim.keymap.set('v', '<F4>', '"+y', { desc = 'Copy selection to clipboard' })

-- Replace with clipboard contents
vim.keymap.set('n', '<F5>', 'viw"+p', { desc = 'Replace word with clipboard'
↪  })
vim.keymap.set('v', '<F5>', '"+p', { desc = 'Replace selection with
↪  clipboard' })
```

Source: vim.fandom.com

2.186 Quick XML Element Creation Mapping

Category: advanced_mappings
Tags: xml, text-expansion, insert-mode

Rapidly create XML elements from any word in insert mode by typing the word followed by three commas

Example: Example (Vim)

```
imap ,,, <Esc>diwi<<Esc>pa><CR></<Esc>pa><Esc>kA
```

Example: Example (Neovim)

```
vim.keymap.set('i', ',,,', function()
  -- Get current word
  local word = vim.fn.expand('<cword>')
  -- Delete word and create XML element
  vim.cmd('normal! diw')
  vim.api.nvim_put({string.format('<%s>', word)}, '', true, true)
  vim.cmd('normal! o')
  vim.api.nvim_put({string.format('</%s>', word)}, '', true, true)
  vim.cmd('normal! k')
end, { buffer = true })
```

Source: vim.fandom.com

2.187 Quickfix-Powered Code Review Workflow

Category: advanced_mappings
Tags: code-review, workflow, quickfix, collaboration

Create a systematic code review process using Vim's quickfix list to track and navigate review comments across files

Example: Example (Vim)

```
function SavePosition()
  let g:file_name=expand("%:t")
  let g:line_number=line(".")
  let g:reviewer_initials="KG" " Your initials
endfunction

function InsertComment()
  execute "normal i". g:file_name . ":" . g:line_number . ": " .
  ↪  g:reviewer_initials . " - "
  startinsert
endfunction
nmap ,sp :call SavePosition()<CR>
nmap ,ic :call InsertComment()<CR>
```

Example: Example (Neovim)

```
function _G.save_position()
  vim.g.file_name = vim.fn.expand('%:t')
  vim.g.line_number = vim.fn.line('.')
  vim.g.reviewer_initials = 'KG' -- Your initials
end

function _G.insert_comment()
  local cmd = string.format('normal i%s:%d: %s - ',
    vim.g.file_name, vim.g.line_number, vim.g.reviewer_initials)
  vim.cmd(cmd)
  vim.cmd('startinsert')
end

vim.keymap.set('n', ',sp', _G.save_position, {})
vim.keymap.set('n', ',ic', _G.insert_comment, {})
```

Source: vim.fandom.com

2.188 Quickly Comment/Uncomment Multiple Lines

Category: advanced_mappings
Tags: commenting, code-editing, key-mapping

Dynamically comment or uncomment lines based on file type using a single function and mapping

Example: Example (Vim)

```
function! Comment()
  let ft = &filetype
  if ft == 'python' || ft == 'sh' || ft == 'ruby'
    silent s/^/\#/
  elseif ft == 'javascript' || ft == 'c' || ft == 'cpp'
    silent s:^:\/\/:g
```

```
  elseif ft == 'vim'
    silent s:^:":g
  endif
endfunction

function! Uncomment()
  let ft = &filetype
  if ft == 'python' || ft == 'sh' || ft == 'ruby'
    silent s/^\#//
  elseif ft == 'javascript' || ft == 'c' || ft == 'cpp'
    silent s:^\/\/::g
  elseif ft == 'vim'
    silent s:^"::g
  endif
endfunction

map <C-a> :call Comment()<CR>
map <C-b> :call Uncomment()<CR>
```

Example: Example (Neovim)

```
function _G.comment_lines()
  local ft = vim.bo.filetype
  local comment_map = {
    python = '#',
    sh = '#',
    ruby = '#',
    javascript = '//',
    c = '//',
    cpp = '//'
  }
  local comment_char = comment_map[ft] or '#'

  -- Use visual line mode to get selected lines
  local start_line = vim.fn.line("'<")
  local end_line = vim.fn.line("'>")

  for line = start_line, end_line do
    vim.fn.setline(line, comment_char .. vim.fn.getline(line))
  end
end

function _G.uncomment_lines()
  local ft = vim.bo.filetype
  local comment_map = {
    python = '#',
    sh = '#',
    ruby = '#',
    javascript = '//',
    c = '//',
    cpp = '//'
  }
  local comment_char = comment_map[ft] or '#'
```

```
  local start_line = vim.fn.line("'<")
  local end_line = vim.fn.line("'>")

  for line = start_line, end_line do
    local current_line = vim.fn.getline(line)
    if current_line:match('^' .. vim.pesc(comment_char)) then
      vim.fn.setline(line, current_line:gsub('^' .. vim.pesc(comment_char),
      ↪  ''))
    end
  end
end

-- Set key mappings
vim.keymap.set('v', '<C-a>', ':lua _G.comment_lines()<CR>', { noremap =
↪  true, silent = true })
vim.keymap.set('v', '<C-b>', ':lua _G.uncomment_lines()<CR>', { noremap =
↪  true, silent = true })
```

Source: vim.fandom.com

2.189 Quickly Insert Non-Commented Lines

Category: advanced_mappings
Tags: mapping, comments, formatting

Override automatic comment insertion when using 'o' or 'O' to open new lines

Example: Example (Vim)

```
nnoremap go o<Esc>S
nnoremap gO O<Esc>S
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'go', 'o<Esc>S', { desc = 'Insert line below without
↪  comments' })
vim.keymap.set('n', 'gO', 'O<Esc>S', { desc = 'Insert line above without
↪  comments' })
```

Source: vim.fandom.com

2.190 Quickly Insert Single Character

Category: advanced_mappings
Tags: key-mapping, editing, productivity

Provides an efficient way to insert a single character without switching to insert mode repeatedly

Example: Example (Vim)

```
nnoremap s :exec "normal i".nr2char(getchar())."\e"<CR>
nnoremap S :exec "normal a".nr2char(getchar())."\e"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 's', function()
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.api.nvim_put({char}, '', true, true)
end, { desc = 'Insert single char before cursor' })

vim.keymap.set('n', 'S', function()
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.api.nvim_put({char}, '', false, true)
end, { desc = 'Insert single char after cursor' })
```

Source: vim.fandom.com

2.191 Quickly Introduce and Extract Variables

Category: advanced_mappings
Tags: refactoring, text-manipulation, key-mapping

A quick mapping to extract a complex expression into a named variable, useful for code refactoring and readability

Example: Example (Vim)

```
" map \v to put x = y on the line above cursor
map <Leader>v 0wh:put .<CR>a = <Esc>pa<CR><Esc>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>v', function()
  -- Go to start of line
  vim.cmd('0wh:put .')
  vim.cmd('a = ')
  vim.cmd('normal! p')
  vim.cmd('normal! o')
end, { desc = 'Extract expression to variable' })
```

Source: vim.fandom.com

2.192 Quickly Launch and Edit Remote Batch Job Results

Category: advanced_mappings
Tags: job-management, remote-execution, file-editing

Provides a workflow for launching remote jobs and quickly editing their output files using custom key mappings

Example: Example (Vim)

```
" F7: Execute current file after saving and making executable
map <F7> :w<Bar>:!(chmod +x %; %)<CR>

" F12: Edit most recent file matching a pattern
map <F12> :exec EditMostRecentFile()<CR>

function! EditMostRecentFile()
  let g:pattern = input("EditMostRecentFile. Pattern of files ?
  ↪  (".g:recent." )")
  if g:pattern != ""
    let g:recent = g:pattern
  endif
  let shell_cmd = "ls -t ".g:recent."| head -1"
  exec "e ".system(shell_cmd)
endfunction
```

Example: Example (Neovim)

```
-- F7: Execute current file after saving and making executable
vim.keymap.set('n', '<F7>', function()
  vim.cmd('w')
  vim.fn.system('chmod +x ' .. vim.fn.expand('%'))
  vim.fn.system(vim.fn.expand('%'))
end, { desc = 'Save and execute current file' })

-- F12: Edit most recent file matching a pattern
vim.keymap.set('n', '<F12>', function()
  local pattern = vim.fn.input('EditMostRecentFile. Pattern of files? ')
  if pattern ~= '' then
    local shell_cmd = 'ls -t ' .. (pattern or '*') .. ' | head -1'
    local recent_file = vim.fn.trim(vim.fn.system(shell_cmd))
    vim.cmd('edit ' .. recent_file)
  end
end, { desc = 'Open most recent file' })
```

Source: vim.fandom.com

2.193 Quickly Replace Words Without Typing

Category: advanced_mappings
Tags: key-mapping, word-replacement, registers

Efficiently replace words by yanking a word to a register and then pasting it over another word, avoiding manual retyping

Example: Example (Vim)

```
" Copy word to register b
noremap gy lb"bye

" Replace word with register b (with space)
noremap go lb"bPldwi <Esc>hbye

" Replace word with register b (without space)
noremap gp lb"bPldwhbyw
```

Example: Example (Neovim)

```
-- Copy word to register b
vim.keymap.set('n', 'gy', 'lb"bye', { desc = 'Yank word to register b' })

-- Replace word with register b (with space)
vim.keymap.set('n', 'go', 'lb"bPldwi <Esc>hbye', { desc = 'Replace word with
↪  register b, add space' })

-- Replace word with register b (without space)
vim.keymap.set('n', 'gp', 'lb"bPldwhbyw', { desc = 'Replace word with
↪  register b, no space' })
```

Source: vim.fandom.com

2.194 Recursive abbreviations

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.keymap.set('ca', 'W', 'w')
vim.keymap.set('ca', 'Wq', 'wq')
vim.keymap.set('ca', 'Q', 'q')
```

Source: Community contributed

2.195 Remap Caps Lock to Escape on macOS

Category: advanced_mappings
Tags: keyboard-layout, productivity, key-mapping

Remap Caps Lock key to Escape in system settings, which is especially useful for Vim/Neovim users who frequently use the Escape key

Example: Example (Neovim)

```
-- No direct Neovim code, this is a system-level configuration
-- Go to System Settings > Keyboard > Modifier Keys
-- Select 'Escape' for Caps Lock key
```

Source: vim.fandom.com

2.196 Remap CapsLock to Escape or Ctrl

Category: advanced_mappings
Tags: key-mapping, productivity, workflow

Remapping CapsLock to Escape or Ctrl can significantly improve typing efficiency and reduce hand strain in Vim/Neovim

Example: Example (Vim)

```
" No native Vim mapping, uses OS-level remapping
```

Example: Example (Neovim)

```
-- Recommended to use AutoHotkey, SharpKeys, or OS-level key remapping
-- In Neovim, you can alternatively create a custom mapping
vim.keymap.set('n', '<CapsLock>', '<Esc>', { noremap = true, silent = true
↪  })
```

Source: vim.fandom.com

2.197 Remap CapsLock to Escape or Ctrl in Neovim

Category: advanced_mappings
Tags: key-mapping, productivity, input-optimization

Efficiently remap CapsLock key to Escape or Ctrl to improve keyboard ergonomics and reduce hand movement in Neovim

Example: Example (Neovim)

```
-- Remap CapsLock to Escape using AutoHotkey (Windows)
-- In Neovim init.lua
vim.keymap.set('i', '<C-c>', '<Esc>', { desc = 'Alternative escape method'
↪  })

-- For system-wide remapping, use AutoHotkey or Keyboard Manager
-- Example Lua mapping to simulate CapsLock behavior
local function caps_escape()
  if vim.fn.mode() == 'i' then
    return '<Esc>'
  else
    return '<C-\>'
  end
end

vim.keymap.set({'i', 'n'}, '<CapsLock>', caps_escape, { expr = true })
```

Source: vim.fandom.com

2.198 Remap Extra Keys on Non-US Keyboards

Category: advanced_mappings
Tags: key-mapping, internationalization, keyboard-layout

Customize keyboard mappings for non-US keyboard layouts to improve navigation and typing efficiency

Example: Example (Vim)

```
" German keyboard example
map ü <C-]>
map ö [
map ä ]
map Ö {
map Ä }
map ß /
```

Example: Example (Neovim)

```
-- German keyboard example
vim.keymap.set('n', 'ü', '<C-]>', { desc = 'Custom mapping for ]' })
vim.keymap.set('n', 'ö', '[', { desc = 'Custom mapping for [' })
vim.keymap.set('n', 'ä', ']', { desc = 'Custom mapping for ]' })
vim.keymap.set('n', 'Ö', '{', { desc = 'Custom mapping for {' })
vim.keymap.set('n', 'Ä', '}', { desc = 'Custom mapping for }' })
vim.keymap.set('n', 'ß', '/', { desc = 'Custom mapping for /' })
```

Source: vim.fandom.com

2.199 Remap Help Navigation for AZERTY Keyboards

Category: advanced_mappings
Tags: keyboard-layout, help-navigation, key-mapping

Provides a mapping solution for AZERTY keyboard users to navigate Vim help, which typically uses Ctrl+] to follow links

Example: Example (Vim)

```
" Map F9 to navigate help links
nmap <F9> <C-]>
map! <F9> <C-]>
```

Example: Example (Neovim)

```
-- Map F9 to navigate help links in all modes
vim.keymap.set({'n', 'i', 'c'}, '<F9>', '<C-]>', { desc = 'Navigate help
↪  links' })
```

Source: vim.fandom.com

2.200 Repeat Last Command at Start of Change

Category: advanced_mappings
Tags: key-mapping, editing, productivity

Modify the dot (.) command to return cursor to start of previous change, making repetitive editing across multiple lines easier

Example: Example (Vim)

```
nmap . .`[
```

Example: Example (Neovim)

```
vim.keymap.set('n', '.', function()
  vim.cmd('normal! .`[')
end, { desc = 'Repeat last command and return to change start' })
```

Source: vim.fandom.com

2.201 Repeat Last Edit Without Moving Cursor

Category: advanced_mappings
Tags: editing, cursor-position, dot-command

Create a mapping that repeats the last edit while maintaining the original cursor position, mimicking Emacs behavior

Example: Example (Vim)

```
" Restore cursor position after dot repeat
noremap . mz.`z
```

Example: Example (Neovim)

```
-- Restore cursor position after dot repeat
vim.keymap.set('n', '.', function()
  local cursor_pos = vim.fn.getpos('.')
  vim.cmd('normal! .')
  vim.fn.setpos('.', cursor_pos)
end, { desc = 'Repeat last edit without moving cursor' })
```

Source: vim.fandom.com

2.202 Repeatable Single Character Insertion

Category: advanced_mappings
Tags: key-mapping, repeat, editing

Enhanced version that supports repeating character insertion with count

Example: Example (Vim)

```
function! RepeatChar(char, count)
  return repeat(a:char, a:count)
endfunction
nnoremap s :<C-U>exec "normal i".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
nnoremap S :<C-U>exec "normal a".RepeatChar(nr2char(getchar()),
↪  v:count1)<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 's', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.api.nvim_put({string.rep(char, count)}, '', true, true)
end, { desc = 'Insert repeatable single char before cursor' })

vim.keymap.set('n', 'S', function()
  local count = vim.v.count1
  local char = vim.fn.nr2char(vim.fn.getchar())
  vim.api.nvim_put({string.rep(char, count)}, '', false, true)
end, { desc = 'Insert repeatable single char after cursor' })
```

Source: vim.fandom.com

2.203 Restore Cursor Position After Repeat Command

Category: advanced_mappings
Tags: cursor-movement, key-mapping, editing

Modify the dot (.) repeat command to maintain cursor position after repeating the last edit, similar to Emacs behavior

Example: Example (Vim)

```
"make . not move the cursor
noremap . mz.`z
```

Example: Example (Neovim)

```
vim.keymap.set('n', '.', function()
  local mark = vim.fn.getpos('.')
  vim.cmd('normal! .')
  vim.fn.setpos('.', mark)
end, { desc = 'Repeat last edit without moving cursor' })
```

Source: vim.fandom.com

2.204 Run Macro on Matching Lines

Category: advanced_mappings
Tags: macros, text-processing

Execute a recorded macro on all lines matching a specific pattern

Example: Example (Vim)

```
" Run macro 'q' on lines containing 'pattern'
:g/pattern/normal @q
```

Example: Example (Neovim)

```
-- Run macro 'q' on lines containing 'pattern'
vim.cmd('g/pattern/normal @q')
```

Source: vim.fandom.com

2.205 Run Python Scripts from Vim

Category: advanced_mappings
Tags: key-mapping, python, external-commands

Quick mapping to execute the current Python file

Example: Example (Vim)

```
map <F5> <Esc>:!python %<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', function() vim.cmd('!python %') end, { desc =
↪  'Run current Python file' })
```

Source: vim.fandom.com

2.206 Run Windows Program with K Keymap

Category: advanced_mappings
Tags: key-mapping, command-execution, windows

Quickly open a Windows program for the current word using the K key without spawning a command prompt

Example: Example (Vim)

```
map K yiw:exec "silent !".&kp." ".@0<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'K', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.system({vim.o.keywordprg, word})
end, { desc = 'Run program on current word' })
```

Source: vim.fandom.com

2.207 Run and Preview Interpreter Output in Split Window

Category: advanced_mappings
Tags: scripting, interpreter, output-preview, development

Allows running selected code snippets in an interpreter and displaying output in a preview window, useful for quick script testing and debugging

Example: Example (Vim)

```
function! Ruby_eval_vsplit() range
  let src = tempname()
  let dst = tempname()
  execute ': ' . a:firstline . ',' . a:lastline . 'w ' . src
  execute ':silent ! ruby ' . src . ' > ' . dst . ' 2>&1'
  execute ':pclose!'
  execute ':redraw!'
  execute ':vsplit'
  execute 'normal \<C-W>l'
  execute ':e! ' . dst
  execute ':set pvw'
  execute 'normal \<C-W>h'
endfunction

vmap <silent> <F7> :call Ruby_eval_vsplit()<CR>
nmap <silent> <F7> mzggVG<F7>`z
imap <silent> <F7> <Esc><F7>a
```

Example: Example (Neovim)

```
function _G.ruby_eval_vsplit(first_line, last_line)
  local src = vim.fn.tempname()
  local dst = vim.fn.tempname()

  -- Write selected lines to temp file
  vim.cmd(string.format(':%d,%dw %s', first_line, last_line, src))

  -- Run Ruby and capture output
  os.execute(string.format('ruby %s > %s 2>&1', src, dst))

  -- Open preview window with results
  vim.cmd('pclose!')
  vim.cmd('redraw!')
  vim.cmd('vsplit')
  vim.cmd('wincmd l')
  vim.cmd('edit! ' .. dst)
  vim.cmd('set pvw')
  vim.cmd('wincmd h')
end

-- Mappings
vim.keymap.set('v', '<F7>', function()
  local first_line = vim.fn.line("'<")
  local last_line = vim.fn.line("'>")
  _G.ruby_eval_vsplit(first_line, last_line)
end, { silent = true })

vim.keymap.set('n', '<F7>', function()
  vim.cmd('normal! mzggVG')
  _G.ruby_eval_vsplit(1, vim.fn.line('$'))
  vim.cmd('normal! `z')
end, { silent = true })
```

Source: vim.fandom.com

2.208 SQL Command History Management in Vim

Category: advanced_mappings
Tags: sql, command-history, buffer-management

Advanced function to manage SQL command history, automatically saving and tracking SQL statements in a separate history file

Example: Example (Vim)

```
fu! VimSQL()
  setf sql
  nnoremap <C-K> :<C-U>
  exe "let linenum=".v:count<CR>:1,$-1d<CR><C-W>j:exe linenum."y"<CR><C-W>kP
  let linenum=line("$")
  1,$-1w! >> ~/.sqlplus.history
  e ~/.sqlplus.history
  execute ":$-".(linenum-1)",$m0"
  %!uniq
  if line("$")>100
    101,$d
  endif
  b#
  set splitbelow
  sp ~/.sqlplus.history
  au! BufEnter afiedt.buf
endf
au BufEnter afiedt.buf call VimSQL()
```

Example: Example (Neovim)

```
local function vim_sql()
  vim.bo.filetype = 'sql'
  vim.keymap.set('n', '<C-K>', function()
    local linenum = vim.v.count
    -- Similar logic would be implemented using vim.api and vim.fn functions
    -- Exact translation requires more complex Lua mapping
  end)
end

vim.api.nvim_create_autocmd('BufEnter', {
  pattern = 'afiedt.buf',
  callback = vim_sql
})
```

Source: vim.fandom.com

2.209 Safely List and Manage Vim Key Mappings

Category: advanced_mappings
Tags: key-mapping, configuration, debugging

Use built-in Vim commands to list, inspect, and manage key mappings across different modes, helping you understand and debug your configuration.

Example: Example (Vim)

```
" List mappings by mode
:nmap   " Normal mode mappings
:imap   " Insert mode mappings
:vmap   " Visual mode mappings

" Remove a specific mapping
:unmap <F2>
```

Example: Example (Neovim)

```
-- List mappings in Neovim
-- Use :nmap, :imap etc. directly or use lua API
vim.cmd('nmap')  -- List normal mode mappings

-- To remove a mapping
vim.keymap.del('n', '<F2>')
```

Source: vim.fandom.com

2.210 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
local function compile_and_run()
    vim.cmd('!ggc % -o %:r && ./%:r')
end

vim.keymap.set('n', '<F5>', compile_and_run, { silent = true })
```

Source: Community contributed

2.211 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>w', '<Cmd>w<CR>', { silent = true })
vim.keymap.set('i', 'jk', '<Esc>')
```

Source: Community contributed

2.212 Simple RCS Version Control Mappings

Category: advanced_mappings
Tags: version-control, rcs, key-mapping

Quick mappings for basic RCS version control operations directly from Vim, allowing check-in and check-out of files with function keys

Example: Example (Vim)

```
map <F1> :write %<CR>:!ci -l %<CR>:edit!<CR>
map <F2> :!co -l %<CR>:edit!<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F1>', function()
  vim.cmd.write()
  os.execute('ci -l ' .. vim.fn.expand('%'))
  vim.cmd.edit({ bang = true })
end, { desc = 'RCS Check-in' })

vim.keymap.set('n', '<F2>', function()
  os.execute('co -l ' .. vim.fn.expand('%'))
  vim.cmd.edit({ bang = true })
end, { desc = 'RCS Check-out' })
```

Source: vim.fandom.com

2.213 Simulate Shift-Arrow Text Selection in Terminals

Category: advanced_mappings
Tags: terminal, key-mapping, visual-mode

Add mappings to enable text selection using Ctrl-arrows in terminals where Shift-arrows don't work correctly

Example: Example (Vim)

```
" Simulate shift-arrows with control-arrows
inoremap <Esc>[A <C-O>vk
vnoremap <Esc>[A k
inoremap <Esc>[B <C-O>vj
vnoremap <Esc>[B j
inoremap <Esc>[C <C-O>vl
vnoremap <Esc>[C l
inoremap <Esc>[D <C-O>vh
vnoremap <Esc>[D h
```

Example: Example (Neovim)

```
-- Simulate shift-arrows with control-arrows
vim.keymap.set({'i', 'v'}, '<Esc>[A', function()
  if vim.fn.mode() == 'i' then
    vim.cmd('normal! vk')
  else
    vim.cmd('normal! k')
  end
end, { desc = 'Select up in terminal' })

-- Similar mappings for other directions can be added accordingly
```

Source: vim.fandom.com

2.214 Smart Bracket and Parenthesis Insertion for Perl

Category: advanced_mappings
Tags: auto-completion, insert-mode, language-specific

Intelligent bracket insertion for Perl, automatically handling different contexts like block creation and hash syntax

Example: Example (Vim)

```
function! InsertBrackets()
  let fileType = &ft
  if fileType == 'perl'
    let col = col('.') - 1
```

```
    if !col || getline('.')[col - 1] !~ '\k' && getline('.')[col - 1] !~
    ↪  '\$' && getline('.')[col - 1] !~ '@' && getline('.')[col - 1] !~ '%'
    ↪  && getline('.')[col - 1] !~ '#'
      return "{\<CR>\<BS>}\<Esc>ko"
    else
      return "{}\<Esc>i\<c-o>:echo \<CR>"
    endif
  else
    return "{\<CR>\<BS>}\<Esc>ko"
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.insert_brackets()
  local filetype = vim.bo.filetype
  if filetype == 'perl' then
    local col = vim.fn.col('.') - 1
    local line = vim.fn.getline('.')
    local prev_char = line:sub(col, col)

    if col == 0 or not prev_char:match('[%w$@%_%#]') then
      return '{\n}\27ko'
    else
      return '{}\27i'
    end
  else
    return '{\n}\27ko'
  end
end

vim.keymap.set('i', '{', '<C-R>=v:lua.insert_brackets()<CR>', { expr = true
↪  })
```

Source: vim.fandom.com

2.215 Smart Tab Insertion with Context

Category: advanced_mappings
Tags: insert-mode, key-mapping, indentation

Dynamically insert tabs or spaces based on cursor position and file context

Example: Example (Vim)

```
" Insert spaces at line start, real tabs elsewhere
inoremap <Silent> <Tab> <C-R>=(col('.') > (matchend(getline('.'), '^\s*') +
↪  1))?'<C-V><C-V><Tab>':'<Tab>'<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<Tab>', function()
  local col = vim.fn.col('.')
  local line = vim.fn.getline('.')
  local non_white_idx = line:find('%S')

  if non_white_idx and col > non_white_idx then
    return vim.api.nvim_replace_termcodes('<Tab>', true, false, true)
  else
    return vim.api.nvim_replace_termcodes('<C-V><Tab>', true, false, true)
  end
, { expr = true, silent = true })
```

Source: vim.fandom.com

2.216 Smart Table Editing with Tab Navigation

Category: advanced_mappings
Tags: navigation, text-editing, tables

Dynamically map Tab and Shift-Tab to navigate between table fields in both insert and normal modes, with automatic padding and alignment

Example: Example (Vim)

```
" In table.vim script
map <Tab> :call NextField()<CR>
map <S-Tab> :call PreviousField()<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent (conceptual)
function _G.next_table_field()
  -- Implement field navigation logic
end

vim.keymap.set({'n', 'i'}, '<Tab>', _G.next_table_field)
vim.keymap.set({'n', 'i'}, '<S-Tab>', _G.previous_table_field)
```

Source: vim.fandom.com

2.217 Smart Table Navigation with Custom Field Movement

Category: advanced_mappings
Tags: navigation, table-editing, custom-function

Provides a custom function to intelligently move between table columns, automatically padding lines and handling variable field lengths

Example: Example (Vim)

```
func! NextField(fieldsep,minlensep,padstr,offset)
  let curposn = col(".")
  let linenum = line(".")
  let prevline = getline(linenum-1)
  let curline = getline(linenum)
  let nextposn = matchend(prevline,a:fieldsep,curposn-a:minlensep)+1
  let padding = ""
  if nextposn > strlen(prevline) || linenum == 1 || nextposn == 0
    echo "last field or no fields on line above"
    return
  endif
  echo ""
  if nextposn > strlen(curline)
    if &modifiable == 0
      return
    endif
    let i = strlen(curline)
    while i < nextposn - 1
      let i = i + 1
      let padding = padding . a:padstr
    endwhile
    call setline(linenum,substitute(curline,"$",padding,""))
  endif
  call cursor(linenum,nextposn+a:offset)
  return
endfunc
```

Example: Example (Neovim)

```
function _G.next_field(fieldsep, minlensep, padstr, offset)
  local curposn = vim.fn.col('.')
  local linenum = vim.fn.line('.')
  local prevline = vim.fn.getline(linenum - 1)
  local curline = vim.fn.getline(linenum)

  local nextposn = vim.fn.matchend(prevline, fieldsep, curposn - minlensep)
  ↪  + 1

  if nextposn > #prevline or linenum == 1 or nextposn == 0 then
    print('last field or no fields on line above')
    return
  end

  if nextposn > #curline then
    if vim.o.modifiable == false then
      return
    end

    local padding = string.rep(padstr, nextposn - #curline)
    vim.fn.setline(linenum, curline .. padding)
  end

  vim.fn.cursor(linenum, nextposn + offset)
```

```
end

-- Example mapping
vim.keymap.set({'n', 'i'}, '<S-Tab>', function()
  _G.next_field(' {2,}', 2, ' ', 0)
end)
```

Source: vim.fandom.com

2.218 Smart Toggle Line Comment

Category: advanced_mappings
Tags: comments, toggle, text-editing

A single function to toggle comments on the current line, handling both single-line and multi-line comment styles

Example: Example (Vim)

```
function! Komment()
  if getline('.') =~ '\/\*'
    let hls=@/
    s/^\/\*//
    s/*\/$//
    let @/=hls
  else
    let hls=@/
    s/^/\/*/
    s/$/*\//
    let @/=hls
  endif
endfunction
map k :call Komment()<CR>
```

Example: Example (Neovim)

```
local function toggle_comment()
  local line = vim.api.nvim_get_current_line()
  local is_commented = line:match('^/%*') and line:match('%*/$')

  if is_commented then
    -- Uncomment
    line = line:gsub('^/%*', ''):gsub('%*/$', '')
  else
    -- Comment
    line = '/*' .. line .. '*/'
  end

  vim.api.nvim_set_current_line(line)
end
```

```
vim.keymap.set('n', 'k', toggle_comment)
```

Source: vim.fandom.comcommenting_out_of_source_code)

2.219 Swap ; and : for Faster Command Entry

Category: advanced_mappings
Tags: key-mapping, productivity, command-line

Swap semicolon and colon to reduce finger strain when entering commands, with an additional mapping to quickly edit command history

Example: Example (Vim)

```
" Swap ; and : keys
silent! nunmap ;
silent! nunmap :
nnoremap <unique> ; :
nnoremap <unique> : ;

" Quick command-line editing
cnoremap <expr> ; (getcmdpos() == 1 && getcmdtype() =~ '\v^:') ? '<C-F>A' :
↪  ';'
```

Example: Example (Neovim)

```
-- Swap ; and : keys
vim.keymap.set('n', ';', ':', { unique = true })
vim.keymap.set('n', ':', ';', { unique = true })

-- Quick command-line editing
vim.keymap.set('c', ';', function()
  if vim.fn.getcmdpos() == 1 and vim.fn.getcmdtype():match('^:') then
    return '<C-F>A'
  else
    return ';'
  end
, { expr = true })
```

Source: vim.fandom.com

2.220 Terminal mode mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('t', '<Esc>', '<C-\\><C-n>')
vim.keymap.set('t', '<C-w>h', '<C-\\><C-n><C-w>h')
vim.keymap.set('t', '<C-w>j', '<C-\\><C-n><C-w>j')
vim.keymap.set('t', '<C-w>k', '<C-\\><C-n><C-w>k')
vim.keymap.set('t', '<C-w>l', '<C-\\><C-n><C-w>l')
```

Source: Community contributed

2.221 Toggle Clojure Comments Easily

Category: advanced_mappings
Tags: commenting, language-specific, key-mapping

A quick function to toggle comments for Clojure files, allowing easy commenting and uncommenting of lines

Example: Example (Vim)

```
map <Leader>. :call ClojureCommentUncomment()<CR>
function! ClojureCommentUncomment()
  let search_saved = @/
  if getline('.') =~ '^;'
    s/^;//  " remove ';' at beginning of line
  else
    s/^/;/  " insert ';' at beginning of line
  endif
  let @/ = search_saved
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>.', function()
  local line = vim.api.nvim_get_current_line()
  if line:match('^;') then
    -- Uncomment: remove leading semicolon
    vim.api.nvim_set_current_line(line:gsub('^;', ''))
  else
    -- Comment: add semicolon at the beginning
    vim.api.nvim_set_current_line('; ' .. line)
  end
```

```
end, { desc = 'Toggle Clojure comment' })
```

Source: vim.fandom.com

2.222 Toggle Less-like Scrolling Mode

Category: advanced_mappings
Tags: scrolling, key-mapping, mode-toggle

Adds a toggle to switch between normal editing and pager-like scrolling, allowing easy navigation without moving the cursor

Example: Example (Vim)

```
function! LessMode()
  if g:lessmode == 0
    let g:lessmode = 1
    noremap <script> d <C-D>
    noremap <script> j <C-E>
    noremap <script> u <C-U>
    noremap <script> k <C-Y>
  else
    let g:lessmode = 0
    unmap d
    unmap j
    unmap u
    unmap k
  endif
endfunction

let g:lessmode = 0
nnoremap <F5> :call LessMode()<CR>
inoremap <F5> <Esc>:call LessMode()<CR>
```

Example: Example (Neovim)

```
local less_mode = false

local function toggle_less_mode()
  less_mode = not less_mode
  if less_mode then
    vim.keymap.set('n', 'd', '<C-D>')
    vim.keymap.set('n', 'j', '<C-E>')
    vim.keymap.set('n', 'u', '<C-U>')
    vim.keymap.set('n', 'k', '<C-Y>')
    print('Less mode on')
  else
    vim.keymap.unset('n', 'd')
    vim.keymap.unset('n', 'j')
    vim.keymap.unset('n', 'u')
    vim.keymap.unset('n', 'k')
```

```
    print('Less mode off')
  end
end

vim.keymap.set('n', '<F5>', toggle_less_mode)
vim.keymap.set('i', '<F5>', '<Esc>' .. toggle_less_mode)
```

Source: vim.fandom.com

2.223 Toggle Line Length Highlighting

Category: advanced_mappings
Tags: key-mapping, highlighting, configuration

Create a toggle mapping to highlight lines exceeding textwidth or a default of 80 columns

Example: Example (Vim)

```
nnoremap <silent> <Leader>l
      \ :if exists('w:long_line_match') <Bar>
      \   silent! call matchdelete(w:long_line_match) <Bar>
      \   unlet w:long_line_match <Bar>
      \ elseif &textwidth > 0 <Bar>
      \   let w:long_line_match = matchadd('ErrorMsg', '\%>'.&tw.'v.\+', -1)
      ↪  <Bar>
      \ else <Bar>
      \   let w:long_line_match = matchadd('ErrorMsg', '\%>80v.\+', -1)
      ↪  <Bar>
      \ endif<CR>
```

Example: Example (Neovim)

```
local function toggle_line_length_highlight()
  local match_exists = vim.w.long_line_match ~= nil

  if match_exists then
    vim.fn.matchdelete(vim.w.long_line_match)
    vim.w.long_line_match = nil
  else
    local textwidth = vim.o.textwidth
    local match_col = textwidth > 0 and textwidth or 80
    vim.w.long_line_match = vim.fn.matchadd('ErrorMsg', '\%>' .. match_col
    ↪  .. 'v.\+', -1)
  end
end

vim.keymap.set('n', '<Leader>l', toggle_line_length_highlight, { desc =
↪  'Toggle line length highlight' })
```

Source: vim.fandom.com

2.224 Toggle Long Line Highlighting

Category: advanced_mappings
Tags: mapping, highlighting, text-width

Create a toggle mapping to highlight lines exceeding text width, with fallback to 80 characters if no textwidth is set

Example: Example (Vim)

```
nnoremap <silent> <Leader>l
      \ :if exists('w:long_line_match') <Bar>
      \   silent! call matchdelete(w:long_line_match) <Bar>
      \   unlet w:long_line_match <Bar>
      \ elseif &textwidth > 0 <Bar>
      \   let w:long_line_match = matchadd('ErrorMsg', '\%>'.&tw.'v.\+', -1)
       ↪  <Bar>
      \ else <Bar>
      \   let w:long_line_match = matchadd('ErrorMsg', '\%>80v.\+', -1)
       ↪  <Bar>
      \ endif<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>l', function()
  local match_id = vim.w.long_line_match

  if match_id then
    vim.fn.matchdelete(match_id)
    vim.w.long_line_match = nil
  else
    local tw = vim.o.textwidth
    if tw > 0 then
      vim.w.long_line_match = vim.fn.matchadd('ErrorMsg', '\%>' .. tw ..
       ↪  'v.\+', -1)
    else
      vim.w.long_line_match = vim.fn.matchadd('ErrorMsg', '\%>80v.\+', -1)
    end
  end
end, { desc = 'Toggle long line highlighting' })
```

Source: vim.fandom.com

2.225 Toggle Option Flags Dynamically

Category: advanced_mappings
Tags: key-mapping, configuration, options

Create a flexible function to toggle Vim option flags, allowing easy runtime configuration of settings

Example: Example (Vim)

```
function ToggleFlag(option,flag)
  exec ('let lopt = &' . a:option)
  if lopt =~ (".*" . a:flag . ".*")
    exec ('set ' . a:option . '-=' . a:flag)
  else
    exec ('set ' . a:option . '+=' . a:flag)
  endif
endfunction

" Example mappings
map <silent> <F8> :call ToggleFlag("guioptions","m")<CR>
map <silent> <F9> :call ToggleFlag("guioptions","T")<CR>
```

Example: Example (Neovim)

```
function _G.toggle_flag(option, flag)
  local current_opts = vim.api.nvim_get_option(option)
  if current_opts:find(flag) then
    vim.api.nvim_set_option(option, current_opts:gsub(flag, ''))
  else
    vim.api.nvim_set_option(option, current_opts .. flag)
  end
end

-- Example mappings
vim.keymap.set('n', '<F8>', function() _G.toggle_flag('guioptions', 'm')
 ↪  end)
vim.keymap.set('n', '<F9>', function() _G.toggle_flag('guioptions', 'T')
 ↪  end)
```

Source: vim.fandom.com

2.226 Toggle Whitespace Ignore in Diff

Category: advanced_mappings
Tags: diff, toggle, whitespace

Add a keymapping to toggle whitespace ignore in diff mode, providing flexible comparison options

Example: Example (Vim)

```
if &diff
    map gs :call IwhiteToggle()<CR>
    function! IwhiteToggle()
        if &diffopt =~ 'iwhite'
            set diffopt-=iwhite
        else
            set diffopt+=iwhite
        endif
```

```
    endfunction
endif
```

Example: Example (Neovim)

```
if vim.o.diff then
    vim.keymap.set('n', 'gs', function()
        if vim.o.diffopt:find('iwhite') then
            vim.o.diffopt = vim.o.diffopt:gsub(',iwhite', '')
        else
            vim.o.diffopt = vim.o.diffopt .. ',iwhite'
        end
    end, { desc = 'Toggle whitespace ignore in diff' })
end
```

Source: vim.fandom.com

2.227 Toggle Word Completion with F12 Key

Category: advanced_mappings
Tags: key-mapping, autocomplete, mode-switching

Create a flexible mapping to toggle word completion and paste mode, providing a quick way to switch between editing modes

Example: Example (Vim)

```
" Toggle word completion and paste mode
fun! SetComplete()
  call DoWordComplete()
  set nopaste
  nunmap <F12>
  iunmap <F12>
  nmap <F12> :call UnsetComplete()<CR>
  imap <F12> <Esc>:call UnsetComplete()<CR>a
  echo
endfun

fun! UnsetComplete()
  call EndWordComplete()
  set paste
  nunmap <F12>
  iunmap <F12>
  nmap <F12> :call SetComplete()<CR>
  imap <F12> <Esc>:call SetComplete()<CR>a
  echo
endfun

nmap <F12> :call UnsetComplete()<CR>
imap <F12> <Esc>:call UnsetComplete()<CR>a
```

Example: Example (Neovim)

```
local function set_complete()
  vim.fn.DoWordComplete()
  vim.o.paste = false
  vim.keymap.del('n', '<F12>')
  vim.keymap.del('i', '<F12>')
  vim.keymap.set('n', '<F12>', unset_complete)
  vim.keymap.set('i', '<F12>', '<Esc>:lua unset_complete()<CR>a')
end

local function unset_complete()
  vim.fn.EndWordComplete()
  vim.o.paste = true
  vim.keymap.del('n', '<F12>')
  vim.keymap.del('i', '<F12>')
  vim.keymap.set('n', '<F12>', set_complete)
  vim.keymap.set('i', '<F12>', '<Esc>:lua set_complete()<CR>a')
end

-- Initial mapping
vim.keymap.set('n', '<F12>', unset_complete)
vim.keymap.set('i', '<F12>', '<Esc>:lua unset_complete()<CR>a')
```

Source: vim.fandom.com

2.228 TortoiseSVN Commands from Vim

Category: advanced_mappings
Tags: version-control, external-commands, subversion

Create custom mappings to invoke TortoiseSVN commands directly from Vim, allowing quick access to diff, log, blame, and revision graph without leaving the editor

Example: Example (Vim)

```
" Diff current file with SVN version
map <silent> ,tdiff :w<CR>:silent
 ↪  !"C:\Progra~1\TortoiseSVN\bin\TortoiseProc.exe" /command:diff /path:"%"
 ↪  /notempfile /closeonend

" Show SVN log for current file
map <silent> ,tlog :w<CR>:silent
 ↪  !"C:\Progra~1\TortoiseSVN\bin\TortoiseProc.exe" /command:log /path:"%"
 ↪  /notempfile /closeonend

" Show blame for current file
function! TortoiseBlame()
  let filename = expand("%")
  let linenum = line(".")
  silent execute('!C:\Progra~1\TortoiseSVN\bin\TortoiseProc.exe
   ↪  /command:blame /path:"' . filename . '" /line:' . linenum . '
   ↪  /notempfile /closeonend')
```

```
endfunction

map <silent> ,tblame :call TortoiseBlame()<CR>
```

Example: Example (Neovim)

```
-- Diff current file with SVN version
vim.keymap.set('n', '<leader>td', function()
  vim.cmd.write()
  vim.fn.system('"C:\\Progra~1\\TortoiseSVN\\bin\\TortoiseProc.exe"
  ↪ /command:diff /path:"' .. vim.fn.expand('%') .. '" /notempfile
  ↪ /closeonend')
end, { silent = true })

-- Show SVN log for current file
vim.keymap.set('n', '<leader>tl', function()
  vim.cmd.write()
  vim.fn.system('"C:\\Progra~1\\TortoiseSVN\\bin\\TortoiseProc.exe"
  ↪ /command:log /path:"' .. vim.fn.expand('%') .. '" /notempfile
  ↪ /closeonend')
end, { silent = true })

-- Show blame for current file at current line
vim.keymap.set('n', '<leader>tb', function()
  local filename = vim.fn.expand('%')
  local linenum = vim.fn.line('.')
  vim.cmd.write()
  vim.fn.system(string.format('"C:\\Progra~1\\TortoiseSVN\\bin\\TortoisePro
  ↪ c.exe" /command:blame /path:"%s" /line:%d /notempfile /closeonend',
  ↪ filename, linenum))
end, { silent = true })
```

Source: vim.fandom.com

2.229 Troubleshoot Backspace Key Mappings

Category: advanced_mappings
Tags: key-mapping, troubleshooting, insert-mode

Check for problematic backspace key mappings and diagnose insert mode mapping issues

Example: Example (Vim)

```
" Check backspace mappings
:verbose imap <BS>
:verbose imap ^H
```

Example: Example (Neovim)

```
-- Check backspace mappings
vim.cmd('verbose imap <BS>')
vim.cmd('verbose imap ^H')
```

Source: vim.fandom.com

2.230 Turkish Keyboard Layout Optimization

Category: advanced_mappings
Tags: key-mapping, productivity, keyboard-layout

Optimize keyboard layout by remapping less accessible keys to more convenient actions

Example: Example (Vim)

```
" Turkish keyboard layout optimization
nmap ş <cr>
let mapleader = "ğ"
```

Example: Example (Neovim)

```
-- Turkish keyboard layout optimization
vim.keymap.set('n', 'ş', '<CR>', { desc = 'Remap Enter key' })
vim.g.mapleader = 'ğ'
```

Source: vim.fandom.com

2.231 Use <Leader> for Plugin-Friendly Key Mappings

Category: advanced_mappings
Tags: key-mapping, configuration, plugins

Use <Leader> to create consistent and customizable key mappings across plugins, allowing users to easily configure their preferred leader key

Example: Example (Vim)

```
let mapleader = '_'
nnoremap <Leader>f :call <SID>JumpToFile()<CR>
```

Example: Example (Neovim)

```
vim.g.mapleader = '_'
vim.keymap.set('n', '<Leader>f', function() -- implement JumpToFile function
↪  end, { desc = 'Jump to file' })
```

Source: vim.fandom.com)

2.232 Use Ctrl-O for Single Normal Mode Command in Insert Mode

Category: advanced_mappings
Tags: key-mapping, insert-mode, productivity

Ctrl-O allows executing a single normal mode command while in insert mode without fully exiting, preventing cursor movement issues

Example: Example (Vim)

```
:imap <F5> <C-o>:set number!<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<F5>', '<C-o>:set number!<CR>', { desc = 'Toggle line
↪  numbers in insert mode' })
```

Source: vim.fandom.com

2.233 Use Ctrl-[as Escape Alternative

Category: advanced_mappings
Tags: key-mapping, insert-mode, keyboard-shortcut

Equivalent to Escape key for quickly exiting insert mode, especially useful on different keyboard layouts

Example: Example (Vim)

```
" Use Ctrl-[ as Escape alternative
" (Already built-in, no additional mapping needed)
```

Example: Example (Neovim)

```
-- Ctrl-[ works as Escape by default in Neovim
-- No additional configuration required
```

Source: vim.fandom.com

2.234 Use Symbolic Key Notation for Mappings

Category: advanced_mappings
Tags: key-mapping, notation

Use symbolic key notation for more readable and consistent key mappings across different environments

Example: Example (Vim)

```
" Examples of symbolic key notation
:map <C-R> :somecommand<CR>
:map <S-F2> :anothercommand<CR>
```

Example: Example (Neovim)

```
-- Examples of symbolic key notation
vim.keymap.set('n', '<C-R>', ':somecommand<CR>')
vim.keymap.set('n', '<S-F2>', ':anothercommand<CR>')
```

Source: vim.fandom.com)

2.235 Useful Vim Mappings for Productivity

Category: advanced_mappings
Tags: key-mapping, productivity, search

Handy key mappings that improve editing efficiency and search experience

Example: Example (Vim)

```
" Productive mappings
map Y y$
nnoremap <C-L> :nohl<CR><C-L>
```

Example: Example (Neovim)

```
-- Equivalent Neovim mappings
vim.keymap.set('n', 'Y', 'y$', { desc = 'Yank to end of line' })
vim.keymap.set('n', '<C-L>', ':nohl<CR><C-L>', { desc = 'Clear search
↪  highlighting' })
```

Source: vim.fandom.com

2.236 Virtual Cursor Movement in Insert Mode

Category: advanced_mappings
Tags: insert-mode, cursor-movement, virtualedit

Allows moving cursor virtually up/down in insert mode, automatically padding with spaces when moving beyond line length

Example: Example (Vim)

```
"cursor down/up existing lines
imap <S-Down> _<Esc>mz:set ve=all<CR>i<Down>_<Esc>my`zi<Del><Esc>:set
↪  ve=<CR>`yi<Del>
imap <S-Up> _<Esc>mz:set ve=all<CR>i<Up>_<Esc>my`zi<Del><Esc>:set
↪  ve=<CR>`yi<Del>
"cursor down with a new line
imap <S-CR> _<Esc>mz:set ve=all<CR>o<C-o>`z<Down>_<Esc>my`zi<Del><Esc>:set
↪  ve=<CR>`yi<Del>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<S-Down>', function()
  local cursor = vim.api.nvim_win_get_cursor(0)
  vim.o.virtualedit = 'all'
  vim.cmd.normal('i<Down>')
  vim.o.virtualedit = ''
  vim.api.nvim_win_set_cursor(0, cursor)
end, { desc = 'Virtual cursor down in insert mode' })

vim.keymap.set('i', '<S-Up>', function()
  local cursor = vim.api.nvim_win_get_cursor(0)
  vim.o.virtualedit = 'all'
  vim.cmd.normal('i<Up>')
  vim.o.virtualedit = ''
  vim.api.nvim_win_set_cursor(0, cursor)
end, { desc = 'Virtual cursor up in insert mode' })
```

Source: vim.fandom.com

2.237 Visible Screen ROT13 Mapping with Cursor Preservation

Category: advanced_mappings
Tags: key-mapping, text-transformation, cursor-position

ROT13 encode only the visible screen while maintaining cursor position

Example: Example (Vim)

```
map <F3> mzHVLg?`z
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F3>', 'mzHVLg?`z', { desc = 'ROT13 visible screen with
↪  cursor position' })
```

Source: vim.fandom.com

2.238 Visual Mode Smart Paste Without Register Overwrite

Category: advanced_mappings
Tags: visual-mode, registers, clipboard

Provides a smart paste mapping in visual mode that preserves the original copied text in the register

Example: Example (Vim)

```
xnoremap <silent> p p:let @"=@0<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('x', 'p', 'p<cmd>let @"=@0<CR>', { silent = true })
```

Source: vim.fandom.com

2.239 Visual Selection Case Conversion

Category: advanced_mappings
Tags: case-conversion, text-transformation, visual-mode

Provides flexible commands to convert case styles within visual selections or entire lines, supporting both CamelCase and snake_case transformations

Example: Example (Vim)

```
function! s:Camelize(range) abort
  if a:range == 0
    s#\(\%(\_l\+\)\%(_\)\@=\)\|_\(\l\)#\u\1\2#g
  else
    s#\%V\(\%(\_l\+\)\%(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g
  endif
endfunction

command! -range CamelCase silent! call <SID>Camelize(<range>)
```

Example: Example (Neovim)

```
function CamelCase(range)
  local start_line, end_line
  if range == 0 then
    -- Convert entire line
    vim.cmd('s#\v(\l+)_|(\l)#\u\1\2#g')
  else
    -- Convert within visual selection
    vim.cmd('s#\%V\v(\l+)_|(\l)\%V#\u\1\2#g')
  end
end

-- Create a Lua command
vim.api.nvim_create_user_command('CamelCase', function()
  CamelCase(vim.fn.line('v') == vim.fn.line('.') and 0 or 1)
end, { range = true })
```

Source: vim.fandom.com

2.240 Visual mode mappings

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.keymap.set('v', '<leader>s', '<Cmd>sort<CR>')
vim.keymap.set('v', '<leader>u', '<Cmd>!uniq<CR>')
vim.keymap.set('v', '*', 'y/\\V<C-r>"<CR>')
vim.keymap.set('v', '#', 'y?\\V<C-r>"<CR>')
```

Source: Community contributed

2.241 Windows-style Word Navigation Shortcuts

Category: advanced_mappings
Tags: key-mapping, navigation, windows-compatibility

Add Ctrl+Left/Right navigation to move between words like in Windows, supporting nor-

mal, visual, and insert modes

Example: Example (Vim)

```
"Edit mapping (make cursor keys work like in Windows: <C-Left><C-Right>
"Move to next word.
nnoremap <C-Left> b
vnoremap <C-S-Left> b
nnoremap <C-S-Left> gh<C-O>b
inoremap <C-S-Left> <C-\><C-O>gh<C-O>b

nnoremap <C-Right> w
vnoremap <C-S-Right> w
nnoremap <C-S-Right> gh<C-O>w
inoremap <C-S-Right> <C-\><C-O>gh<C-O>w
```

Example: Example (Neovim)

```
-- Windows-style word navigation
vim.keymap.set('n', '<C-Left>', 'b', { desc = 'Move back a word' })
vim.keymap.set('v', '<C-S-Left>', 'b', { desc = 'Select and move back a
↪  word' })
vim.keymap.set('n', '<C-S-Left>', 'gh<C-O>b', { desc = 'Move back a word in
↪  select mode' })
vim.keymap.set('i', '<C-S-Left>', '<C-\><C-O>gh<C-O>b', { desc = 'Move back
↪  a word in insert mode' })

vim.keymap.set('n', '<C-Right>', 'w', { desc = 'Move forward a word' })
vim.keymap.set('v', '<C-S-Right>', 'w', { desc = 'Select and move forward a
↪  word' })
vim.keymap.set('n', '<C-S-Right>', 'gh<C-O>w', { desc = 'Move forward a word
↪  in select mode' })
vim.keymap.set('i', '<C-S-Right>', '<C-\><C-O>gh<C-O>w', { desc = 'Move
↪  forward a word in insert mode' })
```

Source: vim.fandom.com

2.242 Write Current Key Mappings to File

Category: advanced_mappings
Tags: mapping, export, configuration

Redirect and save all current key mappings to a text file for reference or backup

Example: Example (Vim)

```
:redir! > vim_keys.txt
:silent verbose map
:redir END
```

Example: Example (Neovim)

```
-- Lua equivalent for saving key mappings
vim.cmd('redir! > vim_keys.txt')
vim.cmd('silent verbose map')
vim.cmd('redir END')
```

Source: vim.fandom.com

CHAPTER 3

Advanced neovim

3.1 Advanced PHP Documentation Lookup Function

Category: advanced_neovim
Tags: documentation, function-lookup, php

Create a sophisticated Vim function to fetch and display PHP function documentation in a preview window

Example: Example (Vim)

```
function! OpenPhpFunction (keyword)
  let proc_keyword = substitute(a:keyword , '_', '-', 'g')
  exe 'pedit'
  exe 'wincmd P'
  exe 'enew'
  exe 'set buftype=nofile'
  exe 'silent r!lynx -dump http://php.net/'.proc_keyword
endfunction
```

Example: Example (Neovim)

```
function _G.open_php_function(keyword)
  local proc_keyword = keyword:gsub('_', '-')
  vim.cmd('pedit')
  vim.cmd('wincmd P')
  vim.cmd('enew')
  vim.bo.buftype = 'nofile'
  local result = vim.fn.systemlist('lynx -dump
  ↪  http://php.net/'..proc_keyword)
  vim.api.nvim_buf_set_lines(0, 0, -1, false, result)
end

-- Optional mapping
vim.keymap.set('n', '<C-p>', function()
  _G.open_php_function(vim.fn.expand('<cword>'))
end)
```

Source: vim.fandom.com

3.2 Advanced Syntax Group Identification Command

Category: advanced_neovim
Tags: syntax, command, highlight-debugging

Create a user command to easily check the syntax highlighting groups at the current cursor position

Example: Example (Vim)

```
com! CheckHighlightUnderCursor echo {l,c,n ->
        \   'hi<'    . synIDattr(synID(l, c, 1), n)             . '> '
        \  .'trans<' . synIDattr(synID(l, c, 0), n)             . '> '
        \  .'lo<'    . synIDattr(synIDtrans(synID(l, c, 1)), n) . '> '
        \ }(line("."), col("."), "name")
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('CheckHighlightUnderCursor', function()
  local line = vim.fn.line('.')
  local col = vim.fn.col('.')
  local synID = vim.fn.synID(line, col, 1)
  local transID = vim.fn.synID(line, col, 0)
  local loID = vim.fn.synIDtrans(synID)

  print(string.format(
    "hi<%s> trans<%s> lo<%s>",
    vim.fn.synIDattr(synID, 'name'),
    vim.fn.synIDattr(transID, 'name'),
    vim.fn.synIDattr(loID, 'name')
  ))
end, {})
```

Source: vim.fandom.com

3.3 Automatic Desktop-Specific Vim Instances

Category: advanced_neovim
Tags: server-mode, multi-desktop, automation

Create a script to launch Vim instances specific to the current desktop, ensuring better window management

Example: Example (Vim)

```
# Bash script to manage desktop-specific Vim instances
exec /usr/bin/vim --servername $desktop --remote-tab-silent "$@"
```

Example: Example (Neovim)

```
-- Lua equivalent requires external script or system integration
-- Can use vim.fn.serverlist() and custom desktop detection logic
```

Source: vim.fandom.com

3.4 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: Example (Vim)

```
: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'
```

Source: Community contributed

3.5 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: Example (Vim)

```
" Vimscript:
: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.
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.inccommand = 'split'     -- preview in split window
vim.opt.inccommand = 'nosplit'   -- preview inline
-- Then use: :%s/old/new/g to see live preview while typing
-- Preview works with :substitute, :global, :sort, etc.
```

Source: Community contributed

3.6 Custom Keyword Completion with Wildcards

Category: advanced_neovim
Tags: completion, custom-functions, text-editing

Implement a custom keyword completion function that allows flexible matching using wildcards, useful for completing identifiers with mixed case or complex naming patterns

Example: Example (Vim)

```
" Custom keyword completion script
setlocal completefunc=MKwCompleteId

" Usage: Type part of a keyword and use CTRL-X CTRL-U to complete
```

Example: Example (Neovim)

```
-- Lua equivalent for custom keyword completion
function MKwCompleteId(findstart, base)
  if findstart == 1 then
    -- Find start of keyword match
    local line = vim.api.nvim_get_current_line()
    local col = vim.api.nvim_win_get_cursor(0)[2]
    local start = vim.fn.match(line:sub(1, col), '\k*$')
    return start
  else
    -- Find matching keywords
    local matches = {}
    local pattern = '\<' .. base:gsub('.', '\k*%0') .. '\>'

    -- Search for matches in current buffer
    for _, line in ipairs(vim.api.nvim_buf_get_lines(0, 0, -1, false)) do
      for match in line:gmatch(pattern) do
        table.insert(matches, match)
      end
    end

    return matches
  end
end

-- Set the custom completion function
vim.bo.completefunc = 'v:lua.MKwCompleteId'
```

Source: vim.fandom.com

3.7 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: Example (Vim)

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

Example: Example (Neovim)

```
-- Lua:
local function my_completion(findstart, base)
  if findstart == 1 then
    return vim.fn.col('.') - 1
  else
    return {'custom1', 'custom2', 'custom3'}
  end
end

vim.opt.omnifunc = 'v:lua.my_completion'
-- Or set it globally:
_G.my_completion = my_completion
vim.opt.omnifunc = 'v:lua.my_completion'
```

Source: Community contributed

3.8 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: Example (Vim)

```
: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
```

Source: Community contributed

3.9 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: Example (Vim)

```
: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
```

Source: Community contributed

3.10 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: Example (Vim)

```
: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
```

Source: Community contributed

3.11 Filetype detection API

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

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

Example: Example (Vim)

```
:lua vim.filetype.add({
  extension = { log = 'log', conf = 'conf' },
  filename = { ['.eslintrc'] = 'json' },
```

```
  pattern = { ['.*%.env%..*'] = 'sh' }
})
```

Source: Community contributed

3.12 Git Merge Conflict Resolution with Vimdiff

Category: advanced_neovim
Tags: git, merge-conflicts, diff-tool

Efficiently resolve Git merge conflicts using a custom Vimdiff layout that focuses on two-way diffs and provides a clearer view of conflicts

Example: Example (Vim)

```
# Git configuration for custom mergetool
git config --global merge.tool diffconflicts
git config --global mergetool.diffconflicts.cmd 'diffconflicts vim $BASE
↪  $LOCAL $REMOTE $MERGED'
git config --global mergetool.diffconflicts.trustExitCode true
git config --global mergetool.keepBackup false
```

Example: Example (Neovim)

```
-- Git configuration can remain the same
-- Use with a custom diffconflicts script
-- Key is to set up a three-tab layout for conflict resolution
```

Source: vim.fandom.com

3.13 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: Example (Vim)

```
: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 = ' '
```

Source: Community contributed

3.14 Health check system

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

Use Neovim's health check system to create custom health checks for your configurations and environments.

Example: Example (Vim)

```
: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
```

Source: Community contributed

3.15 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: Example (Vim)

```
: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))
```

Source: Community contributed

3.16 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: Example (Vim)

```
: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
```

Source: Community contributed

3.17 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: Example (Vim)

```
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
```

Source: Community contributed

3.18 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: Example (Vim)

```
" 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
```

Source: Community contributed

3.19 Namespace management

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

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

or plugins.

Example: Example (Vim)

```
: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
```

Source: Community contributed

3.20 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: Example (Vim)

```
: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
```

Source: Community contributed

3.21 Quick Number Value Inspection

Category: advanced_neovim
Tags: number-conversion, inspection, debugging

Add a custom mapping to quickly inspect decimal and hexadecimal values of a number under the cursor

Example: Example (Vim)

```
function! DecAndHex(number)
  if a:number =~? '^[-+]?\d\+$'
    let dec = a:number
    echo dec . printf('  ->  0x%X, -(0x%X)', dec, -dec)
  elseif a:number =~? '^0x\x\+$'
    let hex = substitute(a:number, '0x', '', '')
    echo '0x' . hex . printf('  ->  %d', eval('0x'.hex))
  endif
endfunction

nnoremap gn :call DecAndHex(expand("<cWORD>"))<CR>
```

Example: Example (Neovim)

```
local function dec_and_hex(number)
  local dec = tonumber(number)
  local hex = tonumber(number, 16)

  if dec then
    print(string.format('%d -> 0x%X, -(0x%X)', dec, dec, -dec))
  elseif hex then
    print(string.format('0x%X -> %d', hex, hex))
  else
    print('Not a valid number')
  end
end

vim.keymap.set('n', 'gn', function()
  local word = vim.fn.expand('<cWORD>')
  dec_and_hex(word)
end)
```

Source: vim.fandom.com

3.22 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: Example (Vim)

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

Source: Community contributed

3.23 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: Example (Vim)

```
: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
```

Source: Community contributed

3.24 Runtime path manipulation

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

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

Example: Example (Vim)

```
: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
```

Source: Community contributed

3.25 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: Example (Vim)

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

Source: Community contributed

3.26 Smart Function Signature Preview

Category: advanced_neovim
Tags: function-preview, autocomplete, code-insight

Automatically preview function signatures when typing, similar to IDE code insight features. This helps developers quickly understand function parameters without leaving the current editing context.

Example: Example (Vim)

```
function! PreviewWord()
  if &previewwindow
    return
  endif
  let w = expand("<cword>")
  if w =~ '\i'
    exe "silent! ptag " . w
    silent! wincmd P
    if &previewwindow
      if has("folding")
        silent! .foldopen
      endif
      call search('\<\V' . w . '\>')
      hi previewWord term=bold ctermbg=green guibg=green
      exe 'match previewWord "\%' . line(".") . 'l\%' . col(".") . 'c\k*"'
      wincmd p
    endif
  endif
endfunction

au! CursorHold *.[ch] nested call PreviewWord()
```

Example: Example (Neovim)

```
local function preview_word()
  -- Check if already in preview window
  if vim.wo.previewwindow then
    return
  end

  -- Get word under cursor
  local word = vim.fn.expand('<cword>')

  -- Only process if word is a valid identifier
  if word:match('\w+') then
    -- Try to preview tag
    vim.cmd('silent! ptag ' .. word)

    -- Switch to preview window and highlight
    vim.cmd('silent! wincmd P')
    if vim.wo.previewwindow then
      -- Open any folded sections
      if vim.o.foldenable then
        vim.cmd('silent! .foldopen')
      end

      -- Highlight the word
      vim.api.nvim_buf_add_highlight(0, -1, 'PreviewWord', 0, 0, -1)
```

```
      vim.cmd('wincmd p')
    end
  end
end

-- Attach to C/H files
vim.api.nvim_create_autocmd('CursorHold', {
  pattern = {'*.c', '*.h'},
  callback = preview_word
})
```

Source: vim.fandom.com

3.27 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: Example (Vim)

```
: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+
```

Source: Community contributed

3.28 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: Example (Vim)

```
: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)
```

Source: Community contributed

3.29 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: Example (Vim)

```
: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
```

Source: Community contributed

3.30 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: Example (Vim)

```
: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
```

Source: Community contributed

3.31 Use Lists for Handling Newlines in Vim Scripts

Category: advanced_neovim
Tags: vim-script, string-manipulation, lists

Prefer using lists over concatenated strings with newline separators to avoid NULL character issues when handling line breaks

Example: Example (Vim)

```
" Vim way of handling newlines
call setline(".", ["a", "b"])
```

```
" Instead of
let @a = "a" . "\n" . "b"
```

Example: Example (Neovim)

```
-- Lua equivalent for handling newlines
vim.fn.setline(".", {"a", "b"})

-- Recommended list-based approach
local lines = {"a", "b"}
```

Source: vim.fandom.com

3.32 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: Example (Vim)

```
: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'
```

Source: Community contributed

3.33 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: Example (Vim)

```
: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
```

Source: Community contributed

3.34 Window configuration API

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

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

Example: Example (Vim)

```
: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))
```

Source: Community contributed

3.35 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: Example (Vim)

```
: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
```

Source: Community contributed

CHAPTER 4

Advanced options

4.1 Automatic session restoration

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

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

4.2 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.opt.textwidth = 80
vim.opt.formatoptions += t
vim.opt.formatoptions += c
vim.opt.formatoptions += r
```

4.3 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.backupdir = '~/.vim/backup//'
vim.opt.directory = '~/.vim/swap//'
vim.opt.undodir = '~/.vim/undo//'
```

4.4 Clipboard integration

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.clipboard = 'unnamedplus'
vim.opt.clipboard = 'unnamed'
vim.opt.clipboard = { 'unnamed', 'unnamedplus' }
```

4.5 Complete options configuration

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.opt.completeopt = 'menu,menuone,noselect,preview'
```

4.6 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.cursorline = true
vim.opt.cursorcolumn = true
```

4.7 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.diffopt = 'internal,filler,closeoff,hiddenoff,algorithm:patience'
```

4.8 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.foldcolumn = 4
vim.opt.foldcolumn = 0
```

4.9 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.inccommand = 'split'
vim.opt.inccommand = 'nosplit'
```

4.10 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.linebreak = true
vim.opt.breakeat = ' \t!@*-+;:,./?'
vim.opt.showbreak = '>> '
```

4.11 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.mouse = 'a'
vim.opt.mouse = 'n'
vim.opt.mouse = ''
```

4.12 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. Or:

Example: Example (Vim)

```
:set undofile
:set undodir=~/.vim/undodir
" Undo history persists even after closing files
```

Example: Example (Neovim)

```
vim.opt.undofile = true
vim.opt.undodir = '~/.vim/undodir
```

4.13 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. Or:

Example: Example (Vim)

```
: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)
```

Example: Example (Neovim)

```
vim.opt.scrolloff = 8
vim.opt.sidescrolloff = 8
vim.opt.scrolloff = 999
```

4.14 Search highlighting timeout

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.hlsearch = true

vim.api.nvim_create_autocmd('CursorHold', {
    pattern = '*',
    command = 'set nohlsearch'
})

vim.api.nvim_create_autocmd('CmdLineEnter', {
    pattern = { '/', '?' },
    command = 'set hlsearch'
})
```

4.15 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.list = true
vim.opt.listchars = 'tab:> ,eol:$,trail:.,space:.'
```

4.16 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.number = true
vim.opt.relativenumber = true
```

4.17 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.ignorecase = 'smartcase'
```

4.18 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`. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.opt.spell = true
vim.opt.spelllang = 'en_us'
vim.opt.spellfile = '~/.config/nvim/spell/en.utf-8.add'
```

4.19 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.virtualedit = 'all'
vim.opt.virtualedit = 'block'
vim.opt.virtualedit = 'insert'
vim.opt.virtualedit = "block,insert"
```

4.20 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.wildmenu = true
vim.opt.wildmode = "longest,full"
```

CHAPTER 5

Advanced search patterns

5.1 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: Example (Vim)

```
/^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
```

5.2 Atom and group matching

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

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

Example: Example (Vim)

```
/\(\w\+\)\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
```

5.3 Branch and alternation

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

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

Example: Example (Vim)

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

5.4 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: Example (Vim)

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

5.5 Change ”Last, First” to ”First Last”

Category: Advanced Search
Tags: search, replace, regex

You have a list of names in this form: You want to change it to: This can be done with just one command:

Example: Example (Vim)

```
Doe, John
Smith, Alan

John Doe
Alan Smith

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

5.6 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: Example (Vim)

```
/[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
```

5.7 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: Example (Vim)

```
/\%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
```

5.8 Composing complex patterns

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

Combine multiple regex features for sophisticated pattern matching.

Example: Example (Vim)

```
/\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
```

5.9 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: Example (Vim)

```
/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'
```

5.10 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: Example (Vim)

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

5.11 Multiline pattern matching

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

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

Example: Example (Vim)

```
/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
```

5.12 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: Example (Vim)

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

5.13 Pattern modifiers and flags

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

Use various flags to modify search behavior and pattern interpretation.

Example: Example (Vim)

```
/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'
```

5.14 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: Example (Vim)

```
/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})
```

5.15 Recursive patterns

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

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

Example: Example (Vim)

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

5.16 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: You can now use your own custom function to capitalize every word in the document using `%s/<pattern>/\=v:lua.your_fun(submatch(0))/`. 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))` Credits: Different-Ad-8707@Reddit

Example: Example (Vim)

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

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

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

5.17 Search and replace with expressions

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

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

Example: Example (Vim)

```
:%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
```

5.18 Search context and ranges

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

Use ranges and context to limit search scope effectively.

Example: Example (Vim)

```
:+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
```

5.19 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: Example (Vim)

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

5.20 Search in specific file types

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

Combine search with file patterns for targeted searching.

Example: Example (Vim)

```
: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
```

5.21 Search with confirmation

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

Use the c flag in substitute commands for interactive confirmation. Prompts: yes, no, all, quit, last, ^E scroll down, ^Y scroll up.

Example: Example (Vim)

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

5.22 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: Example (Vim)

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

5.23 Very magic mode shortcuts

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

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

Example: Example (Vim)

```
/\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
```

5.24 Virtual column matching

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

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

Example: Example (Vim)

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

5.25 Zero-width assertions

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

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

Example: Example (Vim)

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

CHAPTER 6

Advanced text manipulation

6.1 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: Example (Vim)

```
" 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
```

6.2 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: Example (Vim)

```
" 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)
```

6.3 Expression register for calculations

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

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

Example: Example (Vim)

```
" 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
```

6.4 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: Example (Vim)

```
" 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 7

Advanced tricks

7.1 AES256 File Encryption with GPG

Category: advanced_tricks
Tags: encryption, security, file-operations

Use GPG to encrypt and decrypt files with AES256 encryption directly from Vim

Example: Example (Vim)

```
function Scramble()
  %!gpg -q --cipher-algo aes256 --symmetric --armor 2>/dev/null
endfunction

function Unscramble()
  %!gpg -q --cipher-algo aes256 --decrypt --armor 2>/dev/null
endfunction
```

Example: Example (Neovim)

```
function _G.encrypt_file()
  vim.cmd('%!gpg -q --cipher-algo aes256 --symmetric --armor 2>/dev/null')
end

function _G.decrypt_file()
  vim.cmd('%!gpg -q --cipher-algo aes256 --decrypt --armor 2>/dev/null')
end

vim.keymap.set('n', '<F7>', _G.encrypt_file, { desc = 'Encrypt file' })
vim.keymap.set('n', '<F8>', _G.decrypt_file, { desc = 'Decrypt file' })
```

Source: vim.fandom.com

7.2 Auto Highlight Current Word When Idle

Category: advanced_tricks
Tags: highlighting, search, cursor-tracking

Automatically highlight all occurrences of the word under the cursor when idle, useful for exploring unfamiliar source code

Example: Example (Vim)

```
" Highlight all instances of word under cursor
nnoremap z/ :if AutoHighlightToggle()<Bar>set hls<Bar>endif<CR>
function! AutoHighlightToggle()
  let @/ = ''
  if exists('#auto_highlight')
    au! auto_highlight
    augroup! auto_highlight
    setl updatetime=4000
    echo 'Highlight current word: off'
    return 0
  else
    augroup auto_highlight
      au!
      au CursorHold * let @/ = '\V\<'.escape(expand('<cword>'), '\').'\'>
    augroup end
    setl updatetime=500
    echo 'Highlight current word: ON'
    return 1
  endif
endfunction
```

Example: Example (Neovim)

```
local function auto_highlight_toggle()
  -- Toggle highlighting of current word
  local ns = vim.api.nvim_create_namespace('auto_highlight')

  if vim.b.auto_highlight then
    -- Turn off highlighting
    vim.api.nvim_buf_clear_namespace(0, ns, 0, -1)
    vim.b.auto_highlight = false
    vim.o.updatetime = 4000
    print('Highlight current word: off')
  else
    -- Turn on highlighting
    vim.api.nvim_create_autocmd('CursorHold', {
      group = vim.api.nvim_create_augroup('AutoHighlight', { clear = true
      ↪  }),
      callback = function()
        local word = vim.fn.expand('<cword>')
        vim.fn.setreg('/', '\V\<' .. vim.fn.escape(word, '\') .. '\>')
        vim.o.hlsearch = true
      end
    })
    vim.o.updatetime = 500
    vim.b.auto_highlight = true
    print('Highlight current word: ON')
  end
end

-- Optional: Create a keymap to toggle
vim.keymap.set('n', 'z/', auto_highlight_toggle, { desc = 'Toggle auto word
↪  highlight' })
```

Source: vim.fandom.com

7.3 Decode MIME Base64 and Quoted-Printable Encoding

Category: advanced_tricks
Tags: encoding, perl, text-processing

Add custom commands to decode MIME-encoded text directly in Vim using Perl modules

Example: Example (Vim)

```
command! -range=% Decode64 :w | <line1>,<line2>delete | let foo = @"
 \ | perl use MIME::Base64 (); my $foo=VIM::Eval(foo); my ($r,
 ↪  $c)=$curwin->Cursor(); $curbuf->Append($r-1, split '\n',
 ↪  MIME::Base64::decode($foo));

command! -range=% DecodeQP :w | <line1>,<line2>delete | let foo = @"
 \ | perl use MIME::QuotedPrint (); my $foo=VIM::Eval(foo); my ($r,
 ↪  $c)=$curwin->Cursor(); $curbuf->Append($r-1, split '\n',
 ↪  MIME::QuotedPrint::decode_qp($foo));
```

Example: Example (Neovim)

```
-- Note: Requires Vim compiled with Perl support
vim.api.nvim_create_user_command('Decode64', function()
  local lines = vim.api.nvim_buf_get_lines(0, 0, -1, false)
  local decoded_lines = vim.fn.system('perl -MMIME::Base64 -e "print
  ↪  decode_base64(join(\'\', <>))"', lines)
  vim.api.nvim_buf_set_lines(0, 0, -1, false, vim.split(decoded_lines,
  ↪  '\n'))
end, {})

vim.api.nvim_create_user_command('DecodeQP', function()
  local lines = vim.api.nvim_buf_get_lines(0, 0, -1, false)
  local decoded_lines = vim.fn.system('perl -MMIME::QuotedPrint -e "print
  ↪  decode_qp(join(\'\', <>))"', lines)
  vim.api.nvim_buf_set_lines(0, 0, -1, false, vim.split(decoded_lines,
  ↪  '\n'))
end, {})
```

Source: vim.fandom.com

7.4 Enhanced Hex Mode Editing in Vim

Category: advanced_tricks
Tags: hex-editing, binary-files, file-manipulation

Provides an improved method for editing binary files in hex mode using xxd, with easy toggling and automatic conversion

Example: Example (Vim)

```
" Toggle hex mode command
command -bar Hexmode call ToggleHex()

function ToggleHex()
  let l:modified=&mod
  let l:oldreadonly=&readonly
  let &readonly=0
  let l:oldmodifiable=&modifiable
  let &modifiable=1
  if !exists("b:editHex") || !b:editHex
    let b:oldft=&ft
    let b:oldbin=&bin
    setlocal binary
    silent :e
    let &ft="xxd"
    let b:editHex=1
    %!xxd
  else
    let &ft=b:oldft
    if !b:oldbin
      setlocal nobinary
    endif
    let b:editHex=0
    %!xxd -r
  endif
  let &mod=l:modified
  let &readonly=l:oldreadonly
  let &modifiable=l:oldmodifiable
endfunction

" Mapping to toggle hex mode
nnoremap <C-H> :Hexmode<CR>
```

Example: Example (Neovim)

```
local function toggle_hex()
  local modified = vim.o.modified
  local readonly = vim.o.readonly
  local modifiable = vim.o.modifiable

  vim.o.readonly = false
  vim.o.modifiable = true

  if not vim.b.edit_hex then
    vim.b.old_ft = vim.o.filetype
    vim.b.old_bin = vim.o.binary
    vim.o.binary = true
    vim.cmd('silent e')
    vim.o.filetype = 'xxd'
    vim.b.edit_hex = true
    vim.cmd('%!xxd')
  else
    vim.o.filetype = vim.b.old_ft
```

```
    if not vim.b.old_bin then
      vim.o.binary = false
    end
    vim.b.edit_hex = false
    vim.cmd('%!xxd -r')
  end

  vim.o.modified = modified
  vim.o.readonly = readonly
  vim.o.modifiable = modifiable
end

-- Create command and mapping
vim.api.nvim_create_user_command('Hexmode', toggle_hex, {})
vim.keymap.set('n', '<C-H>', ':Hexmode<CR>', { noremap = true, silent = true
↪  })
```

Source: vim.fandom.com

7.5 Preview HTML Text in Vim Buffer

Category: advanced_tricks
Tags: text-extraction, web-preview, buffer-management

Extract and view the text content of an HTML file or URL directly in a Vim buffer using a text-based browser

Example: Example (Vim)

```
function! ViewHtmlText(url)
  if !empty(a:url)
    new
    setlocal buftype=nofile bufhidden=hide noswapfile
    execute 'r !elinks ' . a:url . ' -dump -dump-width ' . winwidth(0)
    1d
  endif
endfunction

" Mappings for different preview scenarios
nnoremap <Leader>H :update<Bar>call ViewHtmlText(expand('%:p'))<CR>
vnoremap <Leader>h y:call ViewHtmlText(@@)<CR>
nnoremap <Leader>h :call ViewHtmlText(@+)<CR>
```

Example: Example (Neovim)

```
function _G.view_html_text(url)
  if url ~= '' then
    vim.cmd('new')
    vim.api.nvim_buf_set_option(0, 'buftype', 'nofile')
    vim.api.nvim_buf_set_option(0, 'bufhidden', 'hide')
    vim.api.nvim_buf_set_option(0, 'swapfile', false)
```

```
    local output = vim.fn.system('elinks ' .. url .. ' -dump -dump-width '
    ↪  .. vim.fn.winwidth(0))
    vim.api.nvim_buf_set_lines(0, 0, 1, false, vim.split(output, '\n'))
    vim.cmd('normal! gg')
  end
end

-- Mappings for different preview scenarios
vim.keymap.set('n', '<Leader>H', function()
  vim.cmd.update()
  _G.view_html_text(vim.fn.expand('%:p'))
end)

vim.keymap.set('v', '<Leader>h', function()
  local selected_text = vim.fn.getreg('*')
  _G.view_html_text(selected_text)
end)

vim.keymap.set('n', '<Leader>h', function()
  _G.view_html_text(vim.fn.getreg('+'))
end)
```

Source: vim.fandom.com

CHAPTER 8

Autocommands

8.1 Advanced HTML Tag Completion Function

Category: autocmds
Tags: html, auto-completion, insert-mode

Comprehensive function to auto-complete HTML tags with additional features like indentation

Example: Example (Vim)

```
function s:CompleteTags()
  inoremap <buffer> > ></<C-x><C-o><Esc>:startinsert!<CR><C-O>?</<CR>
  inoremap <buffer> ><Leader> >
  inoremap <buffer> ><CR>
  ↪  ></<C-x><C-o><Esc>:startinsert!<CR><C-O>?</<CR><CR><Tab><CR><Up><C-O>$
endfunction
autocmd BufRead,BufNewFile *.html,*.js,*.xml call s:CompleteTags()
```

Example: Example (Neovim)

```
-- Lua equivalent for advanced HTML tag completion
local function complete_tags()
  vim.api.nvim_buf_set_keymap(0, 'i', '>',
  ↪  '>/<C-x><C-o><Esc>:startinsert!<CR><C-O>?</<CR>', { noremap = true,
  ↪  silent = true })
  vim.api.nvim_buf_set_keymap(0, 'i', '><Leader>', '>', { noremap = true,
  ↪  silent = true })
  vim.api.nvim_buf_set_keymap(0, 'i', '><CR>', '>/<C-x><C-o><Esc>:startinse
  ↪  rt!<CR><C-O>?</<CR><CR><Tab><CR><Up><C-O>$', { noremap = true, silent
  ↪  = true })
end

vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = {'*.html', '*.js', '*.xml'},
  callback = complete_tags
})
```

Source: vim.fandom.com

8.2 Always Start on First Line of Git Commit

Category: autocmds
Tags: git, commit-message, cursor-positioning

Automatically position cursor at the first line of git commit messages, overriding Vim's default cursor restoration

Example: Example (Vim)

```
autocmd FileType gitcommit call setpos('.', [0, 1, 1, 0])
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'gitcommit',
  callback = function()
    vim.fn.setpos('.', {0, 1, 1, 0})
  end
})
```

Source: vim.fandom.com

8.3 Append SQL Commands to History File

Category: autocmds
Tags: sql, logging, buffer-management

Automatically append SQL command buffer contents to a history file when saving

Example: Example (Vim)

```
fu! Sqllog()
  1,$-1w! >> $SQLPATH/sqlplus.history
endf
au BufWrite afiedt.buf call Sqllog()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWrite', {
  pattern = 'afiedt.buf',
  callback = function()
    local lines = vim.api.nvim_buf_get_lines(0, 0, -2, false)
    local sqlpath = vim.env.SQLPATH .. '/sqlplus.history'
    vim.fn.writefile(lines, sqlpath, 'a')
  end
})
```

Source: vim.fandom.com

8.4 Auto Open Quickfix Window After Make

Category: autocmds
Tags: build, quickfix, error-handling

Automatically open quickfix window after running make, showing compilation errors without manual intervention

Example: Example (Vim)

```
autocmd QuickFixCmdPost [^l]* nested cwindow
autocmd QuickFixCmdPost l* nested lwindow
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('QuickFixCmdPost', {
  pattern = '[^l]*',
  nested = true,
  callback = function()
    vim.cmd('cwindow')
  end
})

vim.api.nvim_create_autocmd('QuickFixCmdPost', {
  pattern = 'l*',
  nested = true,
  callback = function()
    vim.cmd('lwindow')
  end
})
```

Source: vim.fandom.com

8.5 Auto Save Files When Focus Is Lost

Category: autocmds
Tags: auto-save, file-management, workflow

Automatically save all modified files when Vim loses focus, reducing manual save steps and preventing data loss

Example: Example (Vim)

```
:au FocusLost * silent! wa
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FocusLost', {
  pattern = '*',
```

```
  callback = function()
    vim.cmd('silent! wa')
  end
})
```

Source: vim.fandom.com

8.6 Auto-Close Completion Preview Automatically

Category: autocmds
Tags: completion, preview, auto-close

Automatically close the preview window when moving cursor or leaving insert mode

Example: Example (Vim)

```
autocmd CursorMovedI * if pumvisible() == 0|pclose|endif
autocmd InsertLeave * if pumvisible() == 0|pclose|endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'CursorMovedI', 'InsertLeave'}, {
  callback = function()
    if vim.fn.pumvisible() == 0 then
      vim.cmd('pclose')
    end
  end
})
```

Source: vim.fandom.com

8.7 Auto-Close Quickfix Window When Last Buffer

Category: autocmds
Tags: window-management, quickfix, buffer-cleanup

Automatically close Vim/Neovim when the quickfix window is the last open window, keeping your editor clean after completing tasks like compilation or search results

Example: Example (Vim)

```
au BufEnter * call MyLastWindow()
function! MyLastWindow()
  if &buftype=="quickfix"
    if winbufnr(2) == -1
      quit!
    endif
  endif
```

```
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    if vim.bo.buftype == 'quickfix' and vim.fn.winnr('$') == 1 then
      vim.cmd('quit')
    end
  end
})
```

Source: vim.fandom.com

8.8 Auto-Close Vim When Quickfix is Last Window

Category: autocmds
Tags: quickfix, window-management, automation

Automatically close Vim when the quickfix window is the only remaining window, which helps clean up after compilation or search results

Example: Example (Vim)

```
au BufEnter * call MyLastWindow()
function! MyLastWindow()
  if &buftype=="quickfix"
    if winbufnr(2) == -1
      quit!
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    if vim.bo.buftype == 'quickfix' and vim.fn.winnr('$') == 1 then
      vim.cmd('quit')
    end
  end
})
```

Source: vim.fandom.com

8.9 Auto-Generate Java Class Template

Category: autocmds
Tags: java, file-templates, autocommands

Automatically insert a basic Java class template when creating a new .java file, using the filename as the class name

Example: Example (Vim)

```
autocmd BufNewFile *.java \
  \ exe "normal Opublic class " . expand('%:t:r') . "\n{\n}\<Esc>1G"
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("BufNewFile", {
  pattern = "*.java",
  callback = function()
    local filename = vim.fn.expand('%:t:r')
    vim.api.nvim_buf_set_lines(0, 0, -1, false, {
      "public class " .. filename .. " {",
      "",
      "}"
    })
  end
})
```

Source: vim.fandom.com

8.10 Auto-Insert Image Sizes in HTML

Category: autocmds
Tags: html, perl, automation

Automatically retrieve and insert image width and height into HTML <img> tags using Perl's Image::Size module

Example: Example (Vim)

```
function! PerlImageSize()
  g/src
  normal $
  normal N
  normal 2w
  perl << EOF
  use Image::Size;
  $jo = VIM::Eval("expand('<cfile>')");
  $size = Image::Size::html_imgsize("$jo");
  VIM::Eval("setreg('a', '$size')");
EOF
```

```
  normal $
  normal "ap
endfunction
nnoremap <F4> :call PerlImageSize()<CR>
```

Example: Example (Neovim)

```
local function perl_image_size()
  -- Note: Requires Perl and Image::Size module
  vim.cmd('g/src')
  vim.cmd('normal! $N2w')
  local file = vim.fn.expand('<cfile>')

  -- This would require a Lua equivalent or external command
  -- Recommended to use a Lua image library or external command instead
end

-- Map to F4 in Neovim
vim.keymap.set('n', '<F4>', perl_image_size, { desc = 'Insert image sizes in
↪  HTML' })
```

Source: vim.fandom.com)

8.11 Auto-Insert Java Package Name

Category: autocmds
Tags: java, package-management, file-templates

Automatically detect and insert the correct package name based on the file's directory structure

Example: Example (Vim)

```
function! InsertJavaPackage()
  let dir = getcwd()
  let dir = substitute(dir, "^.*\/src\/", "", "")
  let dir = substitute(dir, "\/", ".", "g")
  let dir = "package " . dir . ";"
  let filename = expand("%:t:r")
  let result = append(0, dir)
  let result = append(1, "")
  let result = append(2, "public class " . filename . " {")
  let result = append(3, "}")
endfunction
```

Example: Example (Neovim)

```
function InsertJavaPackage()
  local cwd = vim.fn.getcwd()
  local dir = cwd:gsub("^.*\/src\/", ""):gsub("\/", ".")
```

```
  local filename = vim.fn.expand('%:t:r')

  local lines = {
    "package " .. dir .. ";",
    "",
    "public class " .. filename .. " {",
    "}"
  }

  vim.api.nvim_buf_set_lines(0, 0, 0, false, lines)
end

vim.api.nvim_create_autocmd("BufNewFile", {
  pattern = "*.java",
  callback = InsertJavaPackage
})
```

Source: vim.fandom.com

8.12 Auto-Position Cursor for Email Editing

Category: autocmds
Tags: email-editing, cursor-placement, automation

Automatically position the cursor in the most logical place when editing email messages, such as in an empty subject line or at the start of the email body

Example: Example (Vim)

```
function FirstInPost (...) range
  let cur = a:firstline
  while cur <= a:lastline
    let str = getline(cur)
    if str == 'Subject: '
      execute cur
      :start!
      break
    endif
    if str == ''
      let cur = cur + 1
      if strlen(matchstr(getline(cur), '^On.*wrote:.*')) > 0
        let cur = cur + 1
      endif
      execute cur
      :start
      break
    endif
    let cur = cur + 1
  endwhile
endfunction

com Fip :set tw=0<Bar>:%call FirstInPost()
```

Example: Example (Neovim)

```
local function first_in_post()
  local cur = 1
  local last_line = vim.fn.line('$')
  while cur <= last_line do
    local str = vim.fn.getline(cur)
    if str == 'Subject: ' then
      vim.cmd.normal('gg' .. cur .. 'G^')
      vim.cmd.startinsert()
      return
    end
    if str == '' then
      cur = cur + 1
      if vim.fn.matchstr(vim.fn.getline(cur), '^On.*wrote:.*') ~= '' then
        cur = cur + 1
      end
      vim.cmd.normal('gg' .. cur .. 'G^')
      vim.cmd.startinsert()
      return
    end
    cur = cur + 1
  end
end

vim.api.nvim_create_user_command('Fip', first_in_post, {})
```

Source: vim.fandom.com

8.13 Auto-Save and Highlight Unsaved Buffers on Focus Lost

Category: autocmds
Tags: buffer-management, focus-events, ui-feedback

Automatically check for modified buffers when losing focus and provide visual feedback via background color

Example: Example (Vim)

```
augroup focus_buffer_check
  au FocusGained * :hi Normal guibg='#000000'
  au FocusLost * let i = 0
  au FocusLost * while i < bufnr('$')
  au FocusLost *  if getbufvar(winbufnr(i), '&modified')
  au FocusLost *   break
  au FocusLost *  endif
  au FocusLost *  let i = i+1
  au FocusLost * endwhile
  au FocusLost * if i < bufnr("$")
  au FocusLost *  :hi Normal guibg='#330000'
  au FocusLost * else
  au FocusLost *  :hi Normal guibg='#000033'
  au FocusLost *  cclose
```

```
  au FocusLost * endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('focus_buffer_check', { clear = true })

vim.api.nvim_create_autocmd('FocusGained', {
  group = 'focus_buffer_check',
  callback = function()
    vim.cmd('hi Normal guibg=#000000')
  end
})

vim.api.nvim_create_autocmd('FocusLost', {
  group = 'focus_buffer_check',
  callback = function()
    for i = 1, vim.fn.bufnr('$') do
      if vim.fn.getbufvar(vim.fn.winbufnr(i), '&modified') == 1 then
        vim.cmd('hi Normal guibg=#330000')
        return
      end
    end
    vim.cmd('hi Normal guibg=#000033')
    vim.cmd('cclose')
  end
})
```

Source: vim.fandom.com

8.14 Auto-Update Copyright Notice in Files

Category: autocmds
Tags: copyright, text-manipulation, automation

Automatically update copyright years in file headers when saving modified files, supporting single years and year ranges

Example: Example (Vim)

```
" Automatically update copyright notice with current year
autocmd BufWritePre *
  \ if &modified |
  \   exe "g#\cCOPYRIGHT \(".strftime("%Y")."\)\@![0-9]\{4\}\(-".strftime("
  ↪ %Y")."\)\@!#s#\([0-9]\{4\}\)\(-[0-9]\{4\}\)\?#\1-".strftime("%Y")
  ↪ |
  \ endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePre', {
  pattern = '*',
  callback = function()
    if vim.bo.modified then
      local current_year = os.date('%Y')
      vim.cmd(string.format(
        'silent! g/\cCOPYRIGHT %s/s/\(\d\{4\}\)\(-\d\{4\}\)\?/%s-%s/',
        current_year, '\1', current_year
      ))
    end
  end
})
```

Source: vim.fandom.com

8.15 Auto-Update ctags on File Write

Category: autocmds
Tags: ctags, project-navigation, auto-generation

Automatically update ctags file when saving source code files, keeping tags current for easy code navigation

Example: Example (Vim)

```
function! DelTagOfFile(file)
  let fullpath = a:file
  let cwd = getcwd()
  let tagfilename = cwd . "/tags"
  let f = substitute(fullpath, cwd . "/", "", "")
  let f = escape(f, './')
  let cmd = 'sed -i "/' . f . '/d" "' . tagfilename . '"'
  let resp = system(cmd)
endfunction

function! UpdateTags()
  let f = expand("%:p")
  let cwd = getcwd()
  let tagfilename = cwd . "/tags"
  let cmd = 'ctags -a -f ' . tagfilename . ' --c++-kinds=+p --fields=+iaS
  ↪ --extra=+q ' . '"' . f . '"'
  call DelTagOfFile(f)
  let resp = system(cmd)
endfunction
autocmd BufWritePost *.cpp,*.h,*.c call UpdateTags()
```

Example: Example (Neovim)

```
local function del_tag_of_file(file)
  local fullpath = file
  local cwd = vim.fn.getcwd()
  local tagfilename = cwd .. "/tags"
  local f = fullpath:gsub(cwd .. "/", "")
  f = f:gsub('([%.%/%*])', '\%1')
  local cmd = string.format('sed -i "/%s/d" "%s"', f, tagfilename)
  vim.fn.system(cmd)
end

local function update_tags()
  local f = vim.fn.expand('%:p')
  local cwd = vim.fn.getcwd()
  local tagfilename = cwd .. "/tags"
  local cmd = string.format('ctags -a -f %s --c++-kinds=+p --fields=+iaS
  ↪  --extra=+q "%s"', tagfilename, f)
  del_tag_of_file(f)
  vim.fn.system(cmd)
end

vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = {'*.cpp', '*.h', '*.c'},
  callback = update_tags
})
```

Source: vim.fandom.com

8.16 Auto-backup important files

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePre', {
  pattern = { '.vimrc', 'init.lua', 'init.vim'},
  callback = function()
    local filename = vim.fn.expand('%')
    local timestamp = os.date('%Y%m%d_%H%M%S')
    local backup_name = filename .. '.backup.' .. timestamp
```

```
    vim.cmd('write! ' .. vim.fn.fnameescape(backup_name))
  end
})
```

Source: Community contributed

8.17 Auto-chmod executable scripts

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = { '*.sh', '*.py', '*.pl', '*.rb' },
  command = 'silent !chmod +x %'
})
```

Source: Community contributed

8.18 Auto-compile on save

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
    pattern = { '*.c', '*.cpp' },
    callback = function()
        vim.cmd('!gcc % -o %:r')
    end,
})

vim.api.nvim_create_autocmd('BufWritePost', {
    pattern = { '*.tex' },
    callback = function()
        vim.cmd('!pdflatex %')
    end
})

vim.api.nvim_create_autocmd('BufWritePost', {
    pattern = { 'init.lua' },
    callback = function()
        vim.cmd('source %')
    end
})
```

Source: Community contributed

8.19 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({ 'BufWritePre' }, {
    pattern = { '*.js', '*.ts', '*.jsx', '*.tsx'},
    callback = function()
        vim.lsp.buf.format()
    end,
})

vim.api.nvim_create_autocmd({ 'BufWritePre' }, {
    pattern = '*.py',
    callback = function()
```

```
        vim.cmd('!black %')
    end,
})

vim.api.nvim_create_autocmd({ 'BufWritePre' }, {
    pattern = '*.go',
    callback = function()
        vim.cmd('!gofmt -w %')
    end,
})
```

Source: Community contributed

8.20 Auto-reload changed files

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.opt.autoread = true

vim.api.nvim_create_autocmd({ 'FocusGained', 'BufEnter', 'CurtorHold',
↪ 'CursorHoldI'}, {
    pattern = '*',
    command = 'checktime'
})

vim.api.nvim_create_autocmd('FileChangedShellPost', {
    pattern = '*',
    callback = function()
        vim.notify('File changed on disk. Buffer reloaded.',
        ↪ vim.log.levels.WARN)
    end
})
```

Source: Community contributed

8.21 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('VimResized', {
    pattern = '*',
    command = 'wincmd =',
    desc = 'Equalizes window sizes when vim is resized'
})
```

Source: Community contributed

8.22 Auto-save Buffer Periodically

Category: autocmds
Tags: auto-save, buffer-management, event-handling

Automatically save the current buffer after a specified time interval using CursorHold event, providing a non-intrusive auto-save mechanism

Example: Example (Vim)

```
au BufRead,BufNewFile * let b:start_time=localtime()

au CursorHold * call UpdateFile()

function! UpdateFile()
  if ((localtime() - b:start_time) >= 60)
    update
    let b:start_time=localtime()
  endif
endfunction

au BufWritePre * let b:start_time=localtime()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  callback = function()
    vim.b.start_time = os.time()
  end
})

vim.api.nvim_create_autocmd('CursorHold', {
  callback = function()
    if os.time() - vim.b.start_time >= 60 then
      vim.cmd('update')
      vim.b.start_time = os.time()
    end
  end
})

vim.api.nvim_create_autocmd('BufWritePre', {
  callback = function()
    vim.b.start_time = os.time()
  end
})
```

Source: vim.fandom.com

8.23 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('InsertEnter', {
    pattern = '*',
    callback = function()
        vim.bo.relativenumber = false
    end
})
vim.api.nvim_create_autocmd('InsertLeave', {
    pattern = '*',
    callback = function()
        vim.bo.relativenumber = true
    end
```

```
})
```

Source: Community contributed

8.24 Autocompile JavaScript with Closure Compiler

Category: autocmds
Tags: javascript, compilation, auto-save

Automatically compile JavaScript files using Google Closure Compiler when saving, creating a minified version

Example: Example (Vim)

```
BufWriteCmd *.js :call CompileJS()
function! CompileJS()
  if &modified
    write
    let fn = shellescape(expand('%:p'))
    let pn = shellescape(expand('%:p:h'))
    let fnm = expand('%:r.js')
    let cpa = '/home/username/closure/compiler.jar'
    execute "! java -jar " . cpa . " --js=" . fn . " --js_output_file=" . pn
    ↪  . "/" . fnm . ".min.js"
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWriteCmd', {
  pattern = '*.js',
  callback = function()
    if vim.bo.modified then
      vim.cmd('write')
      local fn = vim.fn.shellescape(vim.fn.expand('%:p'))
      local pn = vim.fn.shellescape(vim.fn.expand('%:p:h'))
      local fnm = vim.fn.expand('%:r.js')
      local cpa = '/home/username/closure/compiler.jar'
      vim.fn.system(string.format('java -jar %s --js=%s
      ↪  --js_output_file=%s/%s.min.js',
        cpa, fn, pn, fnm))
    end
  end
})
```

Source: vim.fandom.com

8.25 Autocompile LaTeX on Write

Category: autocmds
Tags: latex, autocmd, compilation

Automatically compile LaTeX document and refresh xdvi viewer when saving a .tex file

Example: Example (Vim)

```
au BufWritePost *.tex silent call Tex_RunLaTeX()
au BufWritePost *.tex silent !pkill -USR1 xdvi.bin
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = '*.tex',
  callback = function()
    vim.fn['Tex_RunLaTeX']()
    vim.fn.system('pkill -USR1 xdvi.bin')
  end
})
```

Source: vim.fandom.com

8.26 Autoconvert Spaces to Tabs in Makefiles

Category: autocmds
Tags: file-specific, makefile, formatting

Automatically convert leading 8 spaces to a tab in Makefiles when typing or pasting, ensuring consistent indentation

Example: Example (Vim)

```
" Autoconvert 8 spaces to tab in Makefiles
au FileType make :inoremap <buffer><silent><Space> <Space><c-o>:call
 ↪  MapSpaceInMakefile()<CR>

function! MapSpaceInMakefile()
  let line = getline('.')
  let col = col('.')
  if strpart(line, 0, 8) == ' '
    let new = "\t" . strpart(line,8)
    call setline('.', new )
  endif
  return ""
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'make',
  callback = function()
    vim.keymap.set('i', ' ', function()
      local line = vim.api.nvim_get_current_line()
      if line:sub(1, 8):match('^%s+$') then
        local new_line = '\t' .. line:sub(9)
        vim.api.nvim_set_current_line(new_line)
      end
      return ' '
    end, { buffer = true, expr = true })
  end
})
```

Source: vim.fandom.com

8.27 Automate File Explorer Startup and Cleanup

Category: autocmds
Tags: file-explorer, window-management, startup

Automatically open a file explorer when Vim starts and close it when it's the last window, with version-specific handling for different Vim versions

Example: Example (Vim)

```
func OpenFileWindow()
 :20vs
 :Explore
 wincmd p
endfunc

func CloseIfLast()
 if v:version >= 700
  if exists("b:netrw_curdir")
   let n = winnr()
   wincmd p
   if n == winnr()
    quit
   endif
   wincmd p
  endif
 endif
endfunc

if has("autocmd")
 autocmd VimEnter * call OpenFileWindow()
 autocmd WinEnter * call CloseIfLast()
endif
```

Example: Example (Neovim)

```
local function open_file_window()
 vim.cmd('20vs')
 vim.cmd('Explore')
 vim.cmd('wincmd p')
end

local function close_if_last()
 if vim.fn.exists('b:netrw_curdir') == 1 then
  local current_win = vim.fn.winnr()
  vim.cmd('wincmd p')
  if vim.fn.winnr() == current_win then
   vim.cmd('quit')
  end
  vim.cmd('wincmd p')
 end
end

vim.api.nvim_create_augroup('FileExplorer', { clear = true })
vim.api.nvim_create_autocmd('VimEnter', {
 group = 'FileExplorer',
 callback = open_file_window
})

vim.api.nvim_create_autocmd('WinEnter', {
 group = 'FileExplorer',
 callback = close_if_last
})
```

Source: vim.fandom.com

8.28 Automated PHP Syntax Checking on Write

Category: autocmds
Tags: php, syntax-checking, autocmd

Automatically check PHP syntax before saving the file, preventing syntax errors

Example: Example (Vim)

```
au! BufWriteCmd *.php call PHPsynCHK()

function! PHPsynCHK()
  ccl
  let winnum = winnr()
  silent execute "%!php -l -f /dev/stdin | sed
  ↪  's/\/dev\/stdin/".bufname("%").("/g') >.vimerr; cat"
  silent cf .vimerr
  cw
  execute winnum . "wincmd w"
  silent undo
  silent cf
```

```
  if 1 == len(getqflist())
    w
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWriteCmd', {
  pattern = '*.php',
  callback = function()
    vim.cmd('ccl')
    local winnum = vim.fn.winnr()
    -- Note: This is a direct translation and might need refinement
    vim.cmd("silent execute '%!php -l -f /dev/stdin | sed 's/\/dev\/stdin/"
    ↪  .. vim.fn.bufname('%') .. "/g' >.vimerr; cat'")
    vim.cmd('silent cf .vimerr')
    vim.cmd('cw')
    vim.cmd(winnum .. 'wincmd w')
    vim.cmd('silent undo')
    vim.cmd('silent cf')
    if #vim.fn.getqflist() == 1 then
      vim.cmd('w')
    end
  end
})
```

Source: vim.fandom.com

8.29 Automatic Backup on File Write

Category: autocmds
Tags: backup, file-write, auto-commands

Create a backup in a trash directory only if the file content has changed, using diff to check modifications

Example: Example (Vim)

```
set backup

augroup backups
 au!
 autocmd BufWritePost,FileWritePost * !diff -d <afile> <afile>~ &>
 ↪  /dev/null;if [[ $? == 1 ]];then touch <afile>~;mv <afile>~ `mktemp
 ↪  ~/Desktop/Trash/<afile>.XXXXXX`;fi
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('backups', { clear = true })

vim.api.nvim_create_autocmd({'BufWritePost', 'FileWritePost'}, {
  group = 'backups',
  pattern = '*',
  callback = function()
    local file = vim.fn.expand('<afile>')
    local backup = file .. '~'
    local diff_cmd = string.format('diff -d %s %s &> /dev/null', file,
    ↪  backup)

    if vim.fn.system(diff_cmd) ~= 0 then
      vim.fn.system(string.format('touch %s && mv %s $(mktemp
      ↪  ~/Desktop/Trash/%s.XXXXXX)', backup, backup, file))
    end
  end
})
```

Source: vim.fandom.com

8.30 Automatic Binary File Hex Mode Detection

Category: autocmds
Tags: file-detection, binary-files, autocmd

Automatically detect and open binary files in hex mode, with intelligent handling of file types and write operations

Example: Example (Vim)

```
augroup Binary
  au!
  au BufReadPre *.bin,*.hex setlocal binary
  au BufReadPost *
        \ if &binary | Hexmode | endif
  au BufWritePre *
        \ if exists("b:editHex") && b:editHex && &binary |
        \  let oldro=&ro | let &ro=0 |
        \  let oldma=&ma | let &ma=1 |
        \  silent exe "%!xxd -r" |
        \  let &ma=oldma | let &ro=oldro |
        \ endif
augroup END
```

Example: Example (Neovim)

```
local binary_augroup = vim.api.nvim_create_augroup('BinaryFileHandling', {
↪  clear = true })

vim.api.nvim_create_autocmd({'BufReadPre'}, {
```

```
  pattern = {'*.bin', '*.hex'},
  group = binary_augroup,
  callback = function()
    vim.bo.binary = true
  end
})

vim.api.nvim_create_autocmd({'BufReadPost'}, {
  pattern = '*',
  group = binary_augroup,
  callback = function()
    if vim.o.binary then
      vim.cmd('Hexmode')
    end
  end
})

vim.api.nvim_create_autocmd({'BufWritePre'}, {
  pattern = '*',
  group = binary_augroup,
  callback = function()
    if vim.b.edit_hex and vim.o.binary then
      local readonly = vim.o.readonly
      local modifiable = vim.bo.modifiable

      vim.o.readonly = false
      vim.bo.modifiable = true

      vim.cmd('%!xxd -r')

      vim.o.readonly = readonly
      vim.bo.modifiable = modifiable
    end
  end
})
```

Source: vim.fandom.com

8.31 Automatic C/C++ Header Include Guards

Category: autocmds
Tags: c-cpp, code-generation, header-files

Automatically insert preprocessor include guards when creating new C/C++ header files to prevent multiple inclusions

Example: Example (Vim)

```
function! s:insert_gates()
  let gatename = substitute(toupper(expand("%:t")), "\.", "_", "g")
  execute "normal! i#ifndef " . gatename
  execute "normal! o#define " . gatename . " "
```

```
  execute "normal! Go#endif /* " . gatename . " */"
  normal! kk
endfunction
autocmd BufNewFile *.{h,hpp} call <SID>insert_gates()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufNewFile', {
  pattern = {'*.h', '*.hpp'},
  callback = function()
    local filename = vim.fn.expand('%:t')
    local gatename = string.upper(filename:gsub('%.',  '_'))
    local lines = {
      '#ifndef ' .. gatename,
      '#define ' .. gatename .. ' ',
      '',
      '#endif /* ' .. gatename .. ' */'
    }
    vim.api.nvim_buf_set_lines(0, 0, 0, false, lines)
    vim.cmd('normal! 2k')
  end
})
```

Source: vim.fandom.com

8.32 Automatic Filetype Detection for ClearCase Files

Category: autocmds
Tags: version-control, file-detection, autocmd

Automatically detect file types for ClearCase version-controlled files with complex filenames

Example: Example (Vim)

```
augroup filetypedetect
 au BufNewFile,BufRead *@@/*
 \ if expand("<afile>") =~ '@@' |
 \ exe "doau filetypedetect BufRead " . expand("<afile>:s?@@.*$??") |
 \ endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('filetypedetect', { clear = true })
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*@@/*',
  group = 'filetypedetect',
  callback = function()
    local filename = vim.fn.expand('<afile>')
```

```
    if filename:match('@@') then
      local clean_filename = filename:gsub('@@.*$', '')
      vim.cmd('doautocmd filetypedetect BufRead ' .. clean_filename)
    end
  end
})
```

Source: vim.fandom.com

8.33 Automatic Git Project Session Management

Category: autocmds
Tags: session-management, git-integration, workflow

Automatically save and restore Vim sessions for each Git project, enabling quick context switching between projects

Example: Example (Vim)

```
function! FindProjectName()
  let s:name = getcwd()
  if !isdirectory(".git")
    let s:name = substitute(finddir(".git", ".;"), "/.git", "", "")
  end
  if s:name != ""
    let s:name = matchstr(s:name, ".*", strridx(s:name, "/") + 1)
  end
  return s:name
endfunction

" Sessions only restored if we start Vim without args.
function! RestoreSession(name)
  if a:name != ""
    if filereadable($HOME . "/.vim/sessions/" . a:name)
      execute 'source ' . $HOME . "/.vim/sessions/" . a:name
    end
  end
endfunction

" Sessions only saved if we start Vim without args.
function! SaveSession(name)
  if a:name != ""
    execute 'mksession! ' . $HOME . '/.vim/sessions/' . a:name
  end
endfunction

" Restore and save sessions.
if argc() == 0
  autocmd VimEnter * call RestoreSession(FindProjectName())
  autocmd VimLeave * call SaveSession(FindProjectName())
end
```

Example: Example (Neovim)

```
local function find_project_name()
  local name = vim.fn.getcwd()
  if vim.fn.isdirectory('.git') == 0 then
    name = vim.fn.substitute(vim.fn.finddir('.git', '.;'), '/.git', '', '')
  end
  if name ~= '' then
    name = vim.fn.matchstr(name, '.*', vim.fn.strridx(name, '/') + 1)
  end
  return name
end

local function restore_session(name)
  if name ~= '' then
    local session_path = vim.fn.expand('~/.vim/sessions/' .. name)
    if vim.fn.filereadable(session_path) == 1 then
      vim.cmd('source ' .. session_path)
    end
  end
end

local function save_session(name)
  if name ~= '' then
    vim.cmd('mksession! ' .. vim.fn.expand('~/.vim/sessions/' .. name))
  end
end

if vim.fn.argc() == 0 then
  vim.api.nvim_create_autocmd('VimEnter', {
    callback = function()
      restore_session(find_project_name())
    end
  })
  vim.api.nvim_create_autocmd('VimLeave', {
    callback = function()
      save_session(find_project_name())
    end
  })
end
```

Source: vim.fandom.com

8.34 Automatic Header File Template Generation

Category: autocmds
Tags: file-templates, header-files, auto-generation

Automatically generate header file templates with timestamps, filename, and boilerplate content when creating new C/C++ header files

Example: Example (Vim)

```
augroup HeaderTemplate
  autocmd BufNewFile *.h call InsertCHHeader()
autocmd BufWrite *.h call ModifyTime()
endgroup

function! InsertCHHeader()
  call InsertSkeleton('skeleton.h')
  call InsertFname()
  1
  call search('Description:')
  normal $
  startinsert
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('HeaderTemplate', { clear = true })

vim.api.nvim_create_autocmd('BufNewFile', {
  pattern = '*.h',
  group = 'HeaderTemplate',
  callback = function()
    vim.fn.InsertCHHeader()
  end
})

vim.api.nvim_create_autocmd('BufWrite', {
  pattern = '*.h',
  group = 'HeaderTemplate',
  callback = function()
    vim.fn.ModifyTime()
  end
})
```

Source: vim.fandom.com

8.35 Automatic Hex Mode for Binary Files

Category: autocmds
Tags: binary-files, file-detection, auto-conversion

Automatically detect and enter hex mode for binary files with .bin and .hex extensions

Example: Example (Vim)

```
augroup Binary
  au!
  au BufReadPre *.bin,*.hex setlocal binary
  au BufReadPost *
        \ if &binary | Hexmode | endif
```

```
  au BufWritePre *
        \ if exists("b:editHex") && b:editHex && &binary |
        \  let oldro=&ro | let &ro=0 |
        \  let oldma=&ma | let &ma=1 |
        \  silent exe "%!xxd -r" |
        \  let &ma=oldma | let &ro=oldro |
        \ endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('Binary', { clear = true })

vim.api.nvim_create_autocmd({'BufReadPre'}, {
  group = 'Binary',
  pattern = {'*.bin', '*.hex'},
  callback = function()
    vim.o.binary = true
  end
})

vim.api.nvim_create_autocmd({'BufReadPost'}, {
  group = 'Binary',
  callback = function()
    if vim.o.binary then
      vim.cmd('Hexmode')
    end
  end
})

vim.api.nvim_create_autocmd({'BufWritePre'}, {
  group = 'Binary',
  callback = function()
    if vim.b.edit_hex and vim.o.binary then
      local readonly = vim.o.readonly
      local modifiable = vim.o.modifiable
      vim.o.readonly = false
      vim.o.modifiable = true
      vim.cmd('%!xxd -r')
      vim.o.modifiable = modifiable
      vim.o.readonly = readonly
    end
  end
})
```

Source: vim.fandom.com

8.36 Automatic JavaScript Compilation with Error Handling

Category: autocmds
Tags: javascript, compilation, error-handling

Set up automatic JavaScript compilation using Closure compiler with error tracking in quickfix window

Example: Example (Vim)

```
autocmd FileType javascript compiler closure
autocmd BufWritePost *.js make!
autocmd BufWritePost *.js cwindow
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('JavaScriptCompilation', { clear = true })

vim.api.nvim_create_autocmd('FileType', {
  group = 'JavaScriptCompilation',
  pattern = 'javascript',
  command = 'compiler closure'
})

vim.api.nvim_create_autocmd('BufWritePost', {
  group = 'JavaScriptCompilation',
  pattern = '*.js',
  callback = function()
    vim.cmd('make!')
    vim.cmd('cwindow')
  end
})
```

Source: vim.fandom.com

8.37 Automatic Last Modified Timestamp Update

Category: autocmds
Tags: timestamps, file-modification, auto-update

Automatically update the 'Last Modified' timestamp when saving a header file

Example: Example (Vim)

```
function! ModifyTime()
  if &modified == 1
    let current_time = strftime('%b %d %Y %X')
    normal mi
    let modified_line_no = search('Last modified:')
    if modified_line_no != 0 && modified_line_no < 10
      exe "normal f:2l" . strlen(current_time) . "s" . current_time
      normal `i
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.ModifyTime()
  if vim.bo.modified then
    local current_time = os.date('%b %d %Y %X')
    vim.cmd('normal mi')
    local modified_line_no = vim.fn.search('Last modified:')
    if modified_line_no ~= 0 and modified_line_no < 10 then
      vim.cmd('normal f:2l' .. #current_time .. 's' .. current_time)
      vim.cmd('normal `i')
    end
  end
end

vim.cmd('function! ModifyTime() lua _G.ModifyTime() end')
```

Source: vim.fandom.com

8.38 Automatic Perforce File Checkout

Category: autocmds
Tags: version-control, workflow, automation

Automatically check out files from Perforce when modifying read-only files, with user confirmation

Example: Example (Vim)

```
function IsUnderPerforce()
  if exists("$P4HOME")
    if expand("%:p") =~ ("^" . $P4HOME)
      let b:p4path = substitute(expand("%:p"), $P4HOME, "//depot", "")
    endif
  endif
endfunction

function P4Checkout()
  if exists("b:p4path")
    if (confirm("Checkout from Perforce?", "&Yes\n&No", 1) == 1)
      call system("p4 edit " . b:p4path . " > /dev/null")
      if v:shell_error == 0
        set noreadonly
      endif
    endif
  endif
endfunction

if !exists("au_p4_cmd")
  let au_p4_cmd=1
  au BufEnter * call IsUnderPerforce()
  au FileChangedRO * call P4Checkout()
endif
```

Example: Example (Neovim)

```
local function is_under_perforce()
  local p4home = vim.env.P4HOME
  if p4home then
    local current_file = vim.fn.expand('%:p')
    if current_file:match('^' .. p4home) then
      vim.b.p4path = current_file:gsub(p4home, '//depot')
    end
  end
end

local function p4_checkout()
  if vim.b.p4path then
    local choice = vim.fn.confirm('Checkout from Perforce?', '&Yes\n&No', 1)
    if choice == 1 then
      local result = vim.fn.system('p4 edit ' .. vim.b.p4path .. ' >
      ↪  /dev/null')
      if vim.v.shell_error == 0 then
        vim.opt.readonly = false
      end
    end
  end
end

vim.api.nvim_create_augroup('PerforceAutocheckout', { clear = true })
vim.api.nvim_create_autocmd('BufEnter', {
  group = 'PerforceAutocheckout',
  callback = is_under_perforce
})
vim.api.nvim_create_autocmd('FileChangedRO', {
  group = 'PerforceAutocheckout',
  callback = p4_checkout
})
```

Source: vim.fandom.com

8.39 Automatic Tags Generation for Python Projects

Category: autocmds
Tags: tags, python, file-management

Automatically generate ctags for Python and template files on buffer write, ensuring updated tag files

Example: Example (Vim)

```
function WritePythonTags()
  let foo = system("/usr/bin/ctags --langmap=python:.py.ptl *.py *.ptl")
endfunction

au BufWritePost *.py,*.ptl call WritePythonTags()
```

Example: Example (Neovim)

```
local function write_python_tags()
  vim.fn.system("/usr/bin/ctags --langmap=python:.py.ptl *.py *.ptl")
end

vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = {'*.py', '*.ptl'},
  callback = write_python_tags
})
```

Source: vim.fandom.com

8.40 Automatic View and Fold Persistence

Category: autocmds
Tags: views, folds, persistence, buffer-management

Automatically save and restore views (including folds) when opening and closing files, preserving your folding and view state between sessions

Example: Example (Vim)

```
set viewoptions-=options
augroup vimrc
    autocmd BufWritePost *
    \   if expand('%') != '' && &buftype !~ 'nofile'
    \|      mkview
    \|  endif
    autocmd BufRead *
    \   if expand('%') != '' && &buftype !~ 'nofile'
    \|      silent loadview
    \|  endif
augroup END
```

Example: Example (Neovim)

```
vim.opt.viewoptions:remove('options')

vim.api.nvim_create_augroup('view_persistence', { clear = true })
vim.api.nvim_create_autocmd({'BufWritePost', 'BufRead'}, {
  group = 'view_persistence',
  pattern = '*',
  callback = function()
    local filename = vim.fn.expand('%')
    local buftype = vim.o.buftype
    if filename ~= '' and not buftype:match('nofile') then
      if vim.api.nvim_call_function('expand', {'%'}) ~= '' then
        if vim.api.nvim_call_function('filereadable', {filename}) == 1 then
          vim.cmd(vim.api.nvim_call_function('winnr', {'$'}) .. 'wincmd w')
          vim.cmd('silent! loadview')
        end
```

```
      end
    end
  end
})
```

Source: vim.fandom.com

8.41 Automatic XML Formatting

Category: autocmds
Tags: xml, formatting, auto-command

Automatically format XML files using Tidy whenever a file with XML filetype is opened

Example: Example (Vim)

```
" Automatically format XML files
:au FileType xml :%!tidy -i -xml --show-errors 0 2>/dev/null
```

Example: Example (Neovim)

```
-- Automatically format XML files
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'xml',
  callback = function()
    vim.cmd('%!tidy -i -xml --show-errors 0 2>/dev/null')
  end
})
```

Source: vim.fandom.com

8.42 Automatically Check for External File Changes

Category: autocmds
Tags: file-monitoring, buffer-management, autocommands

Automatically detect and reload files that have been modified externally, improving workflow when working with dynamic files like logs or collaborative documents

Example: Example (Vim)

```
" Watch for external file changes
command! -bang WatchForChanges :call WatchForChanges(@%, {'toggle': 1,
↪  'autoread': <bang>0})
```

Example: Example (Neovim)

```
-- Lua equivalent for watching file changes
vim.api.nvim_create_user_command('WatchForChanges', function(opts)
  -- Implementation would require translating the full Vimscript function
  -- Core functionality: periodically check if file changed externally
  vim.opt.autoread = true
end, { bang = true })
```

Source: vim.fandom.com

8.43 Automatically Decompile Java Class Files

Category: autocmds
Tags: java, file-reading, automation

Automatically decompile Java .class files when opened in Vim/Neovim using javap command

Example: Example (Vim)

```
function s:ReadClass(dir, classname)
  execute "cd " . a:dir
  execute "0read !javap -c " . a:classname
  1
  setlocal readonly
  setlocal nomodified
endfunction

autocmd BufReadCmd *.class
  \ call <SID>ReadClass(expand("<afile>:p:h"), expand("<afile>:t:r"))
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadCmd', {
  pattern = '*.class',
  callback = function()
    local file = vim.fn.expand('<afile>')
    local dir = vim.fn.expand('<afile>:p:h')
    local classname = vim.fn.expand('<afile>:t:r')

    vim.cmd('cd ' .. dir)
    vim.cmd('0read !javap -c ' .. classname)
    vim.cmd('1')
    vim.bo.readonly = true
    vim.bo.modified = false
  end
})
```

Source: vim.fandom.com

8.44 Automatically Open Swap Files Read-Only

Category: autocmds
Tags: swap-files, buffer-management, file-handling

Automatically open files with existing swap files in read-only mode, reducing manual intervention when a file is already open elsewhere

Example: Example (Vim)

```
func CheckSwap()
  swapname
  if v:statusmsg =~ '\.sw[^p]$'
    set ro
  endif
endfunc

if &swf
  set shm+=A
  au BufReadPre * call CheckSwap()
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('SwapFileCheck', { clear = true })

vim.api.nvim_create_autocmd('BufReadPre', {
  group = 'SwapFileCheck',
  callback = function()
    local swapname = vim.fn.swapname()
    if swapname:match('\.sw[^p]$') then
      vim.api.nvim_buf_set_option(0, 'readonly', true)
    end
  end
})

-- Optional: Modify short message to suppress swap file warnings
vim.opt.shortmess:append('A')
```

Source: vim.fandom.com

8.45 Automatically Refresh Firefox When Saving HTML/CSS

Category: autocmds
Tags: automation, web-development, integration

Automatically refresh Firefox and preserve scroll position when saving HTML, CSS, or GTPL files using MozRepl

Example: Example (Vim)

```
autocmd BufWriteCmd *.html,*.css,*.gtpl :call Refresh_firefox()
function! Refresh_firefox()
  if &modified
    write
    silent !echo 'vimYo = content.window.pageYOffset;
          \ vimXo = content.window.pageXOffset;
          \ BrowserReload();
          \ content.window.scrollTo(vimXo,vimYo);
          \ repl.quit();' |
          \ nc -w 1 localhost 4242 2>&1 > /dev/null
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWriteCmd', {
  pattern = {'*.html', '*.css', '*.gtpl'},
  callback = function()
    if vim.bo.modified then
      vim.cmd('write')
      vim.fn.system("echo 'vimYo = content.window.pageYOffset; " ..
                    "vimXo = content.window.pageXOffset; " ..
                    "BrowserReload(); " ..
                    "content.window.scrollTo(vimXo,vimYo); " ..
                    "repl.quit();' | nc -w 1 localhost 4242 2>&1 >
                    ↪  /dev/null")
    end
  end
})
```

Source: vim.fandom.com_on_Vim_save,_using_MozRepl)

8.46 Automatically Set Compiler by Filetype

Category: autocmds
Tags: compiler, filetype, automation

Automatically set the appropriate compiler for a file based on its filetype when opened

Example: Example (Vim)

```
au BufRead * try | execute "compiler ".&filetype | catch /./ | endtry
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufRead', {
  callback = function()
    pcall(function()
      vim.cmd('compiler ' .. vim.bo.filetype)
```

```
    end)
  end
})
```

Source: vim.fandom.com

8.47 Automatically Set Listing Files as Read-Only

Category: autocmds
Tags: file-protection, autocmd, read-only

Prevent accidental modifications to batch job output files by setting them read-only

Example: Example (Vim)

```
" Prevents, by default, modification of listing files
autocmd BufReadPre *.o[0-9]* set ro
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPre', {
  pattern = '*.o[0-9]*',
  callback = function()
    vim.opt_local.readonly = true
  end
})
```

Source: vim.fandom.com

8.48 Automatically Sort and Deduplicate Quickfix List

Category: autocmds
Tags: quickfix, sorting, deduplication

Automatically sort quickfix list entries by filename and line number, removing duplicate entries after commands like grep or cscope

Example: Example (Vim)

```
function! s:CompareQuickfixEntries(i1, i2)
  if bufname(a:i1.bufnr) == bufname(a:i2.bufnr)
    return a:i1.lnum == a:i2.lnum ? 0 : (a:i1.lnum < a:i2.lnum ? -1 : 1)
  else
    return bufname(a:i1.bufnr) < bufname(a:i2.bufnr) ? -1 : 1
  endif
endfunction

function! s:SortUniqQFList()
```

```
  let sortedList = sort(getqflist(), 's:CompareQuickfixEntries')
  let uniqedList = []
  let last = ''
  for item in sortedList
    let this = bufname(item.bufnr) . "\t" . item.lnum
    if this !=# last
      call add(uniqedList, item)
      let last = this
    endif
  endfor
  call setqflist(uniqedList)
endfunction

autocmd! QuickfixCmdPost * call s:SortUniqQFList()
```

Example: Example (Neovim)

```
local function compare_quickfix_entries(i1, i2)
  local buf1 = vim.fn.bufname(i1.bufnr)
  local buf2 = vim.fn.bufname(i2.bufnr)

  if buf1 == buf2 then
    return i1.lnum < i2.lnum
  else
    return buf1 < buf2
  end
end

local function sort_uniq_qf_list()
  local qf_list = vim.fn.getqflist()
  table.sort(qf_list, compare_quickfix_entries)

  local uniqed_list = {}
  local last = ''

  for _, item in ipairs(qf_list) do
    local current = vim.fn.bufname(item.bufnr) .. '\t' .. item.lnum
    if current ~= last then
      table.insert(uniqed_list, item)
      last = current
    end
  end

  vim.fn.setqflist(uniqed_list)
end

vim.api.nvim_create_autocmd('QuickfixCmdPost', {
  callback = sort_uniq_qf_list
})
```

Source: vim.fandom.com

8.49 Automatically Update Last Modified Timestamp

Category: autocmds
Tags: file-management, timestamps, autocmd

Automatically update 'Last modified' timestamp when saving a file

Example: Example (Vim)

```
function! LastModified()
  if &modified
    let save_cursor = getpos(".")
    let n = min([20, line("$")])
    keepjumps exe '1,' . n . 's#^\(.*Last modified: \).*#\1' .
          \ strftime('%a %b %d, %Y  %I:%M%p') . '#e'
    call setpos('.', save_cursor)
  endif
endfun

autocmd BufWritePre * call LastModified()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("BufWritePre", {
  callback = function()
    local modified = vim.bo.modified
    if modified then
      local save_cursor = vim.fn.getpos(".")
      local last_line = math.min(20, vim.fn.line("$"))
      vim.cmd(string.format('1,%ds#^\(.*Last modified: \).*#\1%s#e',
        last_line, os.date("%a %b %d, %Y  %I:%M%p")))
      vim.fn.setpos(".", save_cursor)
    end
  end
})
```

Source: vim.fandom.com

8.50 Change Color Scheme for Different Directories

Category: autocmds
Tags: color-scheme, directory-specific, workflow

Dynamically change color scheme based on the current file's directory to provide visual context

Example: Example (Vim)

```
autocmd BufEnter * if match(@%,'/otherdir/')>=0 | colorscheme oceanblack |
↪  else | colorscheme inkpot | end
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("BufEnter", {
  callback = function()
    local current_file = vim.fn.expand('%:p')
    if current_file:match('/otherdir/') then
      vim.cmd('colorscheme oceanblack')
    else
      vim.cmd('colorscheme inkpot')
    end
  end
})
```

Source: vim.fandom.com

8.51 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
    pattern = '*',
    command = 'cd %:p:h',
    desc = 'Always work in current file directory'
})

vim.opt.autochdir = true
```

Source: Community contributed

8.52 Check File Changes on Focus

Category: autocmds
Tags: file-monitoring, events, autocommands

Automatically check and reload files when Vim regains focus or enters a buffer, ensuring you always have the latest version

Example: Example (Vim)

```
au FocusGained,BufEnter * :checktime
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'FocusGained', 'BufEnter'}, {
  pattern = '*',
  callback = function()
    vim.cmd('checktime')
  end
})
```

Source: vim.fandom.com

8.53 Colorscheme Cycling Autocommand

Category: autocmds
Tags: colorscheme, auto-command, window-event

Automatically cycle through color schemes when entering a new window, creating a dynamic visual effect

Example: Example (Vim)

```
:au WinEnter * :silent CN
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('WinEnter', {
  callback = function()
    vim.cmd('silent CN')
  end
})
```

Source: vim.fandom.com

8.54 Colorscheme for Focused Vim Instance

Category: autocmds
Tags: color-scheme, focus, multi-window

Change color scheme to highlight the currently focused Vim instance

Example: Example (Vim)

```
autocmd FocusLost * :colorscheme desert
autocmd FocusGained * :colorscheme default
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("FocusLost", {
  callback = function()
    vim.cmd('colorscheme desert')
  end
})

vim.api.nvim_create_autocmd("FocusGained", {
  callback = function()
    vim.cmd('colorscheme default')
  end
})
```

Source: vim.fandom.com

8.55 Conditional File Encoding Auto-Command

Category: autocmds
Tags: file-encoding, buffer-settings, conditional-config

Safely set file encoding for specific file types while handling non-modifiable buffers

Example: Example (Vim)

```
function! Noma_workaround()
  if &modifiable
    set tw=77 ai nocindent fileencoding=iso-8859-1
  else
    set tw=77 ai nocindent
  endif
endfunction
au BufNewFile,BufRead mutt*,*.txt,*/.followup call Noma_workaround()
```

Example: Example (Neovim)

```
local function noma_workaround()
  if vim.opt.modifiable:get() then
    vim.opt.textwidth = 77
    vim.opt.autoindent = true
    vim.opt.cindent = false
    vim.opt.fileencoding = 'iso-8859-1'
  else
    vim.opt.textwidth = 77
    vim.opt.autoindent = true
    vim.opt.cindent = false
```

```
  end
end

vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = {'mutt*', '*.txt', '*/.followup'},
  callback = noma_workaround
})
```

Source: vim.fandom.com

8.56 Conditional Fold Handling for File Types

Category: autocmds
Tags: folding, file-type, autocmd

Automatically set fold method and open folds for specific file types while maintaining fold control

Example: Example (Vim)

```
autocmd Syntax c,cpp,vim,xml,html,xhtml setlocal foldmethod=syntax
autocmd Syntax c,cpp,vim,xml,html,xhtml,perl normal zR
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('CustomFolding', { clear = true })

vim.api.nvim_create_autocmd('Syntax', {
  group = 'CustomFolding',
  pattern = {'c', 'cpp', 'vim', 'xml', 'html', 'xhtml', 'perl'},
  callback = function()
    vim.opt_local.foldmethod = 'syntax'
    vim.cmd('normal zR')
  end
})
```

Source: vim.fandom.com

8.57 Context-Specific Menus for Different Filetypes

Category: autocmds
Tags: buffer-management, filetype, menu-customization

Dynamically show and hide menus based on the current buffer's filetype, allowing context-specific menu interactions

Example: Example (Vim)

```
au BufEnter * if exists('b:BuffEnter')|exec b:BuffEnter|endif
au BufLeave * if exists('b:BuffEnter')|exec b:BuffLeave|endif

" Example for a specific filetype
let b:BuffEnter='amenu C.added ...'
let b:BuffLeave='unmenu! C|unmenu C'
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufEnter', 'BufLeave'}, {
  callback = function(event)
    local buffer_enter = vim.b[event.buf].BuffEnter
    local buffer_leave = vim.b[event.buf].BuffLeave

    if event.event == 'BufEnter' and buffer_enter then
      vim.cmd(buffer_enter)
    elseif event.event == 'BufLeave' and buffer_leave then
      vim.cmd(buffer_leave)
    end
  end
})
```

Source: vim.fandom.com

8.58 Create directory on save

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePre', {
    pattern = '*',
    callback = function(args)
        local dir = vim.fn.fnamemodify(args.file, ':p:h')
        if vim.fn.isdirectory(dir) == 0 then
            vim.fn.mkdir(dir, 'p')
        end
    end,
    desc = 'Creates parent directories if they don\'t exist'
```

```
})
```

Source: Community contributed

8.59 Cursor Line Only in Active Window

Category: autocmds
Tags: window-management, ui, autocmd

Highlight the cursor line only in the active window to reduce visual clutter

Example: Example (Vim)

```
augroup CursorLine
  au!
  au VimEnter,WinEnter,BufWinEnter * setlocal cursorline
  au WinLeave * setlocal nocursorline
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('CursorLine', { clear = true })
vim.api.nvim_create_autocmd({'VimEnter', 'WinEnter', 'BufWinEnter'}, {
  group = 'CursorLine',
  callback = function()
    vim.wo.cursorline = true
  end
})
vim.api.nvim_create_autocmd('WinLeave', {
  group = 'CursorLine',
  callback = function()
    vim.wo.cursorline = false
  end
})
```

Source: vim.fandom.com

8.60 Debug Window Autocommands Interactively

Category: autocmds
Tags: debugging, autocommands, troubleshooting

Interactively debug autocommands triggered by window actions like closing or loading buffers

Example: Example (Vim)

```
:debug quit
:debug edit <file>
```

Example: Example (Neovim)

```
-- For Neovim, use similar Ex commands with interactive debugging
-- Use :debug to step through autocommands when triggering specific actions
```

Source: vim.fandom.com

8.61 Detect New Window Creation Reliably

Category: autocmds
Tags: window-management, autocmd, initialization

Provides a robust method to detect and initialize new windows, avoiding common pitfalls with WinEnter autocmd

Example: Example (Vim)

```
" Set window-local created flag
autocmd VimEnter * autocmd WinEnter * let w:created=1

" Initialize options only for new windows
autocmd WinEnter * if !exists('w:created') | setlocal nu | endif
```

Example: Example (Neovim)

```
-- Set window-local created flag
vim.api.nvim_create_autocmd('VimEnter', {
  callback = function()
    vim.api.nvim_create_autocmd('WinEnter', {
      callback = function()
        vim.w.created = true
      end
    })
  end
})

-- Initialize options only for new windows
vim.api.nvim_create_autocmd('WinEnter', {
  callback = function()
    if vim.w.created == nil then
      vim.wo.number = true
    end
  end
})
```

Source: vim.fandom.com

8.62 Detect New Window Creation with Window-Local Variable

Category: autocmds
Tags: window-management, autocmd, initialization

Reliably detect when a new window is created using a window-local variable, allowing custom initialization for new windows

Example: Example (Vim)

```
" Set window-local variable when a new window is created
autocmd VimEnter * autocmd WinEnter * let w:created=1

" Initialize window-specific options only for new windows
autocmd WinEnter * if !exists('w:created') | setlocal nu | endif
```

Example: Example (Neovim)

```
-- Set up autocommand to mark new windows
vim.api.nvim_create_autocmd('VimEnter', {
  callback = function()
    vim.api.nvim_create_autocmd('WinEnter', {
      callback = function()
        vim.w.created = true
      end
    })
  end
})

-- Initialize options only for new windows
vim.api.nvim_create_autocmd('WinEnter', {
  callback = function()
    if vim.w.created == nil then
      vim.wo.number = true
    end
  end
})
```

Source: vim.fandom.com

8.63 Detect Perl Batch Files in Vim

Category: autocmds
Tags: filetype, detection, scripting

Automatically detect and set filetype for Perl batch files based on first line content

Example: Example (Vim)

```
au BufRead,BufNewFile *.bat         if getline(1) =~ '--*-Perl-*--' | setf
 ↪  perl | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = '*.bat',
  callback = function()
    local first_line = vim.fn.getline(1)
    if first_line:match('--*-Perl-*--') then
      vim.bo.filetype = 'perl'
    end
  end
})
```

Source: vim.fandom.com

8.64 Detect and Mark Buffer as Modified When File Deleted

Category: autocmds
Tags: file-handling, buffer-management, error-handling

Automatically mark a buffer as modified when its underlying file is deleted, preventing accidental buffer abandonment

Example: Example (Vim)

```
au FileChangedShell * call FCSHandler(expand("<afile>:p"))
function FCSHandler(name)
  if v:fcs_reason == "deleted"
    call setbufvar(expand(a:name), '&modified', '1')
    echohl WarningMsg
    echomsg 'File "'.a:name.'" no longer available - ''modified'' set'
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileChangedShell', {
  callback = function(event)
    local filename = event.file
    if vim.v.fcs_reason == 'deleted' then
      vim.api.nvim_buf_set_option(0, 'modified', true)
      vim.notify('File "' .. filename .. '" no longer available - modified
       ↪  set', vim.log.levels.WARN)
    end
  end
})
```

Source: vim.fandom.com

8.65 Dynamic Color Scheme Based on Buffer/Directory

Category: autocmds
Tags: color-scheme, buffer-management, workflow

Automatically change color scheme to visually distinguish different files, directories, or Vim instances

Example: Example (Vim)

```
au BufEnter * if (exists("b:colors_name")) | let
↪  b:current_colors=colors_name
 | execute "colorscheme " . b:colors_name | endif
au BufLeave * if (exists("b:current_colors")) | execute "colorscheme " .
↪  b:current_colors | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("BufEnter", {
  callback = function()
    if vim.b.colors_name then
      vim.g.current_colors = vim.g.colors_name
      vim.cmd('colorscheme ' .. vim.b.colors_name)
    end
  end
})

vim.api.nvim_create_autocmd("BufLeave", {
  callback = function()
    if vim.g.current_colors then
      vim.cmd('colorscheme ' .. vim.g.current_colors)
    end
  end
})
```

Source: vim.fandom.com

8.66 Dynamic Color Schemes Based on Context

Category: autocmds
Tags: color-scheme, buffer-management, workflow

Automatically change color schemes to visually distinguish between different files, directories, or Vim instances, helping prevent accidental edits

Example: Example (Vim)

```
au BufEnter * if (exists("b:colors_name")) | let
 ↪  b:current_colors=colors_name
  \ | execute "colorscheme " . b:colors_name | endif
au BufLeave * if (exists("b:current_colors")) | execute "colorscheme " .
 ↪  b:current_colors | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("BufEnter", {
  callback = function()
    if vim.b.colors_name then
      vim.g.current_colors = vim.g.colors_name
      vim.cmd('colorscheme ' .. vim.b.colors_name)
    end
  end
})

vim.api.nvim_create_autocmd("BufLeave", {
  callback = function()
    if vim.g.current_colors then
      vim.cmd('colorscheme ' .. vim.g.current_colors)
    end
  end
})
```

Source: vim.fandom.com

8.67 Dynamic Folding with Indent and Manual Override

Category: autocmds
Tags: folding, buffer-management, configuration

Create a flexible folding configuration that uses indent-based folding by default but allows manual fold creation during editing

Example: Example (Vim)

```
augroup vimrc
  au BufReadPre * setlocal foldmethod=indent
  au BufWinEnter * if &fdm == 'indent' | setlocal foldmethod=manual | endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('vimrc', { clear = true })
vim.api.nvim_create_autocmd({'BufReadPre'}, {
  group = 'vimrc',
  pattern = '*',
  callback = function() vim.opt_local.foldmethod = 'indent' end
```

```
})
vim.api.nvim_create_autocmd({'BufWinEnter'}, {
  group = 'vimrc',
  pattern = '*',
  callback = function()
    if vim.o.foldmethod == 'indent' then
      vim.opt_local.foldmethod = 'manual'
    end
  end
})
```

Source: vim.fandom.com

8.68 Dynamic Language-Specific Commenting Functions

Category: autocmds
Tags: language-support, file-type, key-mapping

Automatically set up comment mappings based on the current file type, providing a consistent commenting experience across languages

Example: Example (Vim)

```
function! PoundComment()
  map - :s/^/# /<CR>
  map _ :s/^\s*# \=//<CR>
  set comments=:#
endfunction

autocmd FileType perl,cgi,csh,sh call PoundComment()
autocmd FileType java call SlashComment()
```

Example: Example (Neovim)

```
local function setup_comment_mapping(comment_char)
  vim.keymap.set('n', '-', function()
    vim.api.nvim_command(string.format('s/^/%s /', comment_char))
  end)
  vim.keymap.set('n', '_', function()
    vim.api.nvim_command(string.format('s/^\s*%s \?//', comment_char))
  end)
end

vim.api.nvim_create_autocmd('FileType', {
  pattern = {'perl', 'cgi', 'csh', 'sh'},
  callback = function() setup_comment_mapping('#') end
})

vim.api.nvim_create_autocmd('FileType', {
  pattern = {'java', 'cpp'},
  callback = function() setup_comment_mapping('//') end
```

```
})
```

Source: vim.fandom.comcommenting_out_of_source_code)

8.69 Dynamically Adjust Screen Brightness in Vim

Category: autocmds
Tags: focus-events, macos, display

Automatically adjust screen brightness when Vim gains or loses focus, useful for reducing eye strain and improving visibility

Example: Example (Vim)

```
:au FocusGained * :!brightness 1
:au FocusLost * :!brightness 0.5
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('BrightnessAdjust', { clear = true })

vim.api.nvim_create_autocmd('FocusGained', {
  group = 'BrightnessAdjust',
  callback = function()
    vim.fn.system('brightness 1')
  end
})

vim.api.nvim_create_autocmd('FocusLost', {
  group = 'BrightnessAdjust',
  callback = function()
    vim.fn.system('brightness 0.5')
  end
})
```

Source: vim.fandom.com

8.70 Filetype-Specific Commenting Function

Category: autocmds
Tags: commenting, filetype, autocmd

A function that automatically sets up commenting mappings for different filetypes when a buffer is entered

Example: Example (Vim)

```
function CommentIt()
  if &filetype == "vim"
    vmap +# :s/^/"/<CR>
    vmap -# :s/^"//
  elseif &filetype == "tcl"
    vmap +# :s/^/#/<CR>
    vmap -# :s/^#//
  elseif &filetype == "c"
    vmap +# I/*<Esc>gv<End><Esc>a*/<Esc>
    vmap -# I<Esc>2xgv$<Esc>h2x<Esc>
  endif
endfunction

autocmd BufEnter * call CommentIt()
```

Example: Example (Neovim)

```
local function comment_it()
  local ft = vim.bo.filetype
  if ft == "vim" then
    vim.keymap.set('v', '+#', ':s/^/"/<CR>', { buffer = true })
    vim.keymap.set('v', '-#', ':s/^"///<CR>', { buffer = true })
  elseif ft == "tcl" then
    vim.keymap.set('v', '+#', ':s/^/#/<CR>', { buffer = true })
    vim.keymap.set('v', '-#', ':s/^#///<CR>', { buffer = true })
  elseif ft == "c" then
    vim.keymap.set('v', '+#', 'I/*<Esc>gv<End><Esc>a*/<Esc>', { buffer =
    ↪  true })
    vim.keymap.set('v', '-#', 'I<Esc>2xgv$<Esc>h2x<Esc>', { buffer = true })
  end
end

vim.api.nvim_create_autocmd('BufEnter', {
  callback = comment_it
})
```

Source: vim.fandom.com

8.71 Filetype-Specific Dictionary Completion

Category: autocmds
Tags: filetype, completion, customization

Automatically load filetype-specific dictionaries to provide context-aware word completion

Example: Example (Vim)

```
" Automatically load filetype-specific dictionary
au FileType * execute 'setlocal dict+=~/.vim/words/'.&filetype.'.txt'
```

Example: Example (Neovim)

```
-- Automatically load filetype-specific dictionary
vim.api.nvim_create_autocmd('FileType', {
  pattern = '*',
  callback = function()
    local dict_path = string.format('~/.vim/words/%s.txt', vim.bo.filetype)
    vim.opt_local.dictionary:append(dict_path)
  end
})
```

Source: vim.fandom.com

8.72 Filetype-Specific Periodic Saving

Category: autocmds
Tags: file-type, auto-save, notes

Create a custom auto-save for specific file types like notes, enabling periodic saving only for certain buffers

Example: Example (Vim)

```
au CursorHold <buffer> update
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('CursorHold', {
  buffer = 0,  -- current buffer
  command = 'update'
})
```

Source: vim.fandom.com

8.73 GPG File Encryption Autocommands

Category: autocmds
Tags: encryption, gpg, security

Automatically handle GPG-encrypted files with transparent editing and encryption

Example: Example (Vim)

```
augroup encrypted
  au!
  autocmd BufReadPre,FileReadPre *.gpg set viminfo=
  autocmd BufReadPre,FileReadPre *.gpg set noswapfile noundofile nobackup
  autocmd BufReadPost,FileReadPost *.gpg '[,']!gpg --decrypt 2> /dev/null
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('encrypted', { clear = true })
vim.api.nvim_create_autocmd({'BufReadPre', 'FileReadPre'}, {
  group = 'encrypted',
  pattern = '*.gpg',
  callback = function()
    vim.opt.viminfo = ''
    vim.opt.swapfile = false
    vim.opt.undofile = false
    vim.opt.backup = false
  end
})
```

Source: vim.fandom.com

8.74 Handle File Change Notifications Safely

Category: autocmds
Tags: file-handling, error-management, autocmd

Safely handle file change events and ensure proper cleanup of autocmds after file modification

Example: Example (Vim)

```
checktime
exe "au FileChangedShell " . expand("%") . " let &cpo = &cpo"
" Do stuff to the file.
checktime
exe "au! FileChangedShell " . expand("%")
```

Example: Example (Neovim)

```
vim.cmd('checktime')
vim.cmd(string.format('au FileChangedShell %s let &cpo = &cpo',
↪  vim.fn.expand('%')))
-- Do stuff to the file
vim.cmd('checktime')
vim.cmd(string.format('au! FileChangedShell %s', vim.fn.expand('%')))
```

Source: vim.fandom.com

8.75 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({ 'FileType' }, {
    pattern = 'python',
    callback = function()
        vim.bo.colorcolumn = 88
    end,
})

vim.api.nvim_create_autocmd({ 'FileType' }, {
    pattern = { 'javascript', 'typescript' },
    callback = function()
        vim.bo.colorcolumn = 100
    end,
})

vim.api.nvim_create_autocmd({ 'FileType' }, {
    pattern = 'gitcommit',
    callback = function()
        vim.bo.colorcolumn = 72
    end,
})
```

Source: Community contributed

8.76 Highlight yanked text

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

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

Example: Example (Vim)

```
: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()})
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('TextYankPost', {
  pattern = '*',
  callback = function()
    vim.highlight.on_yank()
  end,
})

vim.api.nvim_create_autocmd('TextYankPost', {
  pattern = '*',
  callback = function()
    vim.highlight.on_yank({
      higroup = 'Search',
      timeout = 150
    })
  end
})
```

Source: Community contributed

8.77 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  pattern = '*',
  callback = function()
    if vim.fn.line("'\"") > 0 and vim.fn.line("'\"") <= vim.fn.line("$")
    ↪  then
      vim.cmd('normal! g`"')
    end
  end
})
```

Source: Community contributed

8.78 Language-Specific Tab Settings

Category: autocmds
Tags: file-type, configuration, indentation

Automatically set different tab settings for specific file types like Python

Example: Example (Vim)

```
autocmd FileType python set tabstop=4 shiftwidth=4 expandtab
autocmd FileType make setlocal noexpandtab
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.opt_local.tabstop = 4
    vim.opt_local.shiftwidth = 4
    vim.opt_local.expandtab = true
  end
})

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'make',
  callback = function()
    vim.opt_local.expandtab = false
  end
})
```

Source: vim.fandom.com

8.79 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: Example (Neovim)

```
-- 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,
})
```

Source: Community contributed

8.80 MoinMoin File Type Detection

Category: autocmds
Tags: file-type, detection

Automatically detect and set filetype for MoinMoin wiki files with .moin extension

Example: Example (Vim)

```
au! BufNewFile,BufRead *.moin setf moin
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.moin',
  command = 'setf moin'
})
```

Source: vim.fandom.com

8.81 Periodic Background Command Execution

Category: autocmds
Tags: timer, background-task, event-handling

Simulate a timer in Vim versions before 8.0 by using CursorHold autocommand to periodically execute a function

Example: Example (Vim)

```
autocmd CursorHold * call Timer()
function! Timer()
  call feedkeys("f\e")
  " Periodically trigger a function
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('CursorHold', {
  callback = function()
    -- Simulate periodic timer execution
    vim.api.nvim_feedkeys(vim.api.nvim_replace_termcodes('f', true, false,
    ↪  true), 'n', true)
  end
})
```

Source: vim.fandom.com

8.82 Persistent Command Window Line Position

Category: autocmds
Tags: command-line, workflow, automation

Create an autocmd to remember and restore the last line position in the command window when executing commands

Example: Example (Vim)

```
" Autocmd to remember command window line position
autocmd CmdwinEnter * nnoremap <buffer> <F5> :let
↪  g:CmdWindowLineMark=line(".")<CR><CR>q::execute "normal
↪  ".g:CmdWindowLineMark."G"<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for remembering command window line position
vim.api.nvim_create_autocmd('CmdwinEnter', {
  callback = function()
    vim.keymap.set('n', '<F5>', function()
```

```
      vim.g.CmdWindowLineMark = vim.fn.line('.')
      vim.cmd('normal! <CR>q:')
      vim.cmd('normal! ' .. vim.g.CmdWindowLineMark .. 'G')
    end, { buffer = true })
  end
})
```

Source: vim.fandom.com

8.83 Persistent Doxygen Comment Configuration

Category: autocmds
Tags: file-type, comments, configuration

Use autocmd to set Doxygen comment behavior for C/C++ files, preventing syntax file overrides

Example: Example (Vim)

```
autocmd FileType c,cpp set comments-=://
autocmd FileType c,cpp set comments+=:///
autocmd FileType c,cpp set comments+=://
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'c', 'cpp'},
  callback = function()
    vim.opt.comments:remove(':/')
    vim.opt.comments:append('://!')
    vim.opt.comments:append('://')
  end
})
```

Source: vim.fandom.com

8.84 Remove Trailing Whitespace on Save

Category: autocmds
Tags: whitespace, cleanup, file-saving

Automatically remove trailing whitespace when saving files, with optional file type filtering

Example: Example (Vim)

```
autocmd BufWritePre *.{cpp,h,c} %s/\s\+$//e

" More selective version
autocmd FileType c,cpp,java,php autocmd BufWritePre <buffer> %s/\s\+$//e
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePre', {
  pattern = {'*.{cpp,h,c}'},
  callback = function()
    local view = vim.fn.winsaveview()
    vim.cmd([[%s/\s\+$//e]])
    vim.fn.winrestview(view)
  end
})

-- With filetype filtering
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'c', 'cpp', 'java', 'php'},
  callback = function()
    vim.api.nvim_create_autocmd('BufWritePre', {
      buffer = true,
      callback = function()
        local view = vim.fn.winsaveview()
        vim.cmd([[%s/\s\+$//e]])
        vim.fn.winrestview(view)
      end
    })
  end
})
```

Source: vim.fandom.com

8.85 Restore Cursor to Specific Line on File Open

Category: autocmds
Tags: cursor-position, file-editing

Automatically position cursor at a specific line when opening a file, useful for consistent editing workflow

Example: Example (Vim)

```
au BufWinEnter * exe 'normal gg'  " Go to first line
au BufWinEnter * exe 'normal 6gg'  " Go to 6th line
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWinEnter', {
  callback = function()
    -- Go to first line
    vim.cmd('normal! gg')

    -- Or go to a specific line (e.g., 6th line)
    -- vim.cmd('normal! 6gg')
  end
})
```

Source: vim.fandom.com

8.86 Restore Mode When Switching Tab Pages

Category: autocmds
Tags: tab-management, mode-persistence, workflow

Automatically restore the previous mode (insert/normal) when switching between tab pages, treating each tab as a separate workspace

Example: Example (Vim)

```
augroup tabInsertMode
  au!
  au TabLeave * let t:tabInsertMode_lastmode = mode()
        \ | let t:tabInsertMode_lastcol = col('.')
        \ | if t:tabInsertMode_lastcol == col('$')
        \ |   let t:tabInsertMode_lastmode = 'A'
        \ | endif
  au TabEnter * if exists('t:tabInsertMode_lastmode')
        \ |   if t:tabInsertMode_lastmode == 'A'
        \ |     if mode() == 'i' | call feedkeys("\<C-O>l")
        \ |     else | call feedkeys("A") | endif
        \ |   elseif t:tabInsertMode_lastmode == 'i'
        \ |     if mode() != 'i' | call feedkeys("i") | endif
        \ |   else
        \ |     if mode() == 'i' | call feedkeys("\<Esc>") | endif
        \ |   endif
        \ | endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('tabInsertMode', { clear = true })

vim.api.nvim_create_autocmd('TabLeave', {
  group = 'tabInsertMode',
  callback = function()
    vim.t.tabInsertMode_lastmode = vim.fn.mode()
    vim.t.tabInsertMode_lastcol = vim.fn.col('.')
```

```
    if vim.t.tabInsertMode_lastcol == vim.fn.col('$') then
      vim.t.tabInsertMode_lastmode = 'A'
    end
  end
})

vim.api.nvim_create_autocmd('TabEnter', {
  group = 'tabInsertMode',
  callback = function()
    if vim.t.tabInsertMode_lastmode then
      if vim.t.tabInsertMode_lastmode == 'A' then
        if vim.fn.mode() == 'i' then
          vim.api.nvim_feedkeys(vim.api.nvim_replace_termcodes('<C-O>l',
          ↪  true, false, true), 'n', false)
        else
          vim.api.nvim_feedkeys('A', 'n', false)
        end
      elseif vim.t.tabInsertMode_lastmode == 'i' then
        if vim.fn.mode() ~= 'i' then
          vim.api.nvim_feedkeys('i', 'n', false)
        end
      else
        if vim.fn.mode() == 'i' then
          vim.api.nvim_feedkeys('<Esc>', 'n', false)
        end
      end
    end
  end
})
```

Source: vim.fandom.com

8.87 Set Default Filetype for New Buffers

Category: autocmds
Tags: filetype, buffer-management, startup

Automatically set a default filetype (e.g., Python) for new blank buffers

Example: Example (Vim)

```
let g:do_filetype = 0
au BufWinEnter,BufAdd * if expand('<afile>') == "" | let g:do_filetype = 1 |
↪  endif
au BufEnter * if g:do_filetype | setf python | let g:do_filetype = 0 | endif
```

Example: Example (Neovim)

```
vim.g.do_filetype = 0

vim.api.nvim_create_augroup('DefaultFiletype', { clear = true })
```

```
vim.api.nvim_create_autocmd({'BufWinEnter', 'BufAdd'}, {
  group = 'DefaultFiletype',
  callback = function()
    if vim.fn.expand('<afile>') == '' then
      vim.g.do_filetype = 1
    end
  end
})

vim.api.nvim_create_autocmd('BufEnter', {
  group = 'DefaultFiletype',
  callback = function()
    if vim.g.do_filetype == 1 then
      vim.bo.filetype = 'python'
      vim.g.do_filetype = 0
    end
  end
})
```

Source: vim.fandom.com

8.88 Set Fortran Source Form Manually

Category: autocmds
Tags: fortran, syntax-highlighting, file-type

Manually set Fortran source form for .for files to fix syntax highlighting issues

Example: Example (Vim)

```
au BufRead,BufNewFile *.for let b:fortran_fixed_source=1
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = '*.for',
  callback = function()
    vim.b.fortran_fixed_source = 1
  end
})
```

Source: vim.fandom.com

8.89 Set Syntax for Files in Specific Directory

Category: autocmds
Tags: autocmd, syntax-highlighting, file-type

Automatically set syntax highlighting for files in a specific directory path, useful for handling files with non-standard extensions or special content types

Example: Example (Vim)

```
autocmd BufRead,BufNewFile ~/.pyblosxom/data/* set syntax=html
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = vim.fn.expand('~/.pyblosxom/data/*'),
  callback = function()
    vim.bo.syntax = 'html'
  end
})
```

Source: vim.fandom.com

8.90 Set UTF-8 Encoding for Mac Quicklook

Category: autocmds
Tags: encoding, mac-os, file-operations

Automatically set UTF-8 extended attributes for files to ensure correct display in Mac Quicklook

Example: Example (Vim)

```
au BufWritePost * :if &fenc=='utf-8' || (&fenc=='' && &enc=='utf-8') | exec
↪  "silent !xattr -w com.apple.TextEncoding 'UTF-8;134217984' '%'" | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
  callback = function()
    if vim.o.fileencoding == 'utf-8' or (vim.o.fileencoding == '' and
    ↪  vim.o.encoding == 'utf-8') then
      vim.fn.system('xattr -w com.apple.TextEncoding "UTF-8;134217984" "' ..
      ↪  vim.fn.expand('%') .. '"')
    end
  end
})
```

Source: vim.fandom.com

8.91 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({ 'BufRead', 'BufNewFile' }, {
    pattern = '*.conf',
    callback = function()
        vim.bo.filetype = 'conf'
    end,
})

vim.api.nvim_create_autocmd({ 'BufRead', 'BufNewFile' }, {
    pattern = 'Jenkinsfile',
    callback = function()
        vim.bo.filetype = 'groovy'
    end,
})

vim.api.nvim_create_autocmd({ 'BufRead' }, {
    pattern = '*',
    callback = function()
        local firt_line = vim.fn.getline(1)

        if first_line:match('^#!/usr/bin/env python') then
            vim.bo.filetype = 'python'
        end
    end,
})
```

Source: Community contributed

8.92 Smart View Saving with Filters

Category: autocmds
Tags: views, file-management, buffer-filtering

Implement advanced filtering for view saving to avoid creating views for temporary, generated, or system files

Example: Example (Vim)

```
let g:skipview_files = [
            \ '[EXAMPLE PLUGIN BUFFER]'
            \ ]
function! MakeViewCheck()
    if has('quickfix') && &buftype =~ 'nofile'
        return 0
    endif
    if empty(glob(expand('%:p')))
        return 0
    endif
    if index(g:skipview_files, expand('%')) >= 0
        return 0
    endif
    return 1
endfunction
```

Example: Example (Neovim)

```
local M = {}

M.skipview_files = {
  '[EXAMPLE PLUGIN BUFFER]'
}

function M.make_view_check()
  -- Skip quickfix and nofile buffers
  if vim.wo.buftype:match('nofile') then
    return false
  end

  -- Skip non-existent files
  local filename = vim.fn.expand('%:p')
  if vim.fn.empty(vim.fn.glob(filename)) == 1 then
    return false
  end

  -- Skip files in skip list
  local current_file = vim.fn.expand('%')
  for _, skip_file in ipairs(M.skipview_files) do
    if current_file == skip_file then
      return false
    end
  end

  return true
end

return M
```

Source: vim.fandom.com

8.93 Speed Up Searches by Disabling Autocmds

Category: autocmds
Tags: performance, search, autocmd

Use noautocmd to significantly speed up vimgrep searches by temporarily disabling autocommands

Example: Example (Vim)

```
:noautocmd vimgrep /{pattern}/[flags] {file(s)}
```

Example: Example (Neovim)

```
vim.cmd('noautocmd vimgrep /{pattern}/[flags] {file(s)}')
```

Source: vim.fandom.com

8.94 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
    pattern = { 'markdown', 'text', 'gitcommit' },
    callback = function()
        vim.bo.spell = true
        vim.bo.spelllang = 'en_us'
    end,
    desc = 'Enable spell check for text files'
})

vim.api.nvim_create_autocmd('FileType', {
    pattern = 'help'
    callback = function()
        vim.bo.spell = false
    end,
    desc = 'Disable spell check for help'
```

```
})
```

Source: Community contributed

8.95 Template insertion for new files

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufNewFile', {
    pattern = '*.html',
    callback = function()
        local template = vim.fn.stdpath('config') ..
         ↪  '/templates/html_template.html'
        if vim.fn.filereadable(template) == 1 then
            vim.cmd('0read ' .. vim.fn.fnameescape(template))
        end
    end
})

vim.api.nvim_create_autocmd('BufNewFile', {
    pattern = '*.py',
    callback = function()
        local template = vim.fn.stdpath('config') ..
         ↪  '/templates/python_template.html'
        if vim.fn.filereadable(template) == 1 then
            vim.cmd('0read ' .. vim.fn.fnameescape(template))
        end
    end
})

vim.api.nvim_create_autocmd('BufNewFile', {
  pattern = '*.sh',
  callback = function()
    local lines = { '#!/bin/bash', '' }
    vim.api.nvim_buf_set_lines(0, 0, 0, false, lines)
    vim.api.nvim_win_set_cursor(0, {2, 0})
  end
})
```

Source: Community contributed

8.96 Throttle Frequent Autocommands

Category: autocmds
Tags: performance, autocommands, cursor-movement

Prevent performance slowdowns by limiting how often autocommands are triggered during frequent events like cursor movement

Example: Example (Vim)

```
function! LongEnough(timer, delay, ...)
  let result = 0
  let suppressionCount = a:0 ? a:1 : 0

  if (!exists(a:timer))
    let result = 1
  else
    let timeElapsed = localtime() - {a:timer}
    if (timeElapsed >= a:delay)
      let result = 1
    elseif (suppressionCount > 0)
      let {a:timer}_callCount += 1
      if ({a:timer}_callCount >= suppressionCount)
        let result = 1
      endif
    endif
  endif

  if (result)
    let {a:timer} = localtime()
    let {a:timer}_callCount = 0
  endif
  return result
endfunction

" Example usage
autocommand CursorMoved * if LongEnough("g:MatchParens", 1, 3) | call
 ↪  MatchParens() | endif
```

Example: Example (Neovim)

```
local function long_enough(timer, delay, suppression_count)
  suppression_count = suppression_count or 0
  local timer_var = vim.g[timer]
  local call_count_var = vim.g[timer .. '_callCount'] or 0

  if not timer_var then
    return true
  end

  local time_elapsed = os.time() - timer_var
  if time_elapsed >= delay then
    return true
```

```
  end

  if suppression_count > 0 then
    call_count_var = call_count_var + 1
    if call_count_var >= suppression_count then
      return true
    end
  end

  return false
end

-- Example usage
vim.api.nvim_create_autocmd('CursorMoved', {
  callback = function()
    if long_enough('MatchParens', 1, 3) then
      -- Call your function here
    end
  end
})
```

Source: vim.fandom.com

8.97 Track Cursor Position in Insert Mode

Category: autocmds
Tags: cursor-tracking, insert-mode, autocmd

Tracks and preserves cursor column position when exiting insert mode

Example: Example (Vim)

```
let CursorColumnI = 0
autocmd InsertEnter * let CursorColumnI = col('.')
autocmd CursorMovedI * let CursorColumnI = col('.')
autocmd InsertLeave * if col('.') != CursorColumnI | call cursor(0,
↪  col('.')+1) | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('CursorPositionTracking', { clear = true })
vim.api.nvim_create_autocmd('InsertEnter', {
  group = 'CursorPositionTracking',
  callback = function()
    vim.b.cursor_column_i = vim.fn.col('.')
  end
})
vim.api.nvim_create_autocmd('CursorMovedI', {
  group = 'CursorPositionTracking',
  callback = function()
    vim.b.cursor_column_i = vim.fn.col('.')
```

```
  end
})
vim.api.nvim_create_autocmd('InsertLeave', {
  group = 'CursorPositionTracking',
  callback = function()
    if vim.fn.col('.') ~= vim.b.cursor_column_i then
      vim.fn.cursor(0, vim.fn.col('.') + 1)
    end
  end
})
```

Source: vim.fandom.com

CHAPTER 9

Buffer management

9.1 Advanced CSV File Navigation and Manipulation

Category: buffer_management
Tags: csv, data-processing, text-manipulation

Powerful functions for working with CSV files, including column highlighting, searching within columns, and sorting by specific columns

Example: Example (Vim)

```
" Highlight a specific column in CSV
function! CSVH(colnr)
  if a:colnr > 1
    let n = a:colnr - 1
    execute 'match Keyword /^([^,]*,)\{'.n.'}\zs[^,]*/'
    execute 'normal! 0'.n.'f,'
  elseif a:colnr == 1
    match Keyword /^[^,]*/
    normal! 0
  else
    match
  endif
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
```

Example: Example (Neovim)

```
-- Highlight a specific column in CSV
local function csv_highlight(colnr)
  if colnr > 1 then
    local n = colnr - 1
    vim.cmd('match Keyword /^([^,]*,){'..n..'}\\zs[^,]*/')
    vim.cmd('normal! 0'..n..'f,')
  elseif colnr == 1 then
    vim.cmd('match Keyword /^[^,]*/')
    vim.cmd('normal! 0')
  else
    vim.cmd('match')
  end
end
```

```
vim.api.nvim_create_user_command('Csv', function(opts)
  csv_highlight(tonumber(opts.args))
end, { nargs = 1 })
```

Source: vim.fandom.com

9.2 Advanced File Reload with Cursor Position

Category: buffer_management
Tags: buffer, reload, cursor-position

A more sophisticated file reload function that preserves cursor position and handles file reloading gracefully

Example: Example (Vim)

```
function! ReloadFile(is_force_pos)
  let position = getpos(".")
  %d
  r
  1d
  if a:is_force_pos
    call setpos(".", position)
  endif
endfunc
```

Example: Example (Neovim)

```
function _G.reload_file(force_pos)
  local pos = vim.fn.getpos('.')
  vim.api.nvim_command(':%delete')
  vim.cmd('edit!')
  if force_pos then
    vim.fn.setpos('.', pos)
  end
end

vim.api.nvim_create_user_command('Reload', function()
  _G.reload_file(true)
end, {})
```

Source: vim.fandom.com

9.3 Advanced Hex Editing in Vim/Neovim

Category: buffer_management
Tags: hex-editing, binary-files, file-manipulation

Provides a robust method to toggle hex mode for editing binary files, with automatic

conversion and handling of binary file types

Example: Example (Vim)

```
" Toggle hex mode function
function ToggleHex()
  let l:modified=&mod
  let l:oldreadonly=&readonly
  let &readonly=0
  let l:oldmodifiable=&modifiable
  let &modifiable=1
  if !exists("b:editHex") || !b:editHex
    let b:oldft=&ft
    let b:oldbin=&bin
    setlocal binary
    silent :e
    let &ft="xxd"
    let b:editHex=1
    %!xxd
  else
    let &ft=b:oldft
    if !b:oldbin
      setlocal nobinary
    endif
    let b:editHex=0
    %!xxd -r
  endif
  let &mod=l:modified
  let &readonly=l:oldreadonly
  let &modifiable=l:oldmodifiable
endfunction

" Mapping to toggle hex mode
command -bar Hexmode call ToggleHex()
```

Example: Example (Neovim)

```
local function toggle_hex()
  local modified = vim.o.modified
  local readonly = vim.o.readonly
  local modifiable = vim.bo.modifiable

  vim.o.readonly = false
  vim.bo.modifiable = true

  if not vim.b.edit_hex then
    vim.b.old_ft = vim.bo.filetype
    vim.b.old_bin = vim.bo.binary
    vim.bo.binary = true
    vim.cmd('silent edit')
    vim.bo.filetype = 'xxd'
    vim.b.edit_hex = true
    vim.cmd('%!xxd')
  else
    vim.bo.filetype = vim.b.old_ft
```

```
    if not vim.b.old_bin then
      vim.bo.binary = false
    end
    vim.b.edit_hex = false
    vim.cmd('%!xxd -r')
  end

  vim.o.modified = modified
  vim.o.readonly = readonly
  vim.bo.modifiable = modifiable
end

-- Create Hexmode command
vim.api.nvim_create_user_command('Hexmode', toggle_hex, {})

-- Optional key mapping
vim.keymap.set('n', '<C-H>', ':Hexmode<CR>', { noremap = true, silent = true
↪  })
```

Source: vim.fandom.com

9.4 Automatic Write on Buffer Switch

Category: buffer_management
Tags: auto-write, buffer, file-save

Automatically save files when switching buffers or performing certain actions

Example: Example (Vim)

```
set autowrite
set autowriteall
```

Example: Example (Neovim)

```
vim.opt.autowrite = true
vim.opt.autowriteall = true
```

Source: vim.fandom.com

9.5 Automatically Open All Buffers in Tabs

Category: buffer_management
Tags: buffers, tabs, workflow

Automatically convert all opened buffers into tabs for easier navigation

Example: Example (Vim)

```
autocmd BufReadPost * tab ball
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    vim.cmd('tab ball')
  end
})
```

Source: vim.fandom.com

9.6 Automatically Switch to Existing Vim Instance

Category: buffer_management
Tags: remote-editing, multi-instance, vim-server

Automatically detect and switch to an existing Vim instance when opening a file that's already open in another Vim window

Example: Example (Vim)

```
function! AskVims()
  let full_name = escape(expand("<afile>:p"), ' ')
  for i in split(serverlist())
    if i != v:servername
      if remote_expr(i, 'bufexists("' . full_name . '")')
        echo 'found in ' . i
        call remote_foreground(i)
        call remote_expr(i, 'foreground()')
        let v:swapchoice = 'q'
        return
      endif
    endif
  endfor
  echo "not found"
  let v:swapchoice = ''
endfunction
auto SwapExists * call AskVims()
```

Example: Example (Neovim)

```
local function ask_vims()
  local full_name = vim.fn.escape(vim.fn.expand('<afile>:p'), ' ')
  local servers = vim.fn.split(vim.fn.serverlist())

  for _, server in ipairs(servers) do
    if server ~= vim.v.servername then
```

```
      if vim.fn.remote_expr(server, 'bufexists("' .. full_name .. '")') == 1
      ↪  then
        print('found in ' .. server)
        vim.fn.remote_foreground(server)
        vim.fn.remote_expr(server, 'foreground()')
        vim.v.swapchoice = 'q'
        return
      end
    end
  end

  print('not found')
  vim.v.swapchoice = ''
end

vim.api.nvim_create_autocmd('SwapExists', {
  callback = ask_vims
})
```

Source: vim.fandom.com

9.7 Buffer namespaces for isolated state

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

Use namespaces to manage buffer decorations and state in isolation.

Example: Example (Neovim)

```
-- 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}
)
```

Source: Community contributed

9.8 Buffer-local autocommands

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

Create autocommands that only trigger for specific buffers.

Example: Example (Neovim)

```
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,
})
```

Source: Community contributed

9.9 Buffer-local keymaps

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

Create keymaps that only work in specific buffers.

Example: Example (Neovim)

```
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
```

Source: Community contributed

9.10 Buffer-local variables

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

Store buffer-specific data using buffer-local variables.

Example: Example (Neovim)

```
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
```

Source: Community contributed

9.11 Build a simple notes buffer system

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

Create a complete notes system using scratch buffers.

Example: Example (Neovim)

```
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
```

Source: Community contributed

9.12 Close all buffers

Category: Buffers
Tags: buffers, close

Example: Example (Neovim)

```
:%bd

%bd!
```

Source: Community contributed

9.13 Close all buffers except the current one

Category: Buffers
Tags: buffers, close

Example: Example (Vim)

```
vim.keymap.set("n", "<leader>X", "<cmd>%bd|e#<cr>", { desc = "Close other",
↪  noremap = true, silent = true })
```

Example: Example (Neovim)

```
:%bd|e#
```

Source: Community contributed

9.14 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: Example (Neovim)

```
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"})
```

Source: Community contributed

9.15 Create Floating Buffer Menu in GVim

Category: buffer_management
Tags: buffer-navigation, menu-customization, gui

Use :tearoff Buffers to create a floating menu for easy buffer switching in GVim

Example: Example (Vim)

```
" Auto-open buffer menu on startup
autocmd VimEnter * tearoff Buffers

" Limit buffer menu path length
let g:bmenu_max_pathlen=0
```

Example: Example (Neovim)

```
-- Neovim equivalent (note: may require different approach)
-- Consider using telescope.nvim or other buffer management plugins
vim.api.nvim_create_autocmd('VimEnter', {
  callback = function()
    vim.cmd('tearoff Buffers')
  end
})

-- Limit buffer menu path length
vim.g.bmenu_max_pathlen = 0
```

Source: vim.fandom.com

9.16 Create a GDB Console Buffer in Vim

Category: buffer_management
Tags: debugging, gdb, buffer-management

Enhance gdbvim to create a console buffer for capturing and displaying GDB output within Vim

Example: Example (Vim)

```
function! s:GdbFocusBuf(nameref)
  let l:oldnr = bufnr("%")
  let l:win_nu = bufwinnr(a:nameref)
  if l:win_nu > 0
    execute l:win_nu "wincmd w"
  else
    if bufexists(a:nameref)
      execute "sbuffer" a:nameref
    else
      execute "new" a:nameref
    endif
  endif
  return l:oldnr
endfunction
```

Example: Example (Neovim)

```
function _G.gdb_focus_buf(nameref)
  local oldnr = vim.fn.bufnr('%')
  local win_nu = vim.fn.bufwinnr(nameref)
  if win_nu > 0 then
    vim.cmd(win_nu .. 'wincmd w')
  else
    if vim.fn.bufexists(nameref) then
      vim.cmd('sbuffer ' .. nameref)
    else
      vim.cmd('new ' .. nameref)
    end
  end
  return oldnr
end
```

Source: vim.fandom.com

9.17 Create custom buffer picker

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

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

Example: Example (Neovim)

```
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, {})
```

Source: Community contributed

9.18 Create floating window with scratch buffer

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

Combine scratch buffers with floating windows for temporary UI elements.

Example: Example (Neovim)

```
-- 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"
```

Source: Community contributed

9.19 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: Example (Neovim)

```
-- 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)
```

Source: Community contributed

9.20 Create read-only buffer

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

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

Example: Example (Neovim)

```
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)
```

Source: Community contributed

9.21 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: Example (Neovim)

```
  -- 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)
```

Source: Community contributed

9.22 Create terminal buffer programmatically

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

Create and manage terminal buffers using the API.

Example: Example (Neovim)

```
  -- 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 = "?"})
```

Source: Community contributed

9.23 Customize Buffer Menu Display for Quick Navigation

Category: buffer_management
Tags: buffers, menu-customization, navigation

Modify buffer menu to show numbered buffers for faster selection, with single-keystroke access for first 9 buffers

Example: Example (Vim)

```
" In menu.vim, modify buffer name display
let name2 = '&' . a:bnum . '. ' . name2

" Conditional display for more than 9 buffers
if a:bnum >= 10
  let name2 = name2 . ' (' . a:bnum . ')'
else
  let name2 = '&' . a:bnum . '. ' . name2
endif
```

Example: Example (Neovim)

```
-- Lua equivalent (conceptual, as this is a menu.vim function)
function BufferMenuDisplay(name, bnum)
  if bnum >= 10 then
    return string.format('%s (%d)', name, bnum)
  else
    return string.format('%d. %s', bnum, name)
  end
end
```

Source: vim.fandom.com

9.24 Cycle Through Buffers Efficiently

Category: buffer_management
Tags: buffer-navigation, key-mapping, cycle-buffers

Provides flexible methods to cycle through buffers, including handling unlisted buffers

and maintaining context between different buffer types

Example: Example (Vim)

```
" Simple buffer cycling
nnoremap <C-n> :bnext<CR>
nnoremap <C-p> :bprevious<CR>

" Advanced buffer cycling with unlisted buffers support
function! SwitchToNextBuffer(incr)
  let help_buffer = (&filetype == 'help')
  let current = bufnr("%")
  let last = bufnr("$")
  let new = current + a:incr
  while 1
    if new != 0 && bufexists(new) && ((getbufvar(new, "&filetype") ==
    ↪  'help') == help_buffer)
      execute ":buffer ".new
      break
    else
      let new = new + a:incr
      if new < 1
        let new = last
      elseif new > last
        let new = 1
      endif
      if new == current
        break
      endif
    endif
  endwhile
endfunction
nnoremap <silent> <C-n> :call SwitchToNextBuffer(1)<CR>
nnoremap <silent> <C-p> :call SwitchToNextBuffer(-1)<CR>
```

Example: Example (Neovim)

```
-- Simple buffer cycling
vim.keymap.set('n', '<C-n>', ':bnext<CR>', { noremap = true })
vim.keymap.set('n', '<C-p>', ':bprevious<CR>', { noremap = true })

-- Advanced buffer cycling with unlisted buffers support
local function switch_to_next_buffer(incr)
  local help_buffer = (vim.bo.filetype == 'help')
  local current = vim.fn.bufnr('%')
  local last = vim.fn.bufnr('$')
  local new = current + incr

  while true do
    if new ~= 0 and vim.fn.bufexists(new) and
       (vim.fn.getbufvar(new, '&filetype') == 'help') == help_buffer then
      vim.cmd('buffer ' .. new)
      break
    else
      new = new + incr
```

```
      if new < 1 then
        new = last
      elseif new > last then
        new = 1
      end

      if new == current then
        break
      end
    end
  end
end

vim.keymap.set('n', '<C-n>', function() switch_to_next_buffer(1) end, {
↪  silent = true })
vim.keymap.set('n', '<C-p>', function() switch_to_next_buffer(-1) end, {
↪  silent = true })
```

Source: vim.fandom.com

9.25 Cycle Through Buffers Intelligently

Category: buffer_management
Tags: buffer-navigation, key-mapping, advanced-cycling

Advanced buffer cycling function that handles help buffers and gaps in buffer numbering, allowing seamless navigation between buffers

Example: Example (Vim)

```
function! SwitchToNextBuffer(incr)
  let help_buffer = (&filetype == 'help')
  let current = bufnr("%")
  let last = bufnr("$")
  let new = current + a:incr
  while 1
    if new != 0 && bufexists(new) && ((getbufvar(new, "&filetype") ==
    ↪  'help') == help_buffer)
      execute ":buffer ".new
      break
    else
      let new = new + a:incr
      if new < 1
        let new = last
      elseif new > last
        let new = 1
      endif
      if new == current
        break
      endif
    endif
  endwhile
```

```
endfunction
nnoremap <silent> <C-n> :call SwitchToNextBuffer(1)<CR>
nnoremap <silent> <C-p> :call SwitchToNextBuffer(-1)<CR>
```

Example: Example (Neovim)

```
function _G.switch_to_next_buffer(incr)
  local help_buffer = vim.bo.filetype == 'help'
  local current = vim.fn.bufnr('%')
  local last = vim.fn.bufnr('$')
  local new = current + incr

  while true do
    if new ~= 0 and vim.fn.bufexists(new) and
       (vim.fn.getbufvar(new, '&filetype') == 'help') == help_buffer then
      vim.cmd('buffer ' .. new)
      break
    else
      new = new + incr
      if new < 1 then
        new = last
      elseif new > last then
        new = 1
      end

      if new == current then
        break
      end
    end
  end
end

vim.keymap.set('n', '<C-n>', function() _G.switch_to_next_buffer(1) end, {
↪  silent = true })
vim.keymap.set('n', '<C-p>', function() _G.switch_to_next_buffer(-1) end, {
↪  silent = true })
```

Source: vim.fandom.com

9.26 Delete Buffer Without Closing Window

Category: buffer_management
Tags: buffer, window-layout, workflow

Provides a custom command to delete a buffer while preserving the current window layout, avoiding accidental window closures

Example: Example (Vim)

```
" Define custom Bclose command in ~/.vim/plugin/bclose.vim
command! -bang -complete=buffer -nargs=? Bclose call <SID>Bclose(<q-bang>,
↪  <q-args>)
nnoremap <silent> <Leader>bd :Bclose<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for buffer management
function _G.bclose(bang, buffer)
  local btarget = vim.fn.bufnr(buffer or '%')
  if btarget < 0 then
    print('No matching buffer')
    return
  end

  -- Switch to alternate buffer or previous buffer if possible
  local alt_buf = vim.fn.bufnr('#')
  if alt_buf > 0 and vim.bo.buflisted then
    vim.cmd.buffer(alt_buf)
  else
    vim.cmd.bprevious()
  end

  -- Delete original buffer
  vim.cmd(string.format('bdelete%s %d', bang and '!' or '', btarget))
end

-- Set up mapping
vim.keymap.set('n', '<leader>bd', _G.bclose, { desc = 'Close buffer without
↪  closing window' })
```

Source: vim.fandom.com

9.27 Delete buffer safely

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

Delete buffers with proper checks and force options.

Example: Example (Neovim)

```
-- 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, {})
```

Source: Community contributed

9.28 Diff Current Buffer with Saved File

Category: buffer_management
Tags: diff, file-comparison, buffer-manipulation

Create a side-by-side diff view comparing the current buffer with its saved version, allowing quick review of unsaved changes

Example: Example (Vim)

```
function! s:DiffWithSaved()
  let filetype=&ft
  diffthis
```

```
  vnew | r # | normal! 1Gdd
  diffthis
  exe "setlocal bt=nofile bh=wipe nobl noswf ro ft=" . filetype
endfunction
com! DiffSaved call s:DiffWithSaved()
```

Example: Example (Neovim)

```
function _G.diff_with_saved()
  local filetype = vim.bo.filetype
  vim.cmd('diffthis')
  vim.cmd('vnew')
  vim.cmd('read #')
  vim.cmd('normal! 1Gdd')
  vim.cmd('diffthis')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'wipe'
  vim.bo.swapfile = false
  vim.bo.readonly = true
  vim.bo.filetype = filetype
end

vim.api.nvim_create_user_command('DiffSaved', _G.diff_with_saved, {})
```

Source: vim.fandom.com

9.29 Easy Buffer Switching with Tab Completion

Category: buffer_management
Tags: buffer, navigation, tab-completion

Use tab completion to quickly switch between buffers with partial name matching

Example: Example (Vim)

```
" Use tab completion for buffer switching
:buffer <Tab>  " Cycle through partial buffer name matches

" Function to select buffer with partial matches
function! BufSel(pattern)
  let bufcount = bufnr("$")
  let currbufnr = 1
  let nummatches = 0
  let firstmatchingbufnr = 0
  while currbufnr <= bufcount
    if(bufexists(currbufnr))
      let currbufname = bufname(currbufnr)
      if(match(currbufname, a:pattern) > -1)
        echo currbufnr . ": ". bufname(currbufnr)
        let nummatches += 1
        let firstmatchingbufnr = currbufnr
```

```
      endif
    endif
    let currbufnr = currbufnr + 1
  endwhile
  if(nummatches == 1)
    execute ":buffer ". firstmatchingbufnr
  elseif(nummatches > 1)
    let desiredbufnr = input("Enter buffer number: ")
    if(strlen(desiredbufnr) != 0)
      execute ":buffer ". desiredbufnr
    endif
  else
    echo "No matching buffers"
  endif
endfunction

command! -nargs=1 Bs :call BufSel("<args>")
```

Example: Example (Neovim)

```
-- Function to select buffer with partial matches
local function buf_sel(pattern)
  local bufcount = vim.fn.bufnr('$')
  local nummatches = 0
  local firstmatchingbufnr = 0

  for currbufnr = 1, bufcount do
    if vim.fn.bufexists(currbufnr) == 1 then
      local currbufname = vim.fn.bufname(currbufnr)
      if currbufname:find(pattern) then
        print(string.format('%d: %s', currbufnr, currbufname))
        nummatches = nummatches + 1
        firstmatchingbufnr = currbufnr
      end
    end
  end

  if nummatches == 1 then
    vim.cmd('buffer ' .. firstmatchingbufnr)
  elseif nummatches > 1 then
    local desiredbufnr = vim.fn.input('Enter buffer number: ')
    if #desiredbufnr > 0 then
      vim.cmd('buffer ' .. desiredbufnr)
    end
  else
    print('No matching buffers')
  end
end

-- Create a user command
vim.api.nvim_create_user_command('Bs', function(opts)
  buf_sel(opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

9.30 Efficient Buffer Navigation

Category: buffer_management
Tags: buffers, navigation, jumping

Quickly navigate and manage buffers using built-in Vim commands

Example: Example (Vim)

```
:bnext
:bprev
:b <number>
:b <partial_name>
Ctrl-^
Ctrl-O
Ctrl-I
```

Example: Example (Neovim)

```
-- Next/Previous Buffer
vim.cmd('bnext')
vim.cmd('bprev')

-- Jump between buffers
vim.keymap.set('n', '<C-^>', '<C-^>', { desc = 'Toggle last buffer' })
vim.keymap.set('n', '<C-o>', '<C-o>', { desc = 'Jump back' })
vim.keymap.set('n', '<C-i>', '<C-i>', { desc = 'Jump forward' })
```

Source: vim.fandom.com

9.31 Efficient Buffer Switching with Tab Support

Category: buffer_management
Tags: buffer-navigation, tab-integration, workflow

Configure buffer switching to use tabs, allowing seamless navigation between buffers across different tabs

Example: Example (Vim)

```
" Configure buffer switching to use tabs
set switchbuf=usetab
nnoremap <F8> :sbnext<CR>
nnoremap <S-F8> :sbprevious<CR>
```

Example: Example (Neovim)

```
-- Configure buffer switching to use tabs
vim.opt.switchbuf = 'usetab'
```

```
vim.keymap.set('n', '<F8>', ':sbnext<CR>', { noremap = true })
vim.keymap.set('n', '<S-F8>', ':sbprevious<CR>', { noremap = true })
```

Source: vim.fandom.com

9.32 Enhanced Alternate File Navigation

Category: buffer_management
Tags: buffer-navigation, key-mapping, workflow

Extends Ctrl-^ to jump to next file when no alternate file exists, improving file switching workflow

Example: Example (Vim)

```
function! MySwitch()
  if expand('#')==="" | silent! next
  else
    exe "normal! \<c-^>"
  endif
endfu
map <C-^> :call MySwitch()<CR>
```

Example: Example (Neovim)

```
function _G.enhanced_file_switch()
  if vim.fn.expand('#') == '' then
    vim.cmd.next()
  else
    vim.cmd('normal! ^')
  end
end

vim.keymap.set('n', '<C-^>', _G.enhanced_file_switch, { desc = 'Switch to
↪  alternate or next file' })
```

Source: vim.fandom.com

9.33 Execute Command Across Multiple Buffers

Category: buffer_management
Tags: buffer, automation, command-line

Run a command or macro across all open buffers, with options to save or update files automatically

Example: Example (Vim)

```
:bufdo execute "normal! @a" | update

" Restore current buffer after command
function! BufDo(command)
  let currBuff=bufnr("%")
  execute 'bufdo ' . a:command
  execute 'buffer ' . currBuff
endfunction
com! -nargs=+ -complete=command Bufdo call BufDo(<q-args>)
```

Example: Example (Neovim)

```
-- Execute macro across all buffers
vim.cmd('bufdo execute "normal! @a" | update')

-- Lua equivalent of BufDo function
function _G.buf_do(command)
  local current_buf = vim.api.nvim_get_current_buf()
  vim.cmd('bufdo ' .. command)
  vim.api.nvim_set_current_buf(current_buf)
end

-- Create a user command
vim.api.nvim_create_user_command('Bufdo', function(opts)
  _G.buf_do(opts.args)
end, { nargs = '+' })
```

Source: vim.fandom.com

9.34 Execute Commands Across Buffers Safely

Category: buffer_management
Tags: buffer, search-replace, automation

Perform substitution and automatically save each buffer in the argument list

Example: Example (Vim)

```
argdo %s/foo/bar/gc | w
```

Example: Example (Neovim)

```
-- Use vim.cmd for executing argdo with chained commands
vim.cmd('argdo s/foo/bar/gc | w')
```

Source: vim.fandom.com

9.35 Fast Buffer Navigation with Ctrl-^

Category: buffer_management
Tags: buffer-navigation, key-mapping

Quickly switch to buffers by number using Ctrl-^ instead of :b command

Example: Example (Vim)

```
" Use Ctrl-^ with buffer number to switch
" 1<C-^> switches to buffer 1
" <C-^> without number switches to previous buffer
```

Example: Example (Neovim)

```
-- Builtin Vim/Neovim behavior, no specific Lua configuration needed
-- Use vim.cmd() if you want to map this programmatically
vim.keymap.set('n', '<C-^>', '<C-^>', { desc = 'Switch to buffer by number'
↪  })
```

Source: vim.fandom.com

9.36 Filter Buffer Lines by Search Pattern

Category: buffer_management
Tags: search, filtering, buffer-manipulation

Quickly create a new buffer with lines matching a specific search pattern, allowing easy inspection of matching lines

Example: Example (Vim)

```
function! Gather(pattern)
  if !empty(a:pattern)
    let save_cursor = getpos(".")
    let orig_ft = &ft
    let results = []
    execute "g/" . a:pattern . "/call add(results, getline('.'))"
    call setpos('.', save_cursor)
    if !empty(results)
      new
      setlocal buftype=nofile bufhidden=hide noswapfile
      execute "setlocal filetype=".orig_ft
      call append(1, results)
      1d
    endif
  endif
endfunction

nnoremap <silent> <Leader>f :call Gather(input("Search for: "))<CR>
```

```
nnoremap <silent> <Leader>F :call Gather(@/)<CR>
```

Example: Example (Neovim)

```
function _G.gather_search_results(pattern)
  if pattern ~= "" then
    local save_cursor = vim.fn.getpos(".")
    local orig_ft = vim.o.filetype
    local results = {}

    vim.cmd(string.format("g/%s/call add(results, getline('.'))", pattern))

    vim.fn.setpos('.', save_cursor)

    if #results > 0 then
      vim.cmd('new')
      vim.bo.buftype = 'nofile'
      vim.bo.bufhidden = 'hide'
      vim.bo.swapfile = false
      vim.bo.filetype = orig_ft

      vim.api.nvim_buf_set_lines(0, 0, -1, false, results)
    end
  end
end

vim.keymap.set('n', '<Leader>f', function()
  local pattern = vim.fn.input('Search for: ')
  _G.gather_search_results(pattern)
end, { silent = true })

vim.keymap.set('n', '<Leader>F', function()
  local last_search = vim.fn.getreg('/')
  _G.gather_search_results(last_search)
end, { silent = true })
```

Source: vim.fandom.com

9.37 Flexible Buffer Switching with Function

Category: buffer_management
Tags: buffer-navigation, custom-function, fuzzy-matching

A custom function that allows fuzzy buffer selection, displaying matching buffers and letting you choose by number if multiple matches exist

Example: Example (Vim)

```
function! BufSel(pattern)
  let bufcount = bufnr("$")
  let currbufnr = 1
```

```
  let nummatches = 0
  let firstmatchingbufnr = 0
  while currbufnr <= bufcount
    if(bufexists(currbufnr))
      let currbufname = bufname(currbufnr)
      if(match(currbufname, a:pattern) > -1)
        echo currbufnr . ": ". bufname(currbufnr)
        let nummatches += 1
        let firstmatchingbufnr = currbufnr
      endif
    endif
    let currbufnr = currbufnr + 1
  endwhile
  if(nummatches == 1)
    execute ":buffer ". firstmatchingbufnr
  elseif(nummatches > 1)
    let desiredbufnr = input("Enter buffer number: ")
    if(strlen(desiredbufnr) != 0)
      execute ":buffer ". desiredbufnr
    endif
  else
    echo "No matching buffers"
  endif
endfunction

command! -nargs=1 Bs :call BufSel("<args>")
```

Example: Example (Neovim)

```
function _G.buf_select(pattern)
  local bufcount = vim.fn.bufnr('$')
  local matches = {}

  for bufnr = 1, bufcount do
    if vim.fn.bufexists(bufnr) == 1 then
      local bufname = vim.fn.bufname(bufnr)
      if bufname:find(pattern) then
        table.insert(matches, {bufnr = bufnr, name = bufname})
      end
    end
  end

  if #matches == 1 then
    vim.cmd('buffer ' .. matches[1].bufnr)
  elseif #matches > 1 then
    print('Matching Buffers:')
    for _, match in ipairs(matches) do
      print(match.bufnr .. ': ' .. match.name)
    end
    local choice = vim.fn.input('Enter buffer number: ')
    if choice ~= '' then
      vim.cmd('buffer ' .. choice)
    end
  else
```

```
    print('No matching buffers')
  end
end

vim.api.nvim_create_user_command('Bs', function(opts)
  _G.buf_select(opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

9.38 Get buffer by name or number

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

Find buffers by name or number programmatically.

Example: Example (Neovim)

```
-- 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
```

Source: Community contributed

9.39 Handle Modified Buffers Flexibly

Category: buffer_management
Tags: buffer-options, file-saving, workflow

Multiple strategies for handling modified buffers when switching, including auto-save and hidden buffer modifications

Example: Example (Vim)

```
" Option 1: Auto save buffers
set autowrite

" Option 2: Allow hidden modified buffers
set hidden

" Option 3: Force buffer switch without saving
:bnext!
:bprevious!
```

Example: Example (Neovim)

```
-- Option 1: Auto save buffers
vim.opt.autowrite = true

-- Option 2: Allow hidden modified buffers
vim.opt.hidden = true

-- Option 3: Force buffer switch without saving can be done via:
-- :bnext! or :bprevious! in command mode
```

Source: vim.fandom.com

9.40 Improved Hex Mode Editing in Vim

Category: buffer_management
Tags: hex-editing, binary-files, file-manipulation

Provides a robust method to toggle hex mode for binary file editing using xxd, with automatic handling of binary file conversions

Example: Example (Vim)

```
" Toggle Hex Mode
command -bar Hexmode call ToggleHex()

function ToggleHex()
  let l:modified=&mod
  let l:oldreadonly=&readonly
  let &readonly=0
  let l:oldmodifiable=&modifiable
  let &modifiable=1
  if !exists("b:editHex") || !b:editHex
    let b:oldft=&ft
    let b:oldbin=&bin
    setlocal binary
    silent :e
    let &ft="xxd"
    let b:editHex=1
    %!xxd
  else
    let &ft=b:oldft
    if !b:oldbin
      setlocal nobinary
    endif
    let b:editHex=0
    %!xxd -r
  endif
  let &mod=l:modified
  let &readonly=l:oldreadonly
  let &modifiable=l:oldmodifiable
endfunction
```

Example: Example (Neovim)

```
-- Hex Mode Toggle in Neovim
local function toggle_hex_mode()
  -- Save current buffer state
  local modified = vim.bo.modified
  local readonly = vim.bo.readonly
  local modifiable = vim.bo.modifiable

  vim.bo.readonly = false
  vim.bo.modifiable = true

  if not vim.b.edit_hex then
    vim.b.old_ft = vim.bo.filetype
    vim.bo.binary = true
    vim.cmd('silent edit')
    vim.bo.filetype = 'xxd'
    vim.b.edit_hex = true
    vim.cmd('%!xxd')
  else
    vim.bo.filetype = vim.b.old_ft
    if not vim.b.old_bin then
      vim.bo.binary = false
```

```
    end
    vim.b.edit_hex = false
    vim.cmd('%!xxd -r')
  end

  vim.bo.modified = modified
  vim.bo.readonly = readonly
  vim.bo.modifiable = modifiable
end

-- Create command and keymapping
vim.api.nvim_create_user_command('HexMode', toggle_hex_mode, {})
vim.keymap.set('n', '<C-H>', ':HexMode<CR>', { noremap = true, silent = true
↪  })
```

Source: vim.fandom.com

9.41 List Buffers Sorted Alphabetically

Category: buffer_management
Tags: buffer, sorting, custom-command

Create a custom command to list buffers sorted by filename instead of buffer number

Example: Example (Vim)

```
command! -bang Ls redir @" | silent ls<bang> | redir END | echo " " |
\ perl {
\ my $msg=VIM::Eval('@"');
\ my %list=();
\ my $key, $value;
\ while($msg =~ m/(.*?line\s+\d+)/g)
\ {
\ $value = $1;
\ $value =~ m/"([^"]+)"/);
\ $key = $1;
\ ($^O =~ /mswin/i) and $key = lc($key);
\ $list{$key} = $value;
\ }
\ my $msg = '';
\ for $key (sort keys %list)
\ {
\ $msg .= "$list{$key}\n";
\ }
\ VIM::Msg($msg);
\ }
\ <CR>
```

Example: Example (Neovim)

```
-- Note: Requires a different approach in Neovim without Perl
-- A pure Lua solution would involve:
function SortBuffersByName()
  local buffers = {}
  for buf = 1, vim.fn.bufnr('$') do
    if vim.fn.bufexists(buf) == 1 then
      local name = vim.fn.bufname(buf)
      table.insert(buffers, {name = name, bufnr = buf})
    end
  end

  table.sort(buffers, function(a, b) return a.name < b.name end)

  for _, buf in ipairs(buffers) do
    print(vim.fn.bufname(buf.bufnr))
  end
end

-- Create a user command
vim.api.nvim_create_user_command('Ls', SortBuffersByName, {})
```

Source: vim.fandom.com

9.42 Open Multiple Files in Separate Tabs

Category: buffer_management
Tags: tabs, file-opening, workflow

Open multiple files, each in its own tab, either via command line or autocmd

Example: Example (Vim)

```
" Open files in tabs from command line
gvim -p *.py

" Automatically open buffers in tabs
:au BufAdd,BufNewFile * nested tab sball
```

Example: Example (Neovim)

```
-- Open files in tabs from command line: use vim.fn.argv() and -p flag

-- Lua equivalent for auto-tabbing buffers
vim.api.nvim_create_autocmd({'BufAdd', 'BufNewFile'}, {
  nested = true,
  callback = function()
    vim.cmd('tab sball')
  end
})
```

Source: vim.fandom.com

9.43 Optimize Vim for Large File Performance

Category: buffer_management
Tags: performance, file-handling, large-files

Automatically disable resource-intensive features for large files to improve loading speed and memory usage

Example: Example (Vim)

```
" Protect large files from overhead
let g:LargeFile = 1024 * 1024 * 10
augroup LargeFile
  au!
  autocmd BufReadPre * let f=getfsize(expand("<afile>")) | if f >
  ↪  g:LargeFile || f == -2 | call LargeFile() | endif
augroup END

function! LargeFile()
  set eventignore+=FileType
  setlocal bufhidden=unload
  setlocal buftype=nowrite
  setlocal undolevels=-1
endfunction
```

Example: Example (Neovim)

```
vim.g.LargeFile = 1024 * 1024 * 10

vim.api.nvim_create_augroup('LargeFile', { clear = true })
vim.api.nvim_create_autocmd('BufReadPre', {
  group = 'LargeFile',
  callback = function()
    local file_size = vim.fn.getfsize(vim.fn.expand('<afile>'))
    if file_size > vim.g.LargeFile or file_size == -2 then
      vim.opt.eventignore:append('FileType')
      vim.bo.bufhidden = 'unload'
      vim.bo.buftype = 'nowrite'
      vim.bo.undolevels = -1
    end
  end
})
```

Source: vim.fandom.com

9.44 Persist Buffer List Between Vim Sessions

Category: buffer_management
Tags: buffer, session-management, viminfo

Automatically remember opened buffers when restarting Vim, allowing you to quickly

restore your previous editing context

Example: Example (Vim)

```
:set viminfo^=%
```

Example: Example (Neovim)

```
-- Set viminfo to remember buffer list
vim.opt.viminfo:append('%')
```

Source: vim.fandom.com

9.45 Preserve Window View When Switching Buffers

Category: buffer_management
Tags: buffer-navigation, window-view, autocmd

Automatically save and restore window view (scroll position) when switching between buffers, preventing unexpected screen repositioning

Example: Example (Vim)

```
function! AutoSaveWinView()
    if !exists("w:SavedBufView")
        let w:SavedBufView = {}
    endif
    let w:SavedBufView[bufnr("%")] = winsaveview()
endfunction

function! AutoRestoreWinView()
    let buf = bufnr("%")
    if exists("w:SavedBufView") && has_key(w:SavedBufView, buf)
        let v = winsaveview()
        let atStartOfFile = v.lnum == 1 && v.col == 0
        if atStartOfFile && !&diff
            call winrestview(w:SavedBufView[buf])
        endif
        unlet w:SavedBufView[buf]
    endif
endfunction

if v:version >= 700
    autocmd BufLeave * call AutoSaveWinView()
    autocmd BufEnter * call AutoRestoreWinView()
endif
```

Example: Example (Neovim)

```
local function auto_save_win_view()
    if vim.w.SavedBufView == nil then
        vim.w.SavedBufView = {}
    end
    vim.w.SavedBufView[vim.api.nvim_get_current_buf()] =
    ↪  vim.fn.winsaveview()
end

local function auto_restore_win_view()
    local buf = vim.api.nvim_get_current_buf()
    if vim.w.SavedBufView and vim.w.SavedBufView[buf] then
        local v = vim.fn.winsaveview()
        local at_start_of_file = v.lnum == 1 and v.col == 0
        if at_start_of_file and not vim.wo.diff then
            vim.fn.winrestview(vim.w.SavedBufView[buf])
        end
        vim.w.SavedBufView[buf] = nil
    end
end

local augroup = vim.api.nvim_create_augroup('PreserveWindowView', { clear =
↪  true })
vim.api.nvim_create_autocmd('BufLeave', {
    group = augroup,
    callback = auto_save_win_view
})
vim.api.nvim_create_autocmd('BufEnter', {
    group = augroup,
    callback = auto_restore_win_view
})
```

Source: vim.fandom.com

9.46 Prevent Tab Opening for Help and Explore

Category: buffer_management
Tags: tabs, autocmd, buffer-management

Conditionally open tabs, avoiding issues with help and file explorer buffers

Example: Example (Vim)

```
au BufNewFile,BufRead * nested
  \ if &buftype != "help" |
  \   tab sball |
  \ endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  nested = true,
  callback = function()
    if vim.opt.buftype:get() ~= 'help' then
      vim.cmd('tab sball')
    end
  end
})
```

Source: vim.fandom.com

9.47 Python Help in Preview Window

Category: buffer_management
Tags: python, documentation, preview-window

Display Python documentation in Vim's preview window, supporting module and method lookups

Example: Example (Vim)

```
command! -nargs=1 -bar Pyhelp :call ShowPydoc(<f-args>)
function! ShowPydoc(what)
  let path = $TEMP . '/' . a:what . '.pydoc'
  call system(s:pydoc_path . " " . a:what . (&shellredir))
  execute "pedit" fnameescape(path)
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Pyhelp', function(opts)
  local what = opts.args
  local path = vim.fn.tempname() .. '/' .. what .. '.pydoc'
  vim.fn.system('pydoc ' .. what .. ' > ' .. path)
  vim.cmd.pedit(path)
end, { nargs = 1 })
```

Source: vim.fandom.com

9.48 Quick Buffer List and Selection Mapping

Category: buffer_management
Tags: buffer, mapping, navigation

Create a quick mapping to list buffers and allow easy selection by number

Example: Example (Vim)

```
" Quick buffer list and selection
:nnoremap <F5> :buffers<CR>:buffer<Space>
```

Example: Example (Neovim)

```
-- Quick buffer list and selection
vim.keymap.set('n', '<F5>', function()
  vim.cmd('buffers')
  vim.cmd('buffer ')
end, { desc = 'List and select buffers' })
```

Source: vim.fandom.com

9.49 Quick Buffer List and Switch Mapping

Category: buffer_management
Tags: key-mapping, buffer-navigation

A simple mapping to quickly list buffers and switch between them using a single key press

Example: Example (Vim)

```
nnoremap <F5> :buffers<CR>:buffer<Space>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', function()
  vim.cmd('buffers')
  vim.cmd('buffer ')
end, { desc = 'List and switch buffers' })
```

Source: vim.fandom.com

9.50 Quick Buffer List with Timed Display

Category: buffer_management
Tags: buffer-navigation, key-mapping, productivity

Quickly display buffer list with a timed display, useful for managing multiple open buffers

Example: Example (Vim)

```
nnoremap <m-:> :ls|sleep<CR><CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<m-:>', function()
  vim.cmd('ls')
  vim.fn.sleep(1)
end, { desc = 'Display buffer list briefly' })
```

Source: vim.fandom.com

9.51 Quick Buffer Navigation Shortcuts

Category: buffer_management
Tags: buffer-navigation, key-mapping, productivity

Easily switch between the first nine buffers using Alt+number keys or cycle through buffers with Alt+Left/Right

Example: Example (Vim)

```
" Method 1: Direct buffer access
:map <M-1> :confirm :b1 <CR>
:map <M-2> :confirm :b2 <CR>
:map <M-9> :confirm :b9 <CR>

" Method 2: Cycle through buffers
:nmap <M-Left> :bprev<CR>
:nmap <M-Right> :bnext<CR>
```

Example: Example (Neovim)

```
-- Method 1: Direct buffer access
for i = 1, 9 do
  vim.keymap.set('n', '<M-' .. i .. '>', function()
    vim.cmd('confirm buffer ' .. i)
  end)
end

-- Method 2: Cycle through buffers
vim.keymap.set('n', '<M-Left>', ':bprev<CR>')
vim.keymap.set('n', '<M-Right>', ':bnext<CR>')
```

Source: vim.fandom.com

9.52 Quick Buffer Navigation with Mouse in GVim

Category: buffer_management
Tags: mouse-navigation, buffer-switching, mapping

Enhance buffer switching in GVim by remapping right mouse click to exit modes and

interact with mini buffer explorer

Example: Example (Vim)

```
imap <RightMouse> <Esc>
nmap <RightMouse> i<LeftMouse>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<RightMouse>', '<Esc>', { desc = 'Exit insert mode with
↪  right mouse' })
vim.keymap.set('n', '<RightMouse>', 'i<LeftMouse>', { desc = 'Enter insert
↪  mode at mouse click' })
```

Source: vim.fandom.com

9.53 Quick Buffer and Tab Management

Category: buffer_management
Tags: buffers, tabs, navigation

Flexible buffer switching with tab support, allowing easy navigation between buffers across tabs

Example: Example (Vim)

```
set switchbuf=usetab
nnoremap <F8> :sbnext<CR>
nnoremap <S-F8> :sbprevious<CR>
```

Example: Example (Neovim)

```
-- Buffer switching with tab support
vim.opt.switchbuf = 'usetab'
vim.keymap.set('n', '<F8>', ':sbnext<CR>', { noremap = true })
vim.keymap.set('n', '<S-F8>', ':sbprevious<CR>', { noremap = true })
```

Source: vim.fandom.com

9.54 Quick Jump to Recent Buffers

Category: buffer_management
Tags: buffer-navigation, jump-list, custom-function

Provides a custom function to quickly jump to recently visited buffers using a numbered list, with support for direct jumping via count

Example: Example (Vim)

```
function! JumpBuffers()
   let jumptxt = ""
   redir! => jumptxt
   silent jumps
   redir END
   let byName = {}
   let byIndex = []
   for line in reverse(split(jumptxt, '\n'))
      let name = strpart(line, 16)
      let bufno = bufnr(name)
      if len(name) > 0 && bufno >= 0 && !has_key(byName, name)
         let byIndex += [{'name': name, 'bufno': bufno, 'ix':
         ↪  len(byIndex)+1}]
         let byName[name] = len(byIndex)
      endif
   endfor
   if v:count > 0
      if len(byIndex) >= v:count
         echomsg "Count ".v:count." Jumps to ".byIndex[v:count-1].bufno
         execute "buffer ".byIndex[v:count-1].bufno
      endif
      return
   endif
   echohl Special
   echo "No Buffer Name"
   echohl None
   for ent in byIndex
      echo printf("%2d %6d %s", ent.ix, ent.bufno, ent.name)
   endfor
   let ix = input("Type number and <Enter> (empty cancels): ") + 0
   if ix > 0 && ix <= len(byIndex)
      execute "buffer ".byIndex[ix-1].bufno
   endif
endfunc

" Mappings
nmap go :<C-U>call JumpBuffers()<CR>
nmap <C-G><C-O> 2go
nnoremap g<C-O> go
```

Example: Example (Neovim)

```
function _G.jump_buffers()
    local jumptxt = vim.fn.execute('jumps')
    local byName = {}
    local byIndex = {}

    for _, line in ipairs(vim.fn.reverse(vim.fn.split(jumptxt, '\n'))) do
        local name = line:sub(17)
        local bufno = vim.fn.bufnr(name)
        if #name > 0 and bufno >= 0 and not byName[name] then
            table.insert(byIndex, {name = name, bufno = bufno, ix = #byIndex
            ↪  + 1})
```

```
            byName[name] = #byIndex
        end
    end

    if vim.v.count > 0 then
        if #byIndex >= vim.v.count then
            vim.cmd('buffer ' .. byIndex[vim.v.count].bufno)
            return
        end
    end

    for _, ent in ipairs(byIndex) do
        print(string.format('%2d %6d %s', ent.ix, ent.bufno, ent.name))
    end

    local ix = tonumber(vim.fn.input('Type number and <Enter> (empty
    ↪  cancels): ')) or 0
    if ix > 0 and ix <= #byIndex then
        vim.cmd('buffer ' .. byIndex[ix].bufno)
    end
end

-- Mappings
vim.keymap.set('n', 'go', _G.jump_buffers, { desc = 'Jump to recent buffers'
↪  })
vim.keymap.set('n', '<C-G><C-O>', '2go', { remap = true })
vim.keymap.set('n', 'g<C-O>', 'go', { remap = true })
```

Source: vim.fandom.com

9.55 Quick SVN Diff in Vim Buffer

Category: buffer_management
Tags: version-control, diff, subversion

Open a new buffer with SVN diff output, set syntax highlighting, and make it a temporary buffer

Example: Example (Vim)

```
map <F9> :new<CR>:read !svn diff<CR>:set syntax=diff buftype=nofile<CR>gg
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F9>', function()
  vim.cmd('new')
  vim.cmd('read !svn diff')
  vim.opt_local.syntax = 'diff'
  vim.opt_local.buftype = 'nofile'
  vim.cmd('normal! gg')
end, { desc = 'Open SVN diff in new buffer' })
```

Source: vim.fandom.com

9.56 Reload File Without Losing Undo History

Category: buffer_management
Tags: buffer, reload, undo-history

Create a custom command to reload a file while preserving undo history, which is useful when you want to revert to the file's saved state without losing recent edit context

Example: Example (Vim)

```
command! Reload %d|r|1d
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Reload', function()
  -- Delete buffer contents
  vim.api.nvim_command('normal! ggdG')
  -- Reload current file
  vim.cmd('edit!')
end, {})
```

Source: vim.fandom.com

9.57 Replace Buffer List When Switching Projects

Category: buffer_management
Tags: session-management, buffer-control, project-switching

Quickly replace existing buffer list when switching between project sessions without exiting Vim

Example: Example (Vim)

```
1,999bdelete
:so ~/v/project2.vim
```

Example: Example (Neovim)

```
-- Delete all existing buffers before loading new session
vim.cmd('1,' .. vim.fn.bufnr('$') .. 'bdelete')
vim.cmd('source ~/v/project2.vim')
```

Source: vim.fandom.com

9.58 Restore Original Buffer After Buffer Operations

Category: buffer_management
Tags: buffer-navigation, command-line, custom-functions

Create a custom function to run a command across buffers while maintaining the original buffer context

Example: Example (Vim)

```
function! BufDo(command)
  let currBuff=bufnr("%")
  execute 'bufdo ' . a:command
  execute 'buffer ' . currBuff
endfunction
com! -nargs=+ -complete=command Bufdo call BufDo(<q-args>)
```

Example: Example (Neovim)

```
function _G.buf_do(command)
  local current_buf = vim.api.nvim_get_current_buf()
  vim.cmd('bufdo ' .. command)
  vim.api.nvim_set_current_buf(current_buf)
end

vim.api.nvim_create_user_command('BufDo', function(opts)
  _G.buf_do(opts.args)
end, { nargs = '+' })
```

Source: vim.fandom.com

9.59 Restore Original Buffer After Multi-Buffer Command

Category: buffer_management
Tags: buffer-navigation, buffer-restore

Create a function to run commands across buffers while preserving the original buffer context

Example: Example (Vim)

```
function! BufDo(command)
  let currBuff=bufnr("%")
  execute 'bufdo ' . a:command
  execute 'buffer ' . currBuff
endfunction
com! -nargs=+ -complete=command Bufdo call BufDo(<q-args>)
```

Example: Example (Neovim)

```
function _G.buf_do(command)
  local original_buf = vim.api.nvim_get_current_buf()
  vim.cmd('bufdo ' .. command)
  vim.api.nvim_set_current_buf(original_buf)
end

vim.api.nvim_create_user_command('Bufdo', function(opts)
  _G.buf_do(opts.args)
end, { nargs = '+' })
```

Source: vim.fandom.com

9.60 Run Command Across Multiple Buffers

Category: buffer_management
Tags: bufdo, buffer-manipulation, batch-operations

Execute a command (like a macro) across all open buffers and optionally save changes

Example: Example (Vim)

```
:bufdo execute "normal! @a" | update
```

Example: Example (Neovim)

```
-- Run macro in register 'a' across all buffers and save changes
for _, bufnr in ipairs(vim.api.nvim_list_bufs()) do
  vim.api.nvim_set_current_buf(bufnr)
  vim.cmd('normal! @a')
  vim.cmd('update')
end
```

Source: vim.fandom.com

9.61 Save All Buffers at Once

Category: buffer_management
Tags: buffer-operations, file-saving, workflow

Quickly save all modified buffers without closing Vim, or save and exit all buffers in a single command

Example: Example (Vim)

```
:wa  # Write all changed buffers
:xa  # Save all and exit
:wqa # Alternative save and exit command
```

Example: Example (Neovim)

```
vim.cmd('wa')  -- Write all changed buffers
vim.cmd('xa')  -- Save all and exit
```

Source: vim.fandom.com

9.62 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: Example (Neovim)

```
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
```

Source: Community contributed

9.63 Single Window Buffer Navigation

Category: buffer_management
Tags: buffer-navigation, key-mapping, workflow

Efficiently navigate between buffers in a single window using custom key mappings

Example: Example (Vim)

```
" Navigate between buffers
:nm <A-Up> :bp!<CR>
:nm <A-Down> :bn!<CR>
:nm <C-F4> :bd!<CR>
```

Example: Example (Neovim)

```
-- Buffer navigation mappings
vim.keymap.set('n', '<A-Up>', ':bp!<CR>', { desc = 'Previous buffer' })
vim.keymap.set('n', '<A-Down>', ':bn!<CR>', { desc = 'Next buffer' })
vim.keymap.set('n', '<C-F4>', ':bd!<CR>', { desc = 'Close current buffer' })
```

Source: vim.fandom.com

9.64 Smart Buffer Switching with Tabs

Category: buffer_management
Tags: buffer-navigation, tab-management, window-management

Configure buffer switching to use tabs intelligently, allowing easy navigation between buffers across different tabs

Example: Example (Vim)

```
" Smart buffer switching
set switchbuf=usetab
nnoremap <F8> :sbnext<CR>
nnoremap <S-F8> :sbprevious<CR>
```

Example: Example (Neovim)

```
-- Smart buffer switching
vim.opt.switchbuf = 'usetab'
vim.keymap.set('n', '<F8>', ':sbnext<CR>', { noremap = true })
vim.keymap.set('n', '<S-F8>', ':sbprevious<CR>', { noremap = true })
```

Source: vim.fandom.com

9.65 Store Grep Results in a Scratch Buffer

Category: buffer_management
Tags: search, command-line, buffer

Create a custom command to capture and display global search results in a temporary buffer, allowing easy browsing of matched lines

Example: Example (Vim)

```
command! -nargs=? F let @a='' | execute 'redir @a | silent g/<args>/print' |
↪ redir END | silent execute 'new | setlocal bt=nofile | put=split(@a,
↪ '\n'') | set syntax=vera | 1d'
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('F', function(opts)
  -- Capture global search results in a scratch buffer
  local pattern = opts.args
  local results = vim.fn.execute('silent g/' .. pattern .. '/print')

  -- Create new buffer
  local buf = vim.api.nvim_create_buf(false, true)
  vim.api.nvim_set_current_buf(buf)

  -- Set buffer options
  vim.api.nvim_buf_set_option(buf, 'buftype', 'nofile')
  vim.api.nvim_buf_set_option(buf, 'swapfile', false)

  -- Insert results
  vim.api.nvim_buf_set_lines(buf, 0, -1, false, vim.split(results, '\n'))
end, { nargs = '?' })
```

Source: vim.fandom.com

9.66 Suppress File Change Warnings in Buffer

Category: buffer_management
Tags: autocmd, file-handling, buffer-settings

Temporarily suppress file change warnings when running shell commands that modify the current buffer

Example: Example (Vim)

```
function ChangeThisBuffer()
  setlocal autoread
  execute('silent !mycommand')
  set autoread<
endfunction
```

Example: Example (Neovim)

```
function ChangeThisBuffer()
  vim.opt_local.autoread = true
  vim.fn.execute('silent !mycommand')
  vim.opt_local.autoread = false
end
```

Source: vim.fandom.com

9.67 Sync Buffer Modifiability with File Readonly State

Category: buffer_management
Tags: buffer-management, file-protection, autocmd

Automatically make buffers non-modifiable when the underlying file is read-only, preventing accidental edits

Example: Example (Vim)

```
function UpdateModifiable()
  if !exists("b:setmodifiable")
    let b:setmodifiable = 0
  endif
  if &readonly
    if &modifiable
      setlocal nomodifiable
      let b:setmodifiable = 1
    endif
  else
    if b:setmodifiable
      setlocal modifiable
    endif
  endif
endfunction
autocmd BufReadPost * call UpdateModifiable()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    local buf = vim.api.nvim_get_current_buf()
    if vim.bo.readonly then
      vim.bo.modifiable = false
    else
      vim.bo.modifiable = true
    end
  end
})
```

Source: vim.fandom.com

9.68 Toggle All Buffers in Tabs

Category: buffer_management
Tags: buffers, tabs, toggle

Quickly toggle between showing all buffers in tabs and closing all but the current tab

Example: Example (Vim)

```
let notabs = 0
nnoremap <silent> <F8> :let notabs=!notabs<Bar>:if
↪  notabs<Bar>:tabo<Bar>:else<Bar>:tab ball<Bar>:tabn<Bar>:endif<CR>
```

Example: Example (Neovim)

```
-- Toggle all buffers in tabs
local notabs = false
vim.keymap.set('n', '<F8>', function()
  notabs = not notabs
  if notabs then
    vim.cmd('tabo')  -- Close all other tabs
  else
    vim.cmd('tab ball')  -- Show all buffers in tabs
    vim.cmd('tabn')  -- Go to the next tab
  end
end, { silent = true })
```

Source: vim.fandom.com

9.69 Use Temporary Files for SQL Query Results

Category: buffer_management
Tags: sql, temporary-files, external-commands

Utilize Vim's tempname() function to create and manage temporary files for SQL query results, which are automatically cleaned up

Example: Example (Vim)

```
au FileType sql map <F12> :let fname = tempname()<CR><C-W><C-O>:silent exe
↪  "w !isql -SYourServerName -DYourDatabaseName -UYourUserName
↪  -PYourPassword > " . fname<CR>:exe "split " . fname<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'sql',
  callback = function()
    vim.keymap.set('n', '<F12>', function()
      local fname = vim.fn.tempname()
      vim.cmd('silent w !isql -SYourServerName -DYourDatabaseName
      ↪  -UYourUserName -PYourPassword > ' .. fname)
      vim.cmd('split ' .. fname)
    end, { buffer = true })
  end
})
```

Source: vim.fandom.com

9.70 Version Control Diff Functions

Category: buffer_management
Tags: git, svn, version-control, diff

Quick diff functions for comparing current buffer with version control checked-out versions for Git and SVN

Example: Example (Vim)

```
function! s:DiffWithGITCheckedOut()
  let filetype=&ft
  diffthis
  vnew | exe "%!git diff " . fnameescape( expand("#:p") ) . "| patch -p 1
  ↪  -Rs -o /dev/stdout"
  exe "setlocal bt=nofile bh=wipe nobl noswf ro ft=" . filetype
  diffthis
endfunction
com! DiffGIT call s:DiffWithGITCheckedOut()
```

Example: Example (Neovim)

```
function _G.diff_with_git_checkout()
  local filetype = vim.bo.filetype
  vim.cmd('diffthis')
  vim.cmd('vnew')
  vim.cmd('read !git diff ' .. vim.fn.fnameescape(vim.fn.expand('#:p')) ..
  ↪  '| patch -p 1 -Rs -o /dev/stdout')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'wipe'
  vim.bo.swapfile = false
  vim.bo.readonly = true
  vim.bo.filetype = filetype
  vim.cmd('diffthis')
end

vim.api.nvim_create_user_command('DiffGIT', _G.diff_with_git_checkout, {})
```

Source: vim.fandom.com

9.71 Watch buffer for changes

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

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

Example: Example (Neovim)

```
-- 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
```

Source: Community contributed

CHAPTER 10

Builtin functions

10.1 Alternative Command Path Lookup in Vim

Category: builtin_functions
Tags: command-lookup, path-search

Search for command in system PATH using Vim's globpath function, works across different shell environments

Example: Example (Vim)

```
:echo globpath(substitute($PATH, ':', ',', 'g'), 'cat')
```

Example: Example (Neovim)

```
-- Lua implementation of PATH search
local function find_in_path(cmd)
  local path = vim.env.PATH:gsub(':', ',')
  return vim.fn.globpath(path, cmd)
end
```

Source: vim.fandom.com

10.2 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
print(vim.fn.bufnr('%'))
print(vim.fn.winnr('%'))
print(vim.fn.tabpagenr('%'))
print(vim.fn.winnr('%'))
```

Source: Community contributed

10.3 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.getbufline(1, 1, 10))
vim.fn.setbufline(2, 1, 'new first line')
vim.fn.append(vim.fn.line('.'), 'new line')
vim.fn.delete(vim.fn.line('.'))
```

Source: Community contributed

10.4 Convert Numbers Between Hex and Decimal

Category: builtin_functions
Tags: conversion, number-manipulation, utility

Easily convert numbers between hexadecimal and decimal formats using built-in Vim/Neovim functions with flexible conversion methods

Example: Example (Vim)

```
" Convert decimal to hex
function! Dec2hex(arg)
  return printf('%x', a:arg + 0)
endfunction

" Convert hex to decimal
```

```
function! Hex2dec(arg)
  return (a:arg =~? '^0x') ? a:arg + 0 : ('0x'.a:arg) + 0
endfunction
```

Example: Example (Neovim)

```
-- Convert decimal to hex
function _G.dec2hex(arg)
  return string.format('%x', tonumber(arg))
end

-- Convert hex to decimal
function _G.hex2dec(arg)
  arg = arg:lower():match('^0x') and arg or '0x' .. arg
  return tonumber(arg)
end
```

Source: vim.fandom.com

10.5 Create Custom Help Tags Programmatically

Category: builtin_functions
Tags: help, custom-tags, automation

Programmatically add custom help tags to make personal notes easily searchable

Example: Example (Vim)

```
function! AddMyHelpTags()
  exe 'echo -e "sfsed\t ~/Documents/mynotesfile.mytext\t/\*sfsed\*" >>
  ↪  ~/.vim/doc/tags'
  exe 'sort ~/.vim/doc/tags > /tmp/foo.vim.tags'
  exe 'mv -f /tmp/foo.vim.tags ~/.vim/doc/tags'
endfunction
```

Example: Example (Neovim)

```
local function add_help_tags()
  local tags_path = vim.fn.expand('~/.vim/doc/tags')
  local new_tag = 'sfsed\t ~/Documents/mynotesfile.mytext\t/*sfsed*\n'

  -- Append tag and sort
  vim.fn.writefile(
    vim.fn.sort(vim.fn.readfile(tags_path) + {new_tag}),
    tags_path
  )
end
```

Source: vim.fandom.com

10.6 Cursor and mark functions

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

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

Example: Example (Vim)

```
: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'
```

Example: Example (Neovim)

```
vim.fn.cursor(10, 5)
local pos = vim.fn.getpos('.')
vim.fn.setpos('.', pos)
print(vim.fn.getpos("'a"))
```

Source: Community contributed

10.7 Date and time functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.strftime('%Y-%m-%d %H:%M:%S'))
print(vim.fn.strftime('%Y-%m-%d', vim.fn.localtime()))
print(vim.fn.getftime('%Y-%m-%d', vim.fn.expand('%')))
vim.api.nvim_put({os.date('%Y-%m-%d')}, 'c', true, true)
```

Source: Community contributed

10.8 Detect Comments Across Filetypes

Category: builtin_functions
Tags: syntax, scripting, highlight

Use syntax highlighting to check if cursor is in a comment, string, or preprocessor statement, independent of filetype

Example: Example (Vim)

```
" Check if cursor is in a comment
let is_comment = synIDattr(synIDtrans(synID(line('.'), col('.'), 0)),
↪  'name') == 'Comment'

" Check if cursor is in comment, string, or preprocessor
let is_special = synIDattr(synIDtrans(synID(line('.'), col('.'), 0)),
↪  'name') =~ 'Comment\|Constant\|PreProc'
```

Example: Example (Neovim)

```
-- Check if cursor is in a comment
local function is_comment()
  return vim.fn.synIDattr(vim.fn.synIDtrans(vim.fn.synID(vim.fn.line('.'),
  ↪  vim.fn.col('.'), 0)), 'name') == 'Comment'
end

-- Check if cursor is in comment, string, or preprocessor
local function is_special()
  local syn_name =
  ↪  vim.fn.synIDattr(vim.fn.synIDtrans(vim.fn.synID(vim.fn.line('.'),
  ↪  vim.fn.col('.'), 0)), 'name')
  return syn_name:match('Comment') or syn_name:match('Constant') or
  ↪  syn_name:match('PreProc')
end
```

Source: vim.fandom.com

10.9 File and directory functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.glob('*.txt'))
print(vim.fn.globpath(&rtp, 'plugin/*.vim'))
print(vim.fn.isdirectory(vim.fn.expand('%:h')))
print(vim.fn.readable(vim.fn.expand('%')))
```

Source: Community contributed

10.10 Flexible External Command Processing with system()

Category: builtin_functions
Tags: shell-integration, scripting, text-manipulation

Demonstrates using system() function to capture and process command output safely in scripts, with proper argument escaping

Example: Example (Vim)

```
function! GetDate(format)
  let format = empty(a:format) ? '+%A %Y-%m-%d %H:%M UTC' : a:format
  let cmd = '/bin/date -u ' . shellescape(format)
  let result = substitute(system(cmd), '[\]\'[[:cntrl:]]', '', 'g')
  call setline(line('.'), getline('.') . ' ' . result)
endfunction
```

Example: Example (Neovim)

```
function GetDate(format)
  local fmt = format == '' and '+%A %Y-%m-%d %H:%M UTC' or format
  local cmd = '/bin/date -u ' .. vim.fn.shellescape(fmt)
  local result = vim.fn.substitute(vim.fn.system(cmd), '[\]\'[[:cntrl:]]',
  ↪  '', 'g')
  vim.fn.setline('.', vim.fn.getline('.') .. ' ' .. result)
end

-- Optional mapping
vim.keymap.set('n', '<F8>', function() GetDate('') end)
```

Source: vim.fandom.com

10.11 Fold information functions

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

Use folding functions to query and manipulate code folds programmatically. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.foldclosed(vim.fn.line('.')))
print(vim.fn.foldlevel(vim.fn.line('.')))
print(vim.fn.foldnext())
vim.opt.foldtext = MyCustomFoldText()
```

Source: Community contributed

10.12 Generate a Customizable Calendar in Vim

Category: builtin_functions
Tags: calendar, date-management, custom-function

Create a flexible calendar generation function that outputs days, weeks, and months to a new buffer with customizable parameters

Example: Example (Vim)

```
function! Calendar(year, month, day, weekday, week, daycount)
  new
  set buftype=nofile bufhidden=hide noswapfile
  " Function implementation to generate calendar days
  " Supports generating calendar for multiple days
endfunction

" Map to generate calendar for 1000 days
map <S-F7> :call Calendar(2002, 12, 30, 1, 1, 1000)<CR><CR>
```

Example: Example (Neovim)

```
function _G.generate_calendar(year, month, day, weekday, week, daycount)
  vim.cmd('new')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'hide'
  vim.bo.swapfile = false

  -- Lua implementation of calendar generation logic
  -- Similar structure to Vimscript version
end
```

```
-- Map to generate calendar
vim.keymap.set('n', '<S-F7>', function()
  _G.generate_calendar(2002, 12, 30, 1, 1, 1000)
end)
```

Source: vim.fandom.com

10.13 Get Cursor Position as Byte Percentage

Category: builtin_functions
Tags: status-line, cursor-position, file-metadata

Calculate and display cursor position as byte percentage instead of default line percentage, which is more accurate for files with varying line lengths

Example: Example (Vim)

```
function! Percent()
  let byte = line2byte(line(".")) + col(".") - 1
  let size = (line2byte(line("$") + 1) - 1)
  return (byte * 100) / size
endfunction
```

Example: Example (Neovim)

```
function _G.byte_percentage()
  local byte = vim.fn.line2byte(vim.fn.line('.')) + vim.fn.col('.') - 1
  local size = vim.fn.line2byte(vim.fn.line('$') + 1) - 1
  return math.floor((byte * 100) / size)
end

-- Add to statusline
vim.opt.statusline:append('%{v:lua.byte_percentage()}%')
```

Source: vim.fandom.com

10.14 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. Or:

Example: Example (Vim)

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

Example: Example (Neovim)

```
print(vim.fn.getftype(vim.fn.expand('%')))
print(vim.fn.getfperm(vim.fn.expand('%')))
print(vim.fn.getfsize(vim.fn.expand('%')))
```

Source: Community contributed

10.15 Highlighting and syntax functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.synID(vim.fn.line('.'), vim.fn.col('.'), 1))
print(vim.fn.synIDattr(vim.fn.synId(vim.fn.line('.'), vim.fn.col('.'), 1),
↪  'name'))
print(vim.fn.hlID('Comment'))
print(vim.fn.synIDattr(vim.fn.hlID('Comment'), 'fg'))
```

Source: Community contributed

10.16 Input and interaction functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

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

Source: Community contributed

10.17 Insert Current Date/Time with Strftime()

Category: builtin_functions
Tags: date, timestamp, abbreviations

Easily insert current date and time using Vim's built-in strftime() function with various formatting options

Example: Example (Vim)

```
" Insert current date/time on F5
:nnoremap <F5> "=strftime("%c")<CR>P
:inoremap <F5> <C-R>=strftime("%c")<CR>

" Abbreviation for quick timestamp
:iab <expr> dts strftime("%c")
```

Example: Example (Neovim)

```
-- Insert current date/time on F5
vim.keymap.set('n', '<F5>', function()
  vim.api.nvim_put({os.date("%c")}, '', true, true)
end)

vim.keymap.set('i', '<F5>', function()
  return os.date("%c")
end, { expr = true })

-- Create abbreviation for quick timestamp
vim.cmd.iabbrev('dts', os.date("%c"))
```

Source: vim.fandom.com

10.18 Line and column functions

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
print(vim.fn.line('.'))
print(vim.fn.col('.'))
print(vim.fn.getline('.'))
vim.fn.setline('.', 'new text')
```

Source: Community contributed

10.19 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. Or:

Example: Example (Vim)

```
: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']
```

Example: Example (Neovim)

```
local mylist = {1, 2, 3}
print(#mylist)
print(vim.tbl_isempty(mylist))
local mydict = {a = 1, b = 2}
print(mydict.a ~= nil)
print(vim.inspect(vim.tbl_keys(mydict)))
```

Source: Community contributed

10.20 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.abs(-5))
print(vim.fn.pow(2, 3))
print(vim.fn.sqrt(16))
print(vim.fn.sin(3.14159/2))
print(vim.fn.round(3.7))
```

Source: Community contributed

10.21 Measure Command Execution Time

Category: builtin_functions
Tags: performance, benchmarking, timing

Function to measure the execution time of Vim commands across multiple iterations, useful for performance testing and optimization

Example: Example (Vim)

```
function! Time(com, ...)
  let time = 0.0
  let numberOfTimes = a:0 ? a:1 : 50000
  for i in range(numberOfTimes + 1)
    let t = reltime()
    execute a:com
    let time += reltime(t)[1]
    echo i.' / '.numberOfTimes
    redraw
  endfor
```

```
  echo 'Average time: '.string(numberOfTimes / i)
endfunction
```

Example: Example (Neovim)

```
function Time(com, numberOfTimes)
  numberOfTimes = numberOfTimes or 50000
  local totalTime = 0
  for i = 1, numberOfTimes do
    local startTime = vim.fn.reltime()
    vim.cmd(com)
    local elapsedTime = vim.fn.reltime(startTime)
    totalTime = totalTime + elapsedTime[1]
    print(string.format('%d / %d', i, numberOfTimes))
    vim.cmd('redraw')
  end
  print(string.format('Average time: %.6f', totalTime / numberOfTimes))
end
```

Source: vim.fandom.com

10.22 Path manipulation functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.fnamemodify(vim.fn.expand('%'), ':p:h'))
print(vim.fn.fnamemodify(vim.fn.expand('%'), ':t:r'))
print(vim.fn.resolve(vim.fn.expand('%')))
print(vim.fn.simplify('../path/./file'))
```

Source: Community contributed

10.23 Print Full-Width Messages Without 'Press Enter'

Category: builtin_functions
Tags: scripting, message-display, vim-functions

A utility function to print messages across full screen width without triggering 'Press Enter' prompt by temporarily modifying ruler and showcmd settings

Example: Example (Vim)

```
function! WideMsg(msg)
  let x=&ruler | let y=&showcmd
  set noruler noshowcmd
  redraw
  echo a:msg
  let &ruler=x | let &showcmd=y
endfun
```

Example: Example (Neovim)

```
function _G.WideMsg(msg)
  local ruler = vim.o.ruler
  local showcmd = vim.o.showcmd
  vim.o.ruler = false
  vim.o.showcmd = false
  vim.cmd.redraw()
  vim.api.nvim_echo({{msg}}, false, {})
  vim.o.ruler = ruler
  vim.o.showcmd = showcmd
end
```

Source: vim.fandom.com

10.24 Recreate Vim Temporary File Directory

Category: builtin_functions
Tags: file-operations, temp-files, system-interaction

Programmatically recreate Vim's temporary file directory if it has been deleted, which can happen on long-running systems

Example: Example (Vim)

```
:call mkdir(fnamemodify(tempname(),':h'),'p',0700)
```

Example: Example (Neovim)

```
vim.fn.mkdir(vim.fn.fnamemodify(vim.fn.tempname(), ':h'), 'p', 700)
```

Source: vim.fandom.com

10.25 Register manipulation functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.getreg('"'))
vim.fn.setreg('a', 'hello world')
print(vim.fn.getregtype('a'))
vim.fn.setreg('+', "@")
```

Source: Community contributed

10.26 Regular expression functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
local m = vim.fn.matchadd('Search', 'TODO')
vim.fn.matchdelete(m)
print(vim.fn.matchlist('file.txt', '\\(.*\\)\\.\\(.*\\)'))
print(vim.fn.matchstr('hello123world', '\\d\\+'))
```

Source: Community contributed

10.27 Save and Restore Cursor Position Precisely

Category: builtin_functions
Tags: cursor-position, screen-position, utility-function

Create a function to precisely save and restore cursor and screen positions, which is useful when performing operations that might disrupt cursor placement

Example: Example (Vim)

```
function CurPos(action)
  if a:action == "save"
    let b:saveve = &virtualedit
    let b:savesiso = &sidescrolloff
    set virtualedit=all
    set sidescrolloff=0
    let b:curline = line(".")
    let b:curvcol = virtcol(".")
    let b:curwcol = wincol()
  elseif a:action == "restore"
    execute "normal! ".b:midline."Gzz".b:curline."G0"
    execute "normal! ".b:curvcol."|")
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.CurPos(action)
  if action == "save" then
    vim.b.saveve = vim.o.virtualedit
    vim.b.savesiso = vim.o.sidescrolloff
    vim.o.virtualedit = "all"
    vim.o.sidescrolloff = 0
    vim.b.curline = vim.fn.line(".")
    vim.b.curvcol = vim.fn.virtcol(".")
    vim.b.curwcol = vim.fn.wincol()
  elseif action == "restore" then
    vim.cmd(string.format("normal! %dGzz%dG0", vim.b.midline,
    ↪  vim.b.curline))
    vim.cmd(string.format("normal! %d|", vim.b.curvcol))
  end
end
```

Source: vim.fandom.com

10.28 Search and match functions

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

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

moving cursor. Or:

Example: Example (Vim)

```
: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)
```

Example: Example (Neovim)

```
print(vim.fn.search('pattern'))
print(vim.fn.searchpos('pattern'))
print(vim.fn.match('hello world', 'wor'))
print(vim.fn.matchend('hello world', 'wor'))
```

Source: Community contributed

10.29 String manipulation functions

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

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

Example: Example (Vim)

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

Example: Example (Neovim)

```
print(vim.fn.substitute("hello world", "world", "vim", "g"))
print(vim.fn.matchstr("file.txt", '\\.\\w\\+$'))
print(vim.fn.split("a,b,c", ","))
```

Source: Community contributed

10.30 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. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.system('date'))
print(vim.fn.systemlist('ls -la'))
print(vim.fn.geten('HOME'))
print(vim.fn.exists('$EDITOR'))
```

Source: Community contributed

10.31 Type checking functions

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

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

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.type(42))
print(vim.fn.type("string"))
print(vim.fn.type([]))
print(vim.fn.type({}))
print(vim.fn.exists('g:my_var'))
```

Source: Community contributed

10.32 Window and tab functions

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

Use window dimension and state functions to manage window layouts programmati-

cally. Or:

Example: Example (Vim)

```
: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
```

Example: Example (Neovim)

```
print(vim.fn.winheight(0))
print(vim.fn.winwidth(0))
local view = vim.fn.winsaveview()
vim.fn.winrestview(view)
print(vim.fn.tabpaebuflist())
```

Source: Community contributed

CHAPTER 11

Clever tricks

11.1 Advanced Number Increment/Decrement

Category: clever_tricks
Tags: number-manipulation, search, editing

Enhanced Ctrl-A/Ctrl-X to increment/decrement numbers across lines, with backward search option

Example: Example (Vim)

```
function! AddSubtract(char, back)
  let pattern = &nrformats =~ 'alpha' ? '[[:alpha:][:digit:]]' :
  ↪  '[[:digit:]]'
  call search(pattern, 'cw' . a:back)
  execute 'normal! ' . v:count1 . a:char
  silent! call repeat#set(":\<C-u>call AddSubtract('" .a:char. "', '"
  ↪  .a:back. "')\<CR>")
endfunction
nnoremap <silent>         <C-a> :<C-u>call AddSubtract("\<C-a>", '')<CR>
nnoremap <silent> <Leader><C-a> :<C-u>call AddSubtract("\<C-a>", 'b')<CR>
nnoremap <silent>         <C-x> :<C-u>call AddSubtract("\<C-x>", '')<CR>
nnoremap <silent> <Leader><C-x> :<C-u>call AddSubtract("\<C-x>", 'b')<CR>
```

Example: Example (Neovim)

```
local function add_subtract(char, back)
  local pattern = vim.o.nrformats:match('alpha') and '[%w]' or '[%d]'
  vim.fn.search(pattern, 'cw' .. back)
  vim.cmd('normal! ' .. (vim.v.count1 or 1) .. char)
end

vim.keymap.set('n', '<C-a>', function() add_subtract('\<C-a>', '') end, {
↪  silent = true })
vim.keymap.set('n', '<leader><C-a>', function() add_subtract('\<C-a>', 'b')
↪  end, { silent = true })
vim.keymap.set('n', '<C-x>', function() add_subtract('\<C-x>', '') end, {
↪  silent = true })
vim.keymap.set('n', '<leader><C-x>', function() add_subtract('\<C-x>', 'b')
↪  end, { silent = true })
```

Source: vim.fandom.com

11.2 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: Example (Vim)

```
: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
```

Source: Community contributed

11.3 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: Example (Vim)

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

Source: Community contributed

11.4 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: Example (Vim)

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

Source: Community contributed

11.5 C/C++ Macro Expansion in Vim

Category: clever_tricks
Tags: macro-expansion, c-cpp, development

Expand C/C++ macros using gcc in a preview window, helping developers understand macro preprocessing

Example: Example (Vim)

```
function! ExpandCMacro()
  let l:macro_file_name = "__macroexpand__" . tabpagenr()
  let l:file_row = line(".")
  let l:file_name = expand("%")
  let l:file_window = winnr()

  execute "normal! Oint " . l:macro_file_name . ";"
  execute "w"

  if bufwinnr(l:macro_file_name) != -1
    execute bufwinnr(l:macro_file_name) . "wincmd w"
    setlocal modifiable
    execute "normal! ggdG"
  else
    execute "bot 10split " . l:macro_file_name
    execute "setlocal filetype=cpp"
    execute "setlocal buftype=nofile"
    nnoremap <buffer> q :q!<CR>
  endif

  silent! execute "r!gcc -E " . l:file_name
  execute "normal! ggV/int " . l:macro_file_name . ";$\<CR>d"
  execute "normal! jdG"
  execute "%!indent -st -kr"
  execute "normal! gg=G"

  execute "normal! G"
  let l:macro_end_row = line(".")
  execute "resize " . l:macro_end_row
  execute "normal! gg"

  setlocal nomodifiable
  execute l:file_window . "wincmd w"
  execute l:file_row
  execute "normal!u"
  execute "w"

  let @/ = getline('.')
endfunction
```

Example: Example (Neovim)

```
function _G.expand_c_macro()
  local macro_file_name = string.format("__macroexpand__%d",
  ↪  vim.fn.tabpagenr())
  local file_row = vim.fn.line(".")
  local file_name = vim.fn.expand("%")
  local file_window = vim.fn.winnr()
```

```
  vim.cmd(string.format("normal! 0int %s;", macro_file_name))
  vim.cmd("w")

  if vim.fn.bufwinnr(macro_file_name) ~= -1 then
    vim.cmd(string.format("%dwincmd w", vim.fn.bufwinnr(macro_file_name)))
    vim.api.nvim_win_set_option(0, 'modifiable', true)
    vim.cmd("normal! ggdG")
  else
    vim.cmd(string.format("bot 10split %s", macro_file_name))
    vim.api.nvim_buf_set_option(0, 'filetype', 'cpp')
    vim.api.nvim_buf_set_option(0, 'buftype', 'nofile')
    vim.api.nvim_buf_set_keymap(0, 'n', 'q', ':q!<CR>', {noremap = true,
    ↪  silent = true})
  end

  vim.cmd(string.format("r!gcc -E %s", file_name))
  vim.cmd(string.format("normal! ggV/int %s;$\<CR>d", macro_file_name))
  vim.cmd("normal! jdG")
  vim.cmd("%!indent -st -kr")
  vim.cmd("normal! gg=G")

  vim.cmd("normal! G")
  local macro_end_row = vim.fn.line(".")
  vim.cmd(string.format("resize %d", macro_end_row))
  vim.cmd("normal! gg")

  vim.api.nvim_win_set_option(0, 'modifiable', false)
  vim.cmd(string.format("%dwincmd w", file_window))
  vim.cmd(tostring(file_row))
  vim.cmd("normal!u")
  vim.cmd("w")

  vim.fn.setreg('/', vim.fn.getline('.'))
end

-- Mapping
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'cpp',
  callback = function()
    vim.keymap.set('n', '<leader>m', _G.expand_c_macro, { desc = 'Expand
    ↪  C/C++ Macro' })
  end
})
```

Source: vim.fandom.com

11.6 Calculate Math Expressions In-Editor

Category: clever_tricks
Tags: calculation, external-tool, visual-mode

Use BC (basic calculator) to evaluate mathematical expressions directly within Vim, supporting complex calculations and trigonometric functions

Example: Example (Vim)

```
function! CalcBC()
  let has_equal = 0
  let @e = substitute(@e, "\n", "", "g")
  let @e = substitute(@e, '\s*$', "", "g")
  if @e =~ "=$"
    let @e = substitute(@e, '=$', "", "")
    let has_equal = 1
  endif
  let @e = substitute(@e, '\csin\s*(', "s (", "")
  let @e = substitute(@e, '\ccos\s*(', "c (", "")
  let answer = substitute(system("echo " . @e . " | bc -l"), "\n", "", "")
  if has_equal == 1
    normal `>
    exec "normal a" . answer
  else
    echo "answer = " . answer
  endif
endfunction

vnoremap ;bc "ey:call CalcBC()<CR>
```

Example: Example (Neovim)

```
function _G.calc_bc()
  local text = vim.fn.getreg('e')
  text = text:gsub('\n', ''):gsub('%s*$', '')

  local has_equal = text:match('=$')
  if has_equal then
    text = text:gsub('=$', '')
  end

  text = text:gsub('sin%s*%(', 's (')
  text = text:gsub('cos%s*%(', 'c (')

  local result = vim.fn.system(string.format('echo "%s" | bc -l',
  ↪  text)):gsub('\n', '')

  if has_equal then
    vim.cmd('normal! `>')
    vim.fn.setreg('e', result)
    vim.cmd('normal! p')
  else
    print('answer = ' .. result)
  end
end

vim.keymap.set('v', ';bc', '"ey:lua _G.calc_bc()<CR>', { noremap = true,
↪  silent = true })
```

Source: vim.fandom.com

11.7 Calculation with expression register

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

Use = register to evaluate mathematical expressions and insert results.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

11.8 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: Example (Vim)

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

Source: Community contributed

11.9 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: Example (Vim)

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

Example: Example (Neovim)

```
-- Lua:
vim.cmd('cd ' .. vim.fn.expand('%:h'))  -- change to current file's
↪  directory
vim.cmd('pwd')  -- verify current directory
vim.cmd('lcd ' .. vim.fn.expand('%:h'))  -- change local directory for
↪  current window only

-- Or using Lua API:
vim.fn.chdir(vim.fn.expand('%:h'))
```

Source: Community contributed

11.10 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: Example (Vim)

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

Source: Community contributed

11.11 Confirm Before Quitting in Insert Mode

Category: clever_tricks
Tags: key-mapping, insert-mode, workflow

Adds a confirmation dialog when accidentally typing 'wq' in insert mode, preventing unintended file saves and quits

Example: Example (Vim)

```
function WQHelper()
  let x = confirm("Current Mode == Insert-Mode!\n Would you like ':wq'?", "
  ↪  &Yes \n &No", 1, 1)
  if x == 1
    silent! :wq
  else
    "???
  endif
endfunction

iab wq <BS><Esc>:call WQHelper()<CR>
```

Example: Example (Neovim)

```
function _G.wq_helper()
  local choice = vim.fn.confirm(
    "Current Mode == Insert-Mode!\nWould you like ':wq'?",
    "&Yes\n&No",
    1
  )
  if choice == 1 then
    vim.cmd('wq')
  end
end

vim.keymap.set('i', 'wq', '<BS><Esc>:lua _G.wq_helper()<CR>')
```

Source: vim.fandom.com

11.12 Convert Between Hex and Decimal Numbers

Category: clever_tricks
Tags: conversion, numbers, utilities

Easily convert between decimal and hexadecimal numbers in Vim/Neovim using built-in functions and custom commands

Example: Example (Vim)

```
" Convert decimal to hex
function! Dec2hex(arg)
  return printf('%x', a:arg + 0)
endfunction

" Convert hex to decimal
function! Hex2dec(arg)
  return (a:arg =~? '^0x') ? a:arg + 0 : ('0x'.a:arg) + 0
endfunction
```

Example: Example (Neovim)

```
-- Convert decimal to hex
function _G.dec_to_hex(arg)
  return string.format('%x', tonumber(arg))
end

-- Convert hex to decimal
function _G.hex_to_dec(arg)
  return tonumber(arg:match('^0x') and arg or '0x' .. arg)
end
```

Source: vim.fandom.com

11.13 Convert Between Hex and Decimal Quickly

Category: clever_tricks
Tags: conversion, number-manipulation, command-line

Provides multiple methods to convert between hex and decimal numbers in Vim/Neovim, including command-line and mapping-based approaches

Example: Example (Vim)

```
" Convert decimal to hex
command! -nargs=? -range Dec2hex call s:Dec2hex(<line1>, <line2>, '<args>')
function! s:Dec2hex(line1, line2, arg) range
  if empty(a:arg)
    let cmd = 's/\<\d\+\>/\=printf("0x%x",submatch(0)+0)/g'
    execute a:line1 . ',' . a:line2 . cmd
  else
    echo printf('%x', a:arg + 0)
  endif
endfunction
```

Example: Example (Neovim)

```
-- Convert decimal to hex
vim.api.nvim_create_user_command('Dec2hex', function(opts)
  local arg = opts.args
  local range = {opts.line1, opts.line2}

  if arg == '' then
    local cmd = 's/\d\+/\=printf("0x%x", submatch(0) + 0)/g'
    vim.cmd(range[1] .. ',' .. range[2] .. cmd)
  else
    print(string.format('%x', tonumber(arg)))
  end
end, { nargs = '?', range = true })
```

Source: vim.fandom.com

11.14 Create Comment Boxes with External Utility

Category: clever_tricks
Tags: text-formatting, external-tools, comments

Use the 'boxes' utility to automatically create formatted comment boxes around text in code

Example: Example (Vim)

```
" Map F2 to create comment boxes for selected text
vmap <F2> !boxes -s 80 <CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for creating comment boxes
vim.keymap.set('v', '<F2>', '!boxes -s 80<CR>', { desc = 'Create comment
↪  box' })
```

Source: vim.fandom.com

11.15 Create HTML Links with Automatic Title Fetch

Category: clever_tricks
Tags: html, web-development, automation

Automatically generate HTML anchor tags by fetching page titles from URLs, saving time when creating links

Example: Example (Vim)

```
" Prompt user to enter URL and optional TARGET.
nnoremap <Leader>al :call AddLink()<CR>
function! AddLink()
  let url = input('URL to add? ')
  if empty(url)
    return
  endif
  let target = input('Target for this link? ')
  if !empty(target)
    let target = ' target="' . target . '"'
  endif
  let html = system('wget -q -O - ' . shellescape(url))
  let regex = '\c.*head.*<title[^>]*>\_s*\zs.\_*\ze\_s*<\/title>'
  let title = substitute(matchstr(html, regex), "\n", ' ', 'g')
  if empty(title)
    let title = 'Unknown'
  endif
  put ='<a href="' . url . '"' . target . '>' . title . '</a>'
endfunction
```

Example: Example (Neovim)

```
function _G.add_html_link()
  local url = vim.fn.input('URL to add? ')
  if url == '' then return end

  local target = vim.fn.input('Target for this link? ')
  local target_attr = target ~= '' and ' target="' .. target .. '"' or ''

  local html = vim.fn.system('wget -q -O - ' .. vim.fn.shellescape(url))
  local title = html:match('<title[^>]*>%s*(.-)%s*</title>') or 'Unknown'
  title = title:gsub('
', ' ')
```

```
  local link = string.format('<a href="%s"%s>%s</a>', url, target_attr,
  ↪  title)
  vim.api.nvim_put({link}, 'l', true, false)
end

-- Map the function
vim.keymap.set('n', '<leader>al', _G.add_html_link, { desc = 'Add HTML link
↪  with title' })
```

Source: vim.fandom.com

11.16 Create Personal Memo File with Vim Help-Style Links

Category: clever_tricks
Tags: documentation, note-taking, custom-markup

Create a personal memo file using Vim help-style markup, enabling easy navigation and organization of notes with internal links and tags

Example: Example (Vim)

```
" Memo file example with Vim help-style markup
*memo.txt* My Personal Memo

*subjects* *toc*
|cmd| Command Tips |dev| Developments

*cmd* Commands Section
*General* General Commands

" Vim settings for help-like file
vim:tw=78:fo=tcq2:isk=!-~,^*,^\|,^":ts=8:ft=help:norl:
```

Example: Example (Neovim)

```
-- Lua approach for creating a memo file
-- Use vim.o and vim.opt to set file-specific settings
vim.o.filetype = 'help'
vim.opt.iskeyword:append({'!-~', '^*', '^\|', '^"'})

-- Example of creating a memo buffer
vim.api.nvim_create_buf(true, false)
vim.api.nvim_buf_set_lines(0, 0, -1, false, {
  '*memo.txt* My Personal Memo',
  '',
  '*subjects* *toc*',
  '|cmd| Command Tips |dev| Developments',
  '',
  '*cmd* Commands Section',
  '*General* General Commands'
})
```

Source: vim.fandom.com

11.17 Create Postfix Abbreviations for Code Snippets

Category: clever_tricks
Tags: abbreviations, code-generation, productivity

Use Vim abbreviations to quickly generate code snippets by typing a word followed by an abbreviation trigger

Example: Example (Vim)

```
:ab ff <Esc>^d$ifor(int i=0;i<<Esc>pi.length;i++){<CR><CR>}//end for loop
↪  over array <Esc>pi[i]<Esc>==k==k==ji<Tab>
```

Example: Example (Neovim)

```
-- Note: In Neovim, you can use vim.cmd to set abbreviations
vim.cmd('ab ff <Esc>^d$ifor(int i=0;i<<Esc>pi.length;i++){<CR><CR>}//end for
↪  loop over array <Esc>pi[i]<Esc>==k==k==ji<Tab>')
```

Source: vim.fandom.com

11.18 Create Text Abbreviations for Frequent Typos

Category: clever_tricks
Tags: abbreviations, typing, autocorrect

Automatically correct common typos or expand shorthand text while typing in insert mode

Example: Example (Vim)

```
iabbrev teh the
iabbrev seperate separate
iabbrev #i #include
iabbrev #d #define
```

Example: Example (Neovim)

```
vim.cmd('iabbrev teh the')
vim.cmd('iabbrev seperate separate')
vim.cmd('iabbrev #i #include')
vim.cmd('iabbrev #d #define')
```

Source: vim.fandom.com

11.19 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: Example (Vim)

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

Source: Community contributed

11.20 Dynamic Mail Signature Rotation in Vim

Category: clever_tricks
Tags: email, scripting, key-mapping

Automatically rotate mail signatures by replacing the existing signature with a new one from a quotes file

Example: Example (Vim)

```
function! RotateSig()
  normal! mQG
  execute '?^-- $'
  execute ':nohl'
  normal! o<Esc>
  normal! dG
  normal! <CR>
  execute 'r !~/bin/autosig ~/.quotes \%'
  normal! `Q
endfunction

" Mapping
nnoremap ,r :call RotateSig()<CR>
```

Example: Example (Neovim)

```
function _G.rotate_signature()
  -- Mark current position
  vim.api.nvim_command('normal! mQ')

  -- Go to signature marker
  vim.api.nvim_command('?^-- $')
  vim.api.nvim_command('nohlsearch')
```

```
  -- Add new line and delete existing signature
  vim.api.nvim_command('normal! o')
  vim.api.nvim_command('normal! dG')

  -- Insert new signature from external command
  vim.api.nvim_command('read !~/bin/autosig ~/.quotes %')

  -- Return to original position
  vim.api.nvim_command('normal! `Q')
end

-- Key mapping
vim.keymap.set('n', ',r', _G.rotate_signature, { desc = 'Rotate email
↪  signature' })
```

Source: vim.fandom.com

11.21 Dynamic Page Count for Hardcopy Printing

Category: clever_tricks
Tags: printing, formatting, hardcopy

Automatically calculate total pages when printing, accounting for varying lines per page

Example: Example (Vim)

```
function! PH_Multiple()
  let lpp = "73" " lpp - lines per page
  let modulo = line('$') % lpp
  if modulo != 0
    return ( line('$') / lpp ) + 1
  else
    return line('$') / lpp
  endif
endfunction
set printheader=%<%f%h%m%=Page\ %N\ of\ %{PH_Multiple()}
```

Example: Example (Neovim)

```
function PrintPageCount()
  local lpp = 73 -- lines per page
  local total_lines = vim.fn.line('$')
  local pages = math.floor(total_lines / lpp)

  if total_lines % lpp ~= 0 then
    pages = pages + 1
  end

  return pages
end
```

```
vim.opt.printheader = '%<%f%h%m%=Page %N of ' ..
↪  vim.fn.eval('PrintPageCount()')
```

Source: vim.fandom.com

11.22 Dynamic ZIP Code Lookup in Vim

Category: clever_tricks
Tags: web-lookup, external-command, utility

Lookup US ZIP code city and state directly from Vim using TCL and web scraping

Example: Example (Vim)

```
function ZIPLookup (word)
  tcl puts [ZipLookup [::vim::expr a:word]]
endfunction

menu &Utilities.&ZipLookup :call ZIPLookup(expand("<cword>")) <CR>
```

Example: Example (Neovim)

```
-- Lua equivalent requires external plugin or custom function
-- Could be implemented using vim.fn.system() and a web request library
function _G.zip_lookup()
  local zipcode = vim.fn.expand('<cword>')
  -- Implement web request and parsing in Lua
  -- Example pseudo-code, actual implementation would require more robust
  ↪  handling
  local result = vim.fn.system('curl -s
  ↪  https://zip4.usps.com/zip4/zip_responseA.jsp?zipcode=' .. zipcode)
  vim.notify(result, vim.log.levels.INFO)
end

-- Set up keymap
vim.keymap.set('n', '<leader>zl', _G.zip_lookup, { desc = 'Lookup ZIP code'
↪  })
```

Source: vim.fandom.com

11.23 Efficient Debug Print Statements in C++

Category: clever_tricks
Tags: debugging, key-mapping, language-specific

Streamline creating print statements by remapping keys to reduce keystrokes and shift key usage when debugging C++ code

Example: Example (Vim)

```
function! CppSetupCout()
  inoremap , <Space><<
  inoremap < ,
  inoremap ' "
  inoremap " '
  imap ; <Esc>:call CppResetCout()<CR>a;
endfunction
```

Example: Example (Neovim)

```
function _G.cpp_setup_cout()
  vim.keymap.set('i', ',', '<Space><<', { buffer = true })
  vim.keymap.set('i', '<', ',', { buffer = true })
  vim.keymap.set('i', "'", '"', { buffer = true })
  vim.keymap.set('i', '"', "'", { buffer = true })
end

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'cpp',
  callback = _G.cpp_setup_cout
})
```

Source: vim.fandom.com

11.24 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: Example (Vim)

```
" Vimscript:
nnoremap <leader>. .`[

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

Example: Example (Neovim)

```
-- Lua:
vim.keymap.set('n', '<leader>.', '.`[', { desc = 'Repeat change and return
↪  to start' })
```

Source: Community contributed

11.25 Evaluate Expressions Across Vim Modes

Category: clever_tricks
Tags: expression-evaluation, key-mapping, calculator

Use Ctrl-R= to evaluate expressions in insert, command, and normal modes, enabling quick calculations and dynamic value insertion

Example: Example (Vim)

```
" Map Ctrl-R= to work in normal mode for expression evaluation
map <CTRL-R>= :echo
```

Example: Example (Neovim)

```
-- Lua equivalent for mapping expression evaluation
vim.keymap.set({'n', 'i', 'c'}, '<C-r>=', function()
  local expr = vim.fn.input('Expression: ')
  return vim.fn.eval(expr)
end, { expr = true })
```

Source: vim.fandom.com

11.26 Evaluate Python Expressions in Vim

Category: clever_tricks
Tags: python, scripting, evaluation

Provides a flexible way to evaluate Python expressions directly within Vim, allowing complex calculations and transformations of current line content

Example: Example (Vim)

```
python << EOL
import vim

def EvaluateCurrentLine(*args):
  cur_str = vim.current.line
  action, symb = None, None
  for i in args:
    if i in ["r","p"]: action = i
    else: symb = i
  try: start = cur_str.rindex(symb)+len(symb)
  except: start = 0
  result = eval(cur_str[start:],globals())
  if action == "r":
    vim.current.line = cur_str[:start]+str(result)
  else:
    print result
EOL
```

```
command -narg=* PyEv python EvaluateCurrentLine(<f-args>)
```

Example: Example (Neovim)

```
function _G.evaluate_current_line(...)
  local cur_str = vim.api.nvim_get_current_line()
  local action, symb = nil, nil
  for _, arg in ipairs({...}) do
    if arg == 'r' or arg == 'p' then
      action = arg
    else
      symb = arg
    end
  end

  local start = 0
  if symb then
    local idx = cur_str:reverse():find(symb:reverse())
    start = idx and (#cur_str - idx + 1) or 0
  end

  local ok, result = pcall(load('return ' .. cur_str:sub(start + 1)))
  if ok then
    if action == 'r' then
      vim.api.nvim_set_current_line(cur_str:sub(1, start) ..
      ↪  tostring(result))
    else
      print(result)
    end
  end
end

vim.api.nvim_create_user_command('PyEv', function(opts)
  _G.evaluate_current_line(unpack(opts.fargs))
end, { nargs = '*' })
```

Source: vim.fandom.com

11.27 Execute Accidentally Inserted Normal Mode Commands

Category: clever_tricks
Tags: insert-mode, mapping, undo

Allows executing normal mode commands accidentally typed in insert mode by quickly undoing and executing the text as a command

Example: Example (Vim)

```
inoremap <somekey> <Esc>u@.

autocmd CursorHoldI * call feedkeys("\<C-G>u", 'tn')
```

Example: Example (Neovim)

```
-- Lua equivalent
vim.keymap.set('i', '<somekey>', '<Esc>u@.', { desc = 'Undo and execute as
↪  command' })

vim.api.nvim_create_autocmd('CursorHoldI', {
  callback = function()
    vim.fn.feedkeys('\<C-G>u', 'tn')
  end
})
```

Source: vim.fandom.com

11.28 Execute Python Code Inside Vim Buffer

Category: clever_tricks
Tags: python, scripting, buffer-manipulation

Execute Python code directly within a Vim buffer, replacing the selected lines with the output or performing in-place transformations

Example: Example (Vim)

```
python << EOL
import vim, StringIO,sys
def PyExecReplace(line1,line2):
  r = vim.current.buffer.range(int(line1),int(line2))
  redirected = StringIO.StringIO()
  sys.stdout = redirected
  exec('\n'.join(r) + '\n')
  sys.stdout = sys.__stdout__
  output = redirected.getvalue().split('\n')
  r[:] = output[:-1]
  redirected.close()
EOL
command -range Pyer python PyExecReplace(<f-line1>,<f-line2>)
```

Example: Example (Neovim)

```
local function py_exec_replace(line1, line2)
  local buffer = vim.api.nvim_get_current_buf()
  local lines = vim.api.nvim_buf_get_lines(buffer, line1-1, line2, false)
  local code = table.concat(lines, '\n')

  local output = vim.fn.system('python3 -c "' .. code .. '"')

  vim.api.nvim_buf_set_lines(buffer, line1-1, line2, false,
  ↪  vim.split(output, '\n'))
end

vim.api.nvim_create_user_command('Pyer', function(opts)
```

```
  py_exec_replace(opts.line1, opts.line2)
end, { range = true })
```

Source: vim.fandom.com

11.29 Extract Phone Number Under Cursor

Category: clever_tricks
Tags: text-extraction, function, cursor-position

Create a function to extract and clean phone numbers from text, with optional error handling

Example: Example (Vim)

```
function! CPhone(check)
  let s = '[-+./()0-9 ]*'
  let nr = matchstr(getline('.'), '\s*\zs'.s.'\%'.col('.').'c'.s)
  let nr = substitute(nr, '\s\+$', '', '')
  if a:check && empty(nr)
    throw 'No phone number under cursor.'
  endif
  return nr
endfunction
```

Example: Example (Neovim)

```
function CPhone(check)
  local s = '[-+./()0-9 ]*'
  local line = vim.fn.getline('.')
  local col = vim.fn.col('.')
  local nr = vim.fn.matchstr(line, '\s*\zs'..s..'\%'..col..'c'..s)
  nr = nr:gsub('%s+$', '')

  if check and nr == '' then
    error('No phone number under cursor')
  end

  return nr
end
```

Source: vim.fandom.com

11.30 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: Example (Vim)

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

Example: Example (Neovim)

```
-- Lua:
vim.opt.statusline = '%f [%{&fileencoding?&fileencoding:&encoding}] %y'
-- Shows filename, encoding, and filetype
```

Source: Community contributed

11.31 Fix Syntax Highlighting with F12 Mapping

Category: clever_tricks
Tags: syntax-highlighting, key-mapping, performance

Create a quick mapping to resync syntax highlighting when it breaks, improving editor responsiveness

Example: Example (Vim)

```
noremap <F12> <Esc>:syntax sync fromstart<CR>
inoremap <F12> <C-o>:syntax sync fromstart<CR>
```

Example: Example (Neovim)

```
vim.keymap.set({'n', 'i'}, '<F12>', ':syntax sync fromstart<CR>', { desc =
↪  'Resync syntax highlighting' })
```

Source: vim.fandom.com

11.32 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: Example (Vim)

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

Source: Community contributed

11.33 G-commands - display command output

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

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

Example: Example (Vim)

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

Source: Community contributed

11.34 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: Example (Vim)

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

Source: Community contributed

11.35 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: Example (Vim)

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

Source: Community contributed

11.36 G-commands - join without space

Category: Clever Tricks
Tags: join, line, space

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

Example: Example (Vim)

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

Source: Community contributed

11.37 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: Example (Vim)

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

Source: Community contributed

11.38 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: Example (Vim)

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

Source: Community contributed

11.39 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: Example (Vim)

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

Source: Community contributed

11.40 G-commands - repeat substitute

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

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

Example: Example (Vim)

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

Source: Community contributed

11.41 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: Example (Vim)

```
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
```

Source: Community contributed

11.42 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: Example (Vim)

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

Source: Community contributed

11.43 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: Example (Vim)

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

Source: Community contributed

11.44 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: Example (Vim)

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

Source: Community contributed

11.45 G-commands - sleep

Category: Clever Tricks
Tags: sleep, delay, pause

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

Example: Example (Vim)

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

Source: Community contributed

11.46 G-commands - undo branches

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

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

Example: Example (Vim)

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

Source: Community contributed

11.47 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: Example (Vim)

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

Source: Community contributed

11.48 Generate Unicode Character Table

Category: clever_tricks
Tags: unicode, character-display, text-manipulation

Create a comprehensive table of Unicode characters within a specified range, useful for exploring character sets and font support

Example: Example (Vim)

```
function! GenerateUnicode(first, last)
  let i = a:first
  while i <= a:last
    if (i%256 == 0)
      $put ='-----------------------------------------------------'
      $put ='     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F '
      $put ='-----------------------------------------------------'
    endif
    let c = printf('%04X ', i)
```

```
    for j in range(16)
      let c = c . nr2char(i) . ' '
      let i += 1
    endfor
    $put =c
  endwhile
endfunction
```

Example: Example (Neovim)

```
function _G.generate_unicode(first, last)
  local buffer = vim.api.nvim_create_buf(true, true)
  vim.api.nvim_set_current_buf(buffer)

  local i = first
  while i <= last do
    if (i % 256 == 0) then
      vim.api.nvim_buf_set_lines(buffer, -1, -1, false, {
        '-----------------------------------------------',
        '     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F ',
        '-----------------------------------------------'
      })
    end

    local line = string.format('%04X ', i)
    for j = 1, 16 do
      line = line .. string.char(i) .. ' '
      i = i + 1
    end

    vim.api.nvim_buf_set_lines(buffer, -1, -1, false, {line})
  end
end

-- Usage: :lua generate_unicode(0x9900, 0x9fff)
```

Source: vim.fandom.com

11.49 Group Lines with Matching Column Pattern

Category: clever_tricks
Tags: line-manipulation, search, text-processing

Insert blank lines between groups of lines that match a specific pattern at a given column position. Useful for organizing and visually separating similar data groups.

Example: Example (Vim)

```
function! GroupMatchingLines()
  let pattern = strpart(getline('.'), col('.') - 1, strlen(@@))
  if !empty(pattern)
```

```
    let pattern = escape(pattern, '\.*$^~[')
    let col_cursor = col('.')
    let pattern = '\%'.col_cursor.'c'.'\('.pattern.'\)\@!'
    let [rowm,colm] = searchpos(pattern,'W')
    if rowm > 0
      '.-1put ='''
      exe 'normal! j'.col_cursor.'|'
      return
    endif
  endif
  echo 'String not found'
endfunction

nnoremap <F9> :call GroupMatchingLines()<CR>
```

Example: Example (Neovim)

```
function _G.group_matching_lines()
  local pattern = vim.fn.strpart(vim.fn.getline('.'), vim.fn.col('.') - 1,
  ↪  #vim.fn.getreg('"'))
  if pattern ~= '' then
    pattern = vim.fn.escape(pattern, '\.*$^~[')
    local col_cursor = vim.fn.col('.')
    local search_pattern = string.format('\%%dc\(\%s\)\@!', pattern)
    local result = vim.fn.searchpos(search_pattern, 'W')

    if result[1] > 0 then
      vim.cmd('.-1put =""')
      vim.cmd(string.format('normal! j%d|', col_cursor))
      return
    end
  end
  print('String not found')
end

vim.keymap.set('n', '<F9>', _G.group_matching_lines, { desc = 'Group
↪  matching lines' })
```

Source: vim.fandom.com

11.50 Highlight Current Word for Easy Cursor Tracking

Category: clever_tricks
Tags: cursor-tracking, highlight, navigation

Toggle highlighting of the word under the cursor to help track cursor position in large files

Example: Example (Vim)

```
nnoremap <C-K> :call HighlightNearCursor()<CR>
function HighlightNearCursor()
  if !exists("s:highlightcursor")
    match Todo /\k*\%#\k*/
    let s:highlightcursor=1
  else
    match None
    unlet s:highlightcursor
  endif
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-K>', function()
  if not vim.g.highlightcursor then
    vim.cmd('match Todo /\\k*\\%#\\k*/')
    vim.g.highlightcursor = true
  else
    vim.cmd('match None')
    vim.g.highlightcursor = nil
  end
end, { desc = 'Toggle cursor word highlight' })
```

Source: vim.fandom.com

11.51 Identify Syntax Highlighting Group Under Cursor

Category: clever_tricks
Tags: syntax, debugging, highlight

Create a mapping to reveal the syntax highlighting groups at the current cursor position, which is useful for understanding and customizing syntax highlighting

Example: Example (Vim)

```
nnoremap <F10> :echo "hi<" . synIDattr(synID(line(".“),col(".“),1),"name") .
↪  '> trans<'
\ . synIDattr(synID(line(".“),col(".“),0),"name") . "> lo<"
\ . synIDattr(synIDtrans(synID(line(".“),col(".“),1)),"name") . ">"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F10>', function()
  local line = vim.fn.line('.')
  local col = vim.fn.col('.')
  local synID = vim.fn.synID(line, col, 1)
  local transID = vim.fn.synID(line, col, 0)
  local loID = vim.fn.synIDtrans(synID)
```

```
  print(string.format(
    "hi<%s> trans<%s> lo<%s>",
    vim.fn.synIDattr(synID, 'name'),
    vim.fn.synIDattr(transID, 'name'),
    vim.fn.synIDattr(loID, 'name')
  ))
end, { desc = 'Show syntax highlighting groups' })
```

Source: vim.fandom.com

11.52 Inline Visual Mode Calculator

Category: clever_tricks
Tags: calculation, visual-mode, command-line

Perform mathematical calculations directly in Vim using visual mode and bc calculator, with options to replace or append results

Example: Example (Vim)

```
function MyCalc(str)
  if exists("g:MyCalcRounding")
    return system("echo 'x=" . a:str . ";d=.5/10^" . g:MyCalcPresition
          \. ";if (x<0) d=-d; x+=d; scale=" . g:MyCalcPresition . ";print
          ↪  x/1' | bc -l")
  else
    return system("echo 'scale=" . g:MyCalcPresition . " ; print " . a:str .
    ↪  "' | bc -l")
  endif
endfunction

" Control precision and rounding
let g:MyCalcPresition = 2
let g:MyCalcRounding = 1

" Replace current line or visual block with calculation result
map <silent> <Leader>c :s/.*/\=MyCalc(submatch(0))/<CR>:noh<CR>
vmap <silent> <Leader>c :B s/.*/\=MyCalc(submatch(0))/<CR>:noh<CR>
```

Example: Example (Neovim)

```
local function my_calc(str)
  local precision = vim.g.MyCalcPresition or 2
  local cmd = string.format("echo 'scale=%d; print %s' | bc -l", precision,
  ↪  str)
  local result = vim.fn.system(cmd):gsub("\n", "")
  return result
end

-- Calculator mappings
vim.keymap.set({'n', 'v'}, '<leader>c', function()
```

```
  local mode = vim.fn.mode()
  if mode == 'v' or mode == 'V' or mode == '\22' then
    local start_pos = vim.fn.getpos("'<")
    local end_pos = vim.fn.getpos("'>")
    local lines = vim.fn.getline(start_pos[2], end_pos[2])

    for i, line in ipairs(lines) do
      local calc_result = my_calc(line)
      vim.fn.setline(start_pos[2] + i - 1, calc_result)
    end
  else
    local line = vim.fn.getline('.')
    local calc_result = my_calc(line)
    vim.fn.setline('.', calc_result)
  end
end, { desc = 'Calculator: Replace line/selection with result' })
```

Source: vim.fandom.com

11.53 Insert Current Directory Name Quickly

Category: clever_tricks
Tags: command-line, insert-mode, productivity

Multiple ways to quickly insert the current working directory name in your buffer

Example: Example (Vim)

```
" Method 1: Using external command
!!pwd

" Method 2: Using getcwd() function
:inoremap \fp <C-R>=getcwd()<CR>
```

Example: Example (Neovim)

```
-- Method 1: Using external command in normal mode
vim.cmd('!!pwd')

-- Method 2: Create a key mapping to insert current directory
vim.keymap.set('i', '<leader>fp', function()
  local cwd = vim.fn.getcwd()
  vim.api.nvim_put({cwd}, '', true, true)
end, { desc = 'Insert current directory path' })
```

Source: vim.fandom.com

11.54 Insert Filename as Classname in Insert Mode

Category: clever_tricks
Tags: insert-mode, abbreviation, filename

Quickly insert the current filename (without path/extension) while in insert mode using an abbreviation

Example: Example (Vim)

```
" Insert classname (filename minus path and extension).
iab <buffer> <unique> ,f <C-r>=expand('%:t:r')<CR>
```

Example: Example (Neovim)

```
-- Create buffer-local abbreviation to insert filename
vim.cmd('iab <buffer> <unique> ,f ' .. vim.fn.expand('%:t:r'))
```

Source: vim.fandom.com

11.55 Insert Numbered Lists Easily

Category: clever_tricks
Tags: list-generation, range, formatting

Generate numbered lists or sequences with built-in Vim range and map functions

Example: Example (Vim)

```
:put =range(11,15)

" Format numbers with leading zeros
:put =map(range(1,150), 'printf(''%04d'', v:val)')
```

Example: Example (Neovim)

```
-- Insert numbered list
vim.cmd.put(vim.fn.range(11, 15))

-- Format numbers with leading zeros
vim.cmd.put(vim.fn.map(vim.fn.range(1, 150), function(_, val)
  return string.format('%04d', val)
end))
```

Source: vim.fandom.com

11.56 Inspect Character Class in Vim

Category: clever_tricks
Tags: character-classes, text-inspection, regex

Display all characters matching a specific character class, showing their character and ASCII value

Example: Example (Vim)

```
fun! s:Show()
  norm! viwy
  echo 'class [:' . @" . ':]' . "\n"
  let pat = '[[:' . @" . ':]]'
  let i = 0
  while i < 256
    let ch = nr2char(i)
    if ch =~ pat | echon ch . '(' . i . ')' . "\t" | endif
    let i = i + 1
  endwhile
endfun
nn cc :call <SID>Show()<Cr>
```

Example: Example (Neovim)

```
function _G.show_character_class()
  vim.cmd('normal! viwy')
  local class = vim.fn.getreg('"')
  print('class [:' .. class .. ':]\')

  local pattern = '[[:' .. class .. ':]]'

  for i = 0, 255 do
    local ch = vim.fn.nr2char(i)
    if ch:match(pattern) then
      io.write(ch .. '(' .. i .. ')\t')
    end
  end
end

vim.keymap.set('n', 'cc', _G.show_character_class, { desc = 'Show character
↪  class details' })
```

Source: vim.fandom.com

11.57 Launch Multiple Vim Instances in Windows

Category: clever_tricks
Tags: windows, multi-instance, launcher

Use Rundll32.exe to launch multiple Vim instances reliably in Windows, bypassing single-

instance limitations

Example: Example (Vim)

```
C:\WINDOWS\system32\rundll32.exe Shell32.dll,ShellExec_RunDLL "C:\Program
↪  Files\Vim\vim72\gvim.exe"
```

Example: Example (Neovim)

```
-- For Neovim, use vim.fn.system() to launch
vim.fn.system('rundll32.exe Shell32.dll,ShellExec_RunDLL "nvim"')
```

Source: vim.fandom.com

11.58 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: Example (Vim)

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

Source: Community contributed

11.59 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: Example (Vim)

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

Source: Community contributed

11.60 Make Echo Persistent in Vim Scripts

Category: clever_tricks
Tags: scripting, messages, autocmd

Ensures echo messages remain visible by using CursorHold autocmd to prevent disappearing messages in Vim scripts

Example: Example (Vim)

```
func! PersistentEcho(msg)
  echo a:msg
  let g:PersistentEcho=a:msg
endfun

let g:PersistentEcho=''
if &ut>200|let &ut=200|endif
au CursorHold * if PersistentEcho!=''|echo PersistentEcho|let
↪  PersistentEcho=''|endif
```

Example: Example (Neovim)

```
local function persistent_echo(msg)
  vim.api.nvim_echo({{msg, 'Normal'}}, false, {})
  vim.g.persistent_echo = msg
end

vim.g.persistent_echo = ''

-- Set up autocmd to display persistent message
vim.api.nvim_create_autocmd('CursorHold', {
  callback = function()
    if vim.g.persistent_echo ~= '' then
      vim.api.nvim_echo({{vim.g.persistent_echo, 'Normal'}}, false, {})
      vim.g.persistent_echo = ''
    end
  end
})
```

Source: vim.fandom.com

11.61 Obscure Vim Command Obfuscation Trick

Category: clever_tricks
Tags: vim-tricks, command-manipulation, text-transformation

Showcase of complex text manipulation and transformation using Vim's powerful command chaining and substitution capabilities

Example: Example (Vim)

```
vim -c ":%s%s*%Cyrnfr)fcbafbe[Oenz(Zbbyranne%|:%s)[[()])-)Ig|norm Vg?"
```

Example: Example (Neovim)

```
-- Lua equivalent demonstrates text transformation
-- Actual implementation would depend on specific transformation intent
vim.cmd([[%s%s*%Cyrnfr)fcbafbe[Oenz(Zbbyranne%]])
vim.cmd([[%s)[[()])-)Ig]])
vim.cmd('norm Vg?')
```

Source: vim.fandom.com

11.62 Open URLs from Current Line in Browser

Category: clever_tricks
Tags: url-handling, external-command, productivity

Extract and open URLs from the current line in a web browser using a custom function

Example: Example (Vim)

```
function! Browser()
  let line = matchstr(getline("."), '[a-z]*://[^ >,;:]*')
  if line != ""
    exec "!xdg-open " . line
  endif
endfunction
map <Leader>w :call Browser()<CR>
```

Example: Example (Neovim)

```
function _G.open_url_from_line()
  local line = vim.fn.matchstr(vim.fn.getline('.'), '[a-z]*://[^ >,;:]*')
  if line ~= "" then
    vim.fn.system({'xdg-open', line})
  end
end

vim.keymap.set('n', '<Leader>w', _G.open_url_from_line, { desc = 'Open URL
↪  from current line' })
```

Source: vim.fandom.com

11.63 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: Example (Vim)

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

Source: Community contributed

11.64 Persistent Echo Messages in Vim

Category: clever_tricks
Tags: scripting, messages, autocmd

Ensure echo messages are visible and don't disappear quickly, especially useful for script debugging and user notifications

Example: Example (Vim)

```
func! PersistentEcho(msg)
  echo a:msg
  let g:PersistentEcho=a:msg
endfun
let g:PersistentEcho=''
if &ut>200|let &ut=200|endif
au CursorHold * if PersistentEcho!=''|echo PersistentEcho|let
↪  PersistentEcho=''|endif
```

Example: Example (Neovim)

```
local function persistent_echo(msg)
  vim.api.nvim_echo({{msg, 'Normal'}}, false, {})
  vim.g.persistent_echo = msg
end

vim.o.updatetime = math.min(vim.o.updatetime, 200)

vim.api.nvim_create_autocmd('CursorHold', {
  callback = function()
    if vim.g.persistent_echo and vim.g.persistent_echo ~= '' then
      vim.api.nvim_echo({{vim.g.persistent_echo, 'Normal'}}, false, {})
      vim.g.persistent_echo = ''
    end
  end
})
```

Source: vim.fandom.com

11.65 Platform-Specific URL Opening

Category: clever_tricks
Tags: cross-platform, url-handling, external-command

Platform-specific methods to open URLs using system commands

Example: Example (Vim)

```
" Mac OS X
function! HandleURI()
  let s:uri = matchstr(getline("."), '[a-z]*://[^ >,;:]*')
  if s:uri != ""
    exec "!open " . s:uri
  endif
endfunction

" Linux
nnoremap <leader>w :silent !xdg-open <C-R>=escape("<C-R><C-F>",
↪  "#?&;|%")<CR><CR>
```

Example: Example (Neovim)

```
-- Mac OS X
function _G.mac_open_uri()
  local uri = vim.fn.matchstr(vim.fn.getline('.'), '[a-z]*://[^ >,;:]*')
  if uri ~= "" then
    vim.fn.system({'open', uri})
  end
end

-- Linux
vim.keymap.set('n', '<Leader>w', function()
  local url = vim.fn.expand('<cfile>')
  vim.fn.system({'xdg-open', url})
end, { desc = 'Open URL under cursor' })
```

Source: vim.fandom.com

11.66 Preserve Cursor Position During Script Undo

Category: clever_tricks
Tags: cursor-position, undo, scripting

Ensure cursor returns to original position after undoing script-made changes by creating a 'fake' text modification

Example: Example (Vim)

```
" In a function before making changes
execute "normal! i "
execute "normal! a\<BS>"
```

Example: Example (Neovim)

```
-- Lua equivalent for preserving cursor position
local function preserve_cursor_on_undo()
  vim.cmd('normal! i ')
  vim.cmd('normal! a\<BS>')
end
```

Source: vim.fandom.com

11.67 Preserve Search Register When Running Commands

Category: clever_tricks
Tags: search, command-preservation, register-management

Create a function that executes commands without modifying the search register, which is useful for preserving search history during operations

Example: Example (Vim)

```
function! SafeSearchCommand(line1, line2, theCommand)
  let search = @/
  execute a:line1 . "," . a:line2 . a:theCommand
  let @/ = search
endfunction

com! -range -nargs=+ SS call SafeSearchCommand(<line1>, <line2>, <q-args>)
com! -range -nargs=* S call SafeSearchCommand(<line1>, <line2>, 's' .
↪  <q-args>)
```

Example: Example (Neovim)

```
function _G.safe_search_command(line1, line2, command)
  local search = vim.fn.getreg('/')
  vim.cmd(string.format('%d,%d%s', line1, line2, command))
  vim.fn.setreg('/', search)
end

vim.api.nvim_create_user_command('SS', function(opts)
  _G.safe_search_command(opts.line1, opts.line2, opts.args)
end, { range = true, nargs = '+' })

vim.api.nvim_create_user_command('S', function(opts)
  _G.safe_search_command(opts.line1, opts.line2, 's' .. opts.args)
```

```
end, { range = true, nargs = '*' })
```

Source: vim.fandom.com

11.68 Quick Hex and Decimal Number Conversion

Category: clever_tricks
Tags: number-conversion, command-line, conversion

Easily convert between hex and decimal numbers using built-in Vim functions and commands

Example: Example (Vim)

```
" Convert decimal to hex
:echo printf('%x', 74565)

" Convert hex to decimal
:echo 0x12345

" In insert mode, convert numbers
" Ctrl-R = 0x09ab Enter (inserts 2475)
" Ctrl-R = printf('0x%04x',2475) Enter (inserts 0x09ab)
```

Example: Example (Neovim)

```
-- Convert decimal to hex
print(string.format('%x', 74565))

-- Convert hex to decimal
print(tonumber('0x12345'))

-- In Neovim, use vim.fn to mimic Vim's conversion
-- Example: printf equivalent
print(vim.fn.printf('%x', 74565))
print(vim.fn.printf('0x%04x', 2475))
```

Source: vim.fandom.com

11.69 Quick LaTeX Environment Creation in Vim

Category: clever_tricks
Tags: latex, text-insertion, custom-commands

Create custom commands to easily insert LaTeX environments with tab completion and optional insert mode functionality

Example: Example (Vim)

```
command -complete=custom,ListE -nargs=1 -range E normal <line1>ggO\begin{<a
↪  rgs>}<Esc><line2>ggjo\end{<args>}<Esc><line1>ggv<line2>ggjj=
command -complete=custom,ListE -nargs=1 Ei execute "normal
↪  \<Esc>i\begin{<args>}\<CR>\end{<args>}<Esc>O<Space>" | startinsert

function ListE(A,L,P)
  return "align*\nenumerate\nitemize\nfigure\ntabular\nbmatrix\npmatrix\nca
  ↪  ses\n\ndocument\narray\nproof"
endfunction

" Optional insert mode mapping
imap <buffer> <C-Space> <Esc>:Ei<Space>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('E', function(opts)
  local env = opts.args
  local line1 = vim.fn.line('.')
  vim.cmd('normal! O\\begin{' .. env .. '}\<Esc>')
  vim.cmd('normal! j o\\end{' .. env .. '}\<Esc>')
  vim.cmd('normal! ' .. line1 .. 'ggv' .. (line1+1) .. 'ggjj=')
end, {
  nargs = 1,
  complete = function()
    return {'align*', 'enumerate', 'itemize', 'figure', 'tabular',
    ↪  'bmatrix', 'pmatrix', 'cases', 'document', 'array', 'proof'}
  end
})

-- Insert mode mapping
vim.keymap.set('i', '<C-Space>', '<Esc>:Ei<Space>', { buffer = true })
```

Source: vim.fandom.com

11.70 Quick Number Base Conversion

Category: clever_tricks
Tags: number-conversion, utility, command-line

Use built-in Vim functions to quickly convert numbers between bases in command mode or insert mode

Example: Example (Vim)

```
" Convert hex to decimal in command mode
:echo str2nr('0x0a2f', 16)

" Convert decimal to hex string
:echo printf('%x', 1234)
```

```
" Insert converted number in insert mode
" <C-R>=0x09ab<Enter> inserts 2475
" <C-R>=printf('0x%04x',2475)<Enter> inserts 0x09ab
```

Example: Example (Neovim)

```
-- Convert hex to decimal
print(tonumber('0x0a2f', 16))

-- Convert decimal to hex string
print(string.format('%x', 1234))

-- For insert mode conversion, use vim.fn.input() or lua's string formatting
-- Example in Neovim lua command mode
vim.fn.feedkeys('i' .. string.format('0x%04x', 2475), 'n')
```

Source: vim.fandom.com

11.71 Quick Number Type Detection and Conversion

Category: clever_tricks
Tags: number-conversion, cursor-word, utility

Function to detect and convert numbers under cursor between decimal and hexadecimal, handling various input formats

Example: Example (Vim)

```
function! DecAndHex(number)
  if a:number =~? '^\d\+$'
    let dec = a:number
    echo dec . printf('  ->  0x%X, -(0x%X)', dec, -dec)
  elseif a:number =~? '^0x\x\+$'
    let hex = substitute(a:number, '^0x', '', '')
    echo '0x' . hex . printf('  ->  %d', eval('0x'.hex))
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.dec_and_hex(number)
  local dec = tonumber(number)
  local hex = tonumber(number, 16)

  if dec then
    print(string.format('%d  ->  0x%X, -(0x%X)', dec, dec, -dec))
  elseif hex then
    print(string.format('0x%X  ->  %d', hex, hex))
  else
    print('Not a valid number')
```

```
  end
end

-- Bind to a key
vim.keymap.set('n', 'gn', function()
  local word = vim.fn.expand('<cword>')
  _G.dec_and_hex(word)
end)
```

Source: vim.fandom.com

11.72 Quick Word Count in Vim

Category: clever_tricks
Tags: text-editing, statistics, productivity

Multiple methods to count words in a document, including partial selections and live word count

Example: Example (Vim)

```
" Count words in current buffer
g Ctrl-g

" Count words in selected block
V5j
g Ctrl-g

" Get word count using external command
:w !wc
```

Example: Example (Neovim)

```
-- Note: Word count functionality is built-in
-- Use g Ctrl-g in normal or visual mode

-- Custom word count function for status line
function WordCount()
  local words = vim.fn.wordcount()
  return words.words .. ' words'
end

-- Add to status line
vim.opt.statusline:append('%{v:lua.WordCount()}')
```

Source: vim.fandom.com

11.73 Quick Word Count in Vim/Neovim

Category: clever_tricks
Tags: word-count, buffer-management, text-editing

Multiple ways to count words in a buffer or selected text, including built-in and external methods

Example: Example (Vim)

```
" Count words in current buffer
g Ctrl-g

" Count words in selected block
V5j
g Ctrl-g

" Count words using external wc command
:w !wc
```

Example: Example (Neovim)

```
-- For current buffer word count
-- Use g Ctrl-g in normal mode

-- For external word count
vim.api.nvim_command('w !wc')
```

Source: vim.fandom.com

11.74 Quick number increment

Category: Clever Tricks
Tags: number, increment, math

Use `Ctrl+a` to increment number under cursor, `Ctrl+x` to decrement. Works with decimals and hex.

Example: Example (Vim)

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

Source: Community contributed

11.75 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: Example (Vim)

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

Source: Community contributed

11.76 Quickly Create Perl Subroutines in Vim

Category: clever_tricks
Tags: mapping, text-insertion, productivity

Adds a mapping to create a new Perl subroutine with the word under the cursor, placing it at the end of the file or before special tokens

Example: Example (Vim)

```
nnoremap <Leader>ns :call Newsub()<CR>
function! Newsub()
  let word = "sub " . expand("<cword>") . "{}"
  let ln = search("__.*__", 'nW')
  if ln == 0
    call append('$', word)
  else
    call append(ln-1, word)
  endif
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>ns', function()
  local word = expand('<cword>')
  local line_num = vim.fn.search('__.*__', 'nW')

  if line_num == 0 then
    -- Append to end of file
    vim.fn.append('$', 'sub ' .. word .. '{}')
  else
    -- Insert before special token
    vim.fn.append(line_num - 1, 'sub ' .. word .. '{}')
  end
end, { desc = 'Create new Perl subroutine' })
```

Source: vim.fandom.com

11.77 Quickly Insert Image Size in HTML Tags

Category: clever_tricks
Tags: html, perl, automation, text-manipulation

Automatically insert image width and height into HTML img tags using Perl's Image::Size module

Example: Example (Vim)

```
function! PerlImageSize()
  g/src
  normal $
  normal N
  normal 2w
  perl << EOF
  use Image::Size;
  $jo = VIM::Eval("expand('<cfile>');");
  $size = Image::Size::html_imgsize("$jo");
  VIM::Eval("setreg('a', '$size')");
EOF
  normal $
  normal "ap
endfunction
nnoremap <F4> :call PerlImageSize()<CR>
```

Example: Example (Neovim)

```
function _G.perl_image_size()
  -- Note: This requires Perl and Image::Size module
  vim.cmd('g/src')
  vim.cmd('normal! $')
  vim.cmd('normal! N')
  vim.cmd('normal! 2w')

  -- Perl functionality would need to be replicated differently in Lua
  -- Potential alternative: Use Lua image libraries or external commands
end

vim.keymap.set('n', '<F4>', _G.perl_image_size, { desc = 'Insert image size
↪  in HTML tag' })
```

Source: vim.fandom.com

11.78 Quickly Obscure Text with Rot13 Encoding

Category: clever_tricks
Tags: text-manipulation, key-mapping, security

Instantly encode text using rot13 to quickly obscure screen contents when someone approaches

Example: Example (Vim)

```
map <F3> ggg?G
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F3>', 'ggg?G', { desc = 'Obscure text with rot13
↪  encoding' })
```

Source: vim.fandom.com

11.79 Recursive Macros for Automated Text Manipulation

Category: clever_tricks
Tags: macros, text-processing, automation

Use recursive macros to perform complex text transformations across an entire file by recording a macro that calls itself, allowing for dynamic and flexible editing

Example: Example (Vim)

```
" Record a recursive macro
qqq  " Clear register q
qq   " Start recording
f2   " Find '2' character
D    " Delete rest of line
Fr   " Find last 'r'
p    " Paste deleted text
j    " Move to next line
@q   " Execute macro recursively
q    " Stop recording
```

Example: Example (Neovim)

```
-- Lua equivalent of recursive macro recording
-- Note: Macros are typically still recorded in normal mode
-- Can be executed in Lua using vim.fn.execute('@q')
```

Source: vim.fandom.com

11.80 Repeat Last Change Efficiently

Category: clever_tricks
Tags: productivity, editing, normal-mode

Use the dot (.) command to repeat the last change in normal mode, which can dramatically speed up repetitive editing tasks

Example: Example (Vim)

```
# Example use cases:
# After deleting a word with 'dw', press '.' to delete another word
# After joining lines with 'J', press '.' to join more lines
```

Example: Example (Neovim)

```
-- Dot command works the same in Neovim
-- Simply press '.' to repeat the last normal mode change
-- Can be used with a count, e.g. '5.' to repeat 5 times
```

Source: vim.fandom.com

11.81 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: Example (Vim)

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

Source: Community contributed

11.82 Reverse Block Order Using Macros

Category: clever_tricks
Tags: macros, text-manipulation, recording

Efficiently reorder blocks of text by recording a macro to move blocks to the beginning of the file

Example: Example (Vim)

```
ma
qa
jV/<\/div><CR>
:m0<CR>
'a
q
@a@@@@@@
```

Example: Example (Neovim)

```
-- Note: Translated macro approach
-- 1. Set mark at first block end
-- 2. Record macro to move subsequent blocks
-- 3. Replay macro multiple times

-- Example implementation
local function reverse_blocks()
  -- Start at first block end
  -- Record macro to move next block to top
  -- Replay macro until blocks reordered
end
```

Source: vim.fandom.com

11.83 Rot13 Encoding for Visible Screen Only

Category: clever_tricks
Tags: text-manipulation, cursor-position, key-mapping

Encode only the visible screen with rot13, preserving cursor position

Example: Example (Vim)

```
map <F3> mzHVLg?`z
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F3>', 'mzHVLg?`z', { desc = 'Rot13 encode visible
↪  screen' })
```

Source: vim.fandom.com

11.84 Rotate Email Signatures Dynamically

Category: clever_tricks
Tags: email, mapping, text-manipulation

A clever way to rotate through multiple email signatures from a text file, automatically moving the current signature to the end after use

Example: Example (Vim)

```
map <Leader>ms :e c:\sign.txt<CR>ggV/^--
↪  $<CR>k"*xG$a<C-R><C-O>*<Esc>:w<CR>:bd<CR>G$a<C-M><Esc>"*P
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>ms', function()
  -- Open signatures file
  vim.cmd.edit('~/sign.txt')

  -- Extract first signature
  -- Note: This would require more complex Lua manipulation
  -- Recommend reimplementing with more robust file handling
  vim.cmd('normal! ggV/^--  $
k"*xG$a<C-R><C-O>*')
  vim.cmd.write()
  vim.cmd.bdelete()
  vim.cmd('normal! G$a
')
end, { desc = 'Rotate Email Signature' })
```

Source: vim.fandom.com

11.85 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: Example (Vim)

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

Source: Community contributed

11.86 Scroll windows together

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

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

Example: Example (Vim)

```
" Vimscript:
" In first window:
:set scrollbind

" In second window:
:set scrollbind

" Now both windows scroll together
```

```
" To disable:
:set noscrollbind
```

Example: Example (Neovim)

```
-- Lua:
-- In first window:
vim.wo.scrollbind = true

-- In second window:
vim.wo.scrollbind = true

-- To disable:
vim.wo.scrollbind = false
```

Source: Community contributed

11.87 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: Example (Vim)

```
: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)
```

Source: Community contributed

11.88 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: Example (Vim)

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

Source: Community contributed

11.89 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: Example (Vim)

```
" 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
```

Source: Community contributed

11.90 Swap two characters

Category: Clever Tricks
Tags: character, swap, transpose

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

Example: Example (Vim)

```
xp  " swap characters
```

Source: Community contributed

11.91 Toggle File Path Slashes Quickly

Category: clever_tricks
Tags: file-paths, search-replace, mapping

Easily convert between backslashes and forward slashes in file paths, especially useful for cross-platform file handling

Example: Example (Vim)

```
function! ToggleSlash(independent) range
  let from = ''
  for lnum in range(a:firstline, a:lastline)
    let line = getline(lnum)
    let first = matchstr(line, '[/\\]')
    if !empty(first)
      if a:independent || empty(from)
        let from = first
```

```
      endif
      let opposite = (from == '/' ? '\\' : '/')
      call setline(lnum, substitute(line, from, opposite, 'g'))
    endif
  endfor
endfunction
command! -bang -range ToggleSlash <line1>,<line2>call ToggleSlash(<bang>1)
noremap <silent> <F8> :ToggleSlash<CR>
```

Example: Example (Neovim)

```
function _G.toggle_slashes(independent, first_line, last_line)
  for lnum = first_line, last_line do
    local line = vim.fn.getline(lnum)
    local first = line:match('[/\\]')
    if first then
      local from = independent and first or (from or first)
      local opposite = from == '/' and '\\' or '/'
      vim.fn.setline(lnum, line:gsub(vim.pesc(from), opposite))
    end
  end
end

vim.api.nvim_create_user_command('ToggleSlash', function(opts)
  local first_line = opts.line1
  local last_line = opts.line2
  local independent = opts.bang
  _G.toggle_slashes(independent, first_line, last_line)
end, { range = true, bang = true })

vim.keymap.set('n', '<F8>', ':ToggleSlash<CR>', { silent = true })
```

Source: vim.fandom.com

11.92 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: Example (Vim)

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

Source: Community contributed

11.93 Transfer Text Between Vim Instances

Category: clever_tricks
Tags: file-transfer, cross-session, text-exchange

Easily copy and paste text between different Vim/Neovim sessions using a temporary file

Example: Example (Vim)

```
" Copy and paste between different Vim sessions
nmap _Y :!echo "" > ~/.vi_tmp<CR><CR>:w! ~/.vi_tmp<CR>
vmap _Y :w! ~/.vi_tmp<CR>
nmap _P :r ~/.vi_tmp<CR>
```

Example: Example (Neovim)

```
-- Function to copy text to a temp file
function CopyToTempFile()
  vim.fn.writefile(vim.fn.getline(1, '$'), vim.fn.expand('~/.vi_tmp'))
end

-- Function to paste from temp file
function PasteFromTempFile()
  vim.cmd('r ~/.vi_tmp')
end

-- Set up keymappings
vim.keymap.set({'n', 'v'}, '_Y', CopyToTempFile, { desc = 'Copy to temp
↪  file' })
vim.keymap.set('n', '_P', PasteFromTempFile, { desc = 'Paste from temp file'
↪  })
```

Source: vim.fandom.com

11.94 Use Vim as a Bookmark Manager

Category: clever_tricks
Tags: url-management, browser-integration, productivity

Create a centralized URL bookmark file that can be easily opened in a web browser directly from Vim

Example: Example (Vim)

```
autocmd BufRead ~/url map <F8> :call BrowserURL()<CR>
let g:web_browser = "konqueror"
function! BrowserURL()
  if getline('.') !~ '^\s*\#'
    if g:web_browser == "konqueror"
```

```
      exe ":!dcop `dcopfind -a 'konqueror*'` konqueror-mainwindow\#1 newTab
      ↪  ".getline('.')
    elseif g:web_browser == "mozilla"
      exe ":!mozilla -remote \"openurl(".getline('.')", new-tab)\""
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufRead', {
  pattern = '~/url',
  callback = function()
    vim.keymap.set('n', '<F8>', function()
      local line = vim.fn.getline('.')
      if not line:match('^%s*#') then
        vim.fn.system('xdg-open ' .. line)
      end
    end)
  end
})
```

Source: vim.fandom.com

11.95 Vim-Like Browser Navigation Extensions

Category: clever_tricks
Tags: browser, productivity, navigation

Multiple browser extensions provide Vim-style keyboard navigation, allowing users to browse without a mouse using familiar Vim keybindings

Example: Example (Neovim)

```
-- Recommended extensions for Vim-like browser navigation:
-- Chrome/Firefox: Vimium, Tridactyl, Surfingkeys
-- Dedicated Vim browsers: Qutebrowser, Vieb, Vimb
```

Source: vim.fandom.com

11.96 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: Example (Vim)

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

Source: Community contributed

11.97 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: Example (Vim)

```
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
```

Source: Community contributed

CHAPTER 12
Clipboard

12.1 Auto Copy Mouse Selection to Clipboard

Category: clipboard
Tags: mouse, clipboard, cross-platform

Automatically copy mouse-selected text to system clipboard in Vim/Neovim

Example: Example (Vim)

```
" Method 1 (for X11 and Windows)
:noremap <LeftRelease> "+y<LeftRelease>

" Method 2 (Windows only)
:set guioptions+=a
```

Example: Example (Neovim)

```
-- Method 1: Map mouse release to copy
vim.keymap.set('n', '<LeftRelease>', '"+y<LeftRelease>', { noremap = true })

-- Method 2: Enable auto-copy option
vim.o.guioptions = vim.o.guioptions .. 'a'
```

Source: vim.fandom.com

12.2 Automatic Visual Selection Clipboard Copy

Category: clipboard
Tags: clipboard, visual-mode, gui

Automatically copy visual selections to clipboard, allowing easy pasting in other applications

Example: Example (Vim)

```
set go+=a
```

Example: Example (Neovim)

```
vim.opt.guioptions:append('a')
```

Source: vim.fandom.com

12.3 Clipboard Access in Cygwin Vim

Category: clipboard
Tags: cross-platform, windows, clipboard

Provides robust methods for copying and pasting between Vim and Windows clipboard using /dev/clipboard

Example: Example (Vim)

```
function! Putclip(type, ...) range
  let sel_save = &selection
  let &selection = "inclusive"
  let reg_save = @@
  if a:type == 'n'
    silent exe a:firstline . "," . a:lastline . "y"
  elseif a:type == 'c'
    silent exe a:1 . "," . a:2 . "y"
  else
    silent exe "normal! `<" . a:type . "`>y"
  endif

  call writefile(split(@@,"\n"), '/dev/clipboard')

  let &selection = sel_save
  let @@ = reg_save
endfunction
```

Example: Example (Neovim)

```
function _G.put_clipboard(type)
  local sel_save = vim.o.selection
  vim.o.selection = 'inclusive'

  local reg_save = vim.fn.getreg('@')
  local lines

  if type == 'n' then
    lines = vim.api.nvim_buf_get_lines(0, vim.fn.line('.')-1,
    ↪  vim.fn.line('.'), false)
  elseif type == 'v' then
    lines = vim.fn.getline(vim.fn.line("'<"), vim.fn.line("'>"))
  end

  vim.fn.writefile(lines, '/dev/clipboard')
```

```
  vim.o.selection = sel_save
  vim.fn.setreg('@', reg_save)
end

-- Mappings
vim.keymap.set('n', '<leader>y', function() _G.put_clipboard('n') end)
vim.keymap.set('v', '<leader>y', function() _G.put_clipboard('v') end)
```

Source: vim.fandom.com

12.4 Copy Entire Buffer to Clipboard

Category: clipboard
Tags: registers, copy, clipboard

Multiple methods to copy entire buffer contents to system clipboard across different modes

Example: Example (Vim)

```
" Normal mode copy
gg"+yG

" Ex mode copy
:%y+
```

Example: Example (Neovim)

```
-- Lua equivalent for copying entire buffer
vim.cmd(':%y+')
```

Source: vim.fandom.com

12.5 Copy Filename to Clipboard

Category: clipboard
Tags: file-operations, key-mapping, clipboard

Quickly copy current file's name or full path to system clipboard using custom key mappings

Example: Example (Vim)

```
" Copy filename mappings
" Short filename
nmap ,cs :let @*=expand("%")<CR>
" Full path
nmap ,cl :let @*=expand("%:p")<CR>
```

Example: Example (Neovim)

```
-- Copy filename to clipboard
vim.keymap.set('n', ',cs', function()
  vim.fn.setreg('+', vim.fn.expand('%'))
end, { desc = 'Copy short filename' })

vim.keymap.set('n', ',cl', function()
  vim.fn.setreg('+', vim.fn.expand('%:p'))
end, { desc = 'Copy full file path' })
```

Source: vim.fandom.com

12.6 Copy to Windows Clipboard in Cygwin Vim

Category: clipboard
Tags: cross-platform, system-clipboard, utilities

Function to copy text from Vim to Windows clipboard using /dev/clipboard device, supporting UTF-8 characters

Example: Example (Vim)

```
function! Putclip(type, ...) range
  let sel_save = &selection
  let &selection = "inclusive"
  let reg_save = @@
  if a:type == 'n'
    silent exe a:firstline . "," . a:lastline . "y"
  elseif a:type == 'c'
    silent exe a:1 . "," . a:2 . "y"
  else
    silent exe "normal! `<" . a:type . "`>y"
  endif

  call writefile(split(@@,"\n"), '/dev/clipboard')

  let &selection = sel_save
  let @@ = reg_save
endfunction

" Mappings
vnoremap <silent> <leader>y :call Putclip(visualmode(), 1)<CR>
nnoremap <silent> <leader>y :call Putclip('n', 1)<CR>
```

Example: Example (Neovim)

```
function Putclip(type, ...)
  local sel_save = vim.o.selection
  vim.o.selection = 'inclusive'

  local reg_save = vim.fn.getreg('@')
```

```
    if type == 'n' then
      vim.cmd(string.format('%d,%dy', vim.fn.line("'"), vim.fn.line("'")))
    elseif type == 'c' then
      vim.cmd(string.format('%d,%dy', ...[1], ...[2]))
    else
      vim.cmd('normal! `<' .. type .. '`>y')
    end

    local clipboard_content = vim.fn.getreg('@')
    vim.fn.writefile(vim.split(clipboard_content, "\n"), '/dev/clipboard')

    vim.o.selection = sel_save
    vim.fn.setreg('@', reg_save)
  end

  -- Mappings
  vim.keymap.set('v', '<leader>y', function() Putclip(vim.fn.visualmode(), 1)
  ↪  end)
  vim.keymap.set('n', '<leader>y', function() Putclip('n', 1) end)
```

Source: vim.fandom.com

12.7 Cross-Platform Filename Copying

Category: clipboard
Tags: cross-platform, file-operations, clipboard

Platform-specific clipboard handling for Windows and Unix-like systems, including conversion of path separators

Example: Example (Vim)

```
if has('win32')
  nmap ,cs :let @*=substitute(expand("%"), "/", "\\", "g")<CR>
  nmap ,cl :let @*=substitute(expand("%:p"), "/", "\\", "g")<CR>
else
  nmap ,cs :let @*=expand("%")<CR>
  nmap ,cl :let @*=expand("%:p")<CR>
endif
```

Example: Example (Neovim)

```
if vim.fn.has('win32') == 1 then
  vim.keymap.set('n', ',cs', function()
    local filename = vim.fn.expand('%'):gsub('/', '\\')
    vim.fn.setreg('*', filename)
  end)
  vim.keymap.set('n', ',cl', function()
    local filepath = vim.fn.expand('%:p'):gsub('/', '\\')
    vim.fn.setreg('*', filepath)
```

```
  end)
else
  vim.keymap.set('n', ',cs', function()
    vim.fn.setreg('*', vim.fn.expand('%'))
  end)
  vim.keymap.set('n', ',cl', function()
    vim.fn.setreg('*', vim.fn.expand('%:p'))
  end)
end
```

Source: vim.fandom.com

12.8 Cross-Platform System Clipboard Integration

Category: clipboard
Tags: clipboard, cross-platform, system-copy

Provides universal system clipboard copy/paste methods for Mac, Linux, and Ubuntu using external utilities

Example: Example (Vim)

```
" Mac OS X Clipboard
vmap <C-c> y:call system("pbcopy", getreg(""))<CR>
nmap <C-v> :call setreg("",system("pbpaste"))<CR>p

" Ubuntu Clipboard
vmap <C-c> y:call system("xclip -i -selection clipboard", getreg(""))<CR>
```

Example: Example (Neovim)

```
-- Mac OS X Clipboard
vim.keymap.set({'v'}, '<C-c>', function()
  vim.fn.system('pbcopy', vim.fn.getreg('"'))
end, { noremap = true, silent = true })

-- Ubuntu Clipboard
vim.keymap.set({'v'}, '<C-c>', function()
  vim.fn.system('xclip -i -selection clipboard', vim.fn.getreg('"'))
end, { noremap = true, silent = true })
```

Source: vim.fandom.com

12.9 Cross-Session Clipboard Sharing

Category: clipboard
Tags: registers, clipboard, cross-session

Copy and paste between multiple Vim/Neovim sessions using a temporary file or regis-

ter

Example: Example (Vim)

```
let g:session_yank_file="~/.vim_yank"
map <silent> <Leader>y :call Session_yank()<CR>
vmap <silent> <Leader>y y:call Session_yank()<CR>

function Session_yank()
  new
  call setline(1,getregtype())
  put
  silent exec 'wq! ' . g:session_yank_file
  exec 'bdelete ' . g:session_yank_file
endfunction

function Session_paste(command)
  silent exec 'sview ' . g:session_yank_file
  let l:opt=getline(1)
  silent 2,$yank
  call setreg('"', @", l:opt)
  exec 'bdelete ' . g:session_yank_file
  exec 'normal ' . a:command
endfunction
```

Example: Example (Neovim)

```
local session_yank_file = vim.fn.expand('~/.vim_yank')

local function session_yank()
  vim.cmd('new')
  vim.fn.setline(1, vim.fn.getregtype())
  vim.cmd('put')
  vim.fn.writefile(vim.fn.getline(1, '$'), session_yank_file)
  vim.cmd('bdelete ' .. session_yank_file)
end

local function session_paste(command)
  vim.cmd('sview ' .. session_yank_file)
  local opt = vim.fn.getline(1)
  vim.cmd('2,$yank')
  vim.fn.setreg('"', vim.fn.getreg('"'), opt)
  vim.cmd('bdelete ' .. session_yank_file)
  vim.cmd('normal ' .. command)
end

-- Add keymaps to use these functions
vim.keymap.set('n', '<Leader>y', session_yank, { desc = 'Yank to session
↪  file' })
vim.keymap.set('n', '<Leader>p', function() session_paste('p') end, { desc =
↪  'Paste from session file' })
```

Source: vim.fandom.com

12.10 Disable Auto Copy Clipboard Function

Category: clipboard
Tags: configuration, clipboard, settings

Remove automatic clipboard copying to prevent overwriting clipboard buffer

Example: Example (Vim)

```
" Disable auto copy function
set guioptions-=a
set guioptions-=A
set guioptions-=aA
```

Example: Example (Neovim)

```
-- Disable auto copy in Neovim
vim.o.guioptions = vim.o.guioptions:gsub('[aA]', '')
```

Source: vim.fandom.com

12.11 Easy System Clipboard Integration

Category: clipboard
Tags: clipboard, registers, cross-platform

Configure Vim/Neovim to use system clipboard by default, making copy/paste across applications seamless

Example: Example (Vim)

```
" Use system clipboard as default register
set clipboard=unnamed
" For X11 systems, use unnamedplus
set clipboard=unnamedplus
```

Example: Example (Neovim)

```
-- Use system clipboard as default register
vim.opt.clipboard = 'unnamed'
-- For X11 systems
vim.opt.clipboard = 'unnamedplus'
```

Source: vim.fandom.com

12.12 Easy Windows Clipboard Interaction

Category: clipboard
Tags: clipboard, register, cross-platform

Simplify clipboard operations between Vim and Windows applications by automatically syncing the unnamed register with the system clipboard

Example: Example (Vim)

```
set clipboard=unnamed
```

Example: Example (Neovim)

```
vim.opt.clipboard = 'unnamed'
```

Source: vim.fandom.com

12.13 Mac OS X Clipboard Integration

Category: clipboard
Tags: macos, clipboard, system-integration

Enable seamless clipboard sharing between Vim and macOS system clipboard

Example: Example (Vim)

```
set clipboard=unnamed
```

Example: Example (Neovim)

```
vim.opt.clipboard = 'unnamed'
```

Source: vim.fandom.com

12.14 Mac OS clipboard sharing

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

Integrate Vim with macOS clipboard using pbcopy and pbpaste utilities.

Example: Example (Vim)

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

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

Example: Example (Neovim)

```
-- Lua - macOS clipboard integration
vim.keymap.set('v', '<C-c>', ':w !pbcopy<CR><CR>', { desc = 'Copy to macOS
↪  clipboard' })
vim.keymap.set('n', '<C-v>', ':r !pbpaste<CR>', { desc = 'Paste from macOS
↪  clipboard' })

-- Use system clipboard by default
vim.opt.clipboard = 'unnamed'
```

Source: Community contributed

12.15 Middle-Click Paste Handling

Category: clipboard
Tags: paste, clipboard, mouse

Improve middle-click paste behavior across different Vim modes, with special handling for insert and visual modes

Example: Example (Vim)

```
"Middle-click paste handling
nnoremap <MiddleMouse> i<MiddleMouse>
vnoremap <MiddleMouse> s<MiddleMouse>
```

Example: Example (Neovim)

```
-- Improved middle-click paste handling
vim.keymap.set('n', '<MiddleMouse>', 'i<MiddleMouse>', { noremap = true })
vim.keymap.set('v', '<MiddleMouse>', 's<MiddleMouse>', { noremap = true })
```

Source: vim.fandom.com

12.16 Paste Clipboard Contents in New Vim Window

Category: clipboard
Tags: clipboard, paste, register

Quickly open a new Vim buffer with clipboard contents using built-in register

Example: Example (Vim)

```
:new
"*p
```

Example: Example (Neovim)

```
-- Open new buffer and paste from clipboard register
vim.cmd('new')
vim.cmd('normal "*p')
```

Source: vim.fandom.com

12.17 Paste from Windows Clipboard in Cygwin Vim

Category: clipboard
Tags: cross-platform, system-clipboard, utilities

Function to paste text from Windows clipboard into Vim using /dev/clipboard device

Example: Example (Vim)

```
function! Getclip()
  let reg_save = @@
  let @@ = join(readfile('/dev/clipboard'), "\n")
  setlocal paste
  exe 'normal p'
  setlocal nopaste
  let @@ = reg_save
endfunction

nnoremap <silent> <leader>p :call Getclip()<CR>
```

Example: Example (Neovim)

```
function Getclip()
  local reg_save = vim.fn.getreg('@')
  local clipboard_content = table.concat(vim.fn.readfile('/dev/clipboard'),
  ↪  '\n')

  vim.o.paste = true
  vim.fn.setreg('@', clipboard_content)
  vim.cmd('normal! p')
  vim.o.paste = false

  vim.fn.setreg('@', reg_save)
end
```

```
vim.keymap.set('n', '<leader>p', Getclip)
```

Source: vim.fandom.com

12.18 Paste from Windows Clipboard in Vim

Category: clipboard
Tags: clipboard, cross-platform, paste

Easy method to paste from Windows clipboard in Cygwin Vim, supporting UTF-8

Example: Example (Vim)

```
function! Getclip()
  let reg_save = @@
  let @@ = join(readfile('/dev/clipboard'), "\n")
  setlocal paste
  exe 'normal p'
  setlocal nopaste
  let @@ = reg_save
endfunction
```

Example: Example (Neovim)

```
function _G.get_clipboard()
  local reg_save = vim.fn.getreg('@')
  local clipboard_content = table.concat(vim.fn.readfile('/dev/clipboard'),
  ↪  '\n')

  vim.o.paste = true
  vim.api.nvim_put({clipboard_content}, '', true, true)
  vim.o.paste = false

  vim.fn.setreg('@', reg_save)
end

-- Mapping
vim.keymap.set('n', '<leader>p', _G.get_clipboard)
```

Source: vim.fandom.com

12.19 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: Example (Vim)

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

Example: Example (Neovim)

```
-- 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 })
```

Source: Community contributed

12.20 Quick Clipboard Copy/Paste Mappings

Category: clipboard
Tags: mapping, clipboard, productivity

Create custom keymaps for easy copying and pasting using pbcopy/pbpaste

Example: Example (Vim)

```
nmap <F1> :set paste<CR>:r !pbpaste<CR>:set nopaste<CR>
imap <F1> <Esc>:set paste<CR>:r !pbpaste<CR>:set nopaste<CR>
nmap <F2> :.w !pbcopy<CR><CR>
vmap <F2> :w !pbcopy<CR><CR>
```

Example: Example (Neovim)

```
vim.keymap.set({'n', 'i'}, '<F1>', function()
  vim.o.paste = true
  vim.cmd('r !pbpaste')
  vim.o.paste = false
end)

vim.keymap.set('n', '<F2>', ':.w !pbcopy<CR>', { silent = true })
vim.keymap.set('v', '<F2>', ':w !pbcopy<CR>', { silent = true })
```

Source: vim.fandom.com

12.21 Quick Cross-Window Copy and Paste

Category: clipboard
Tags: key-mapping, clipboard, visual-mode

One-key solution for copying and pasting between windows using clipboard register

Example: Example (Vim)

```
" Copy current word or selection to clipboard
nnoremap <F4> "+yiw
vnoremap <F4> "+y

" Replace with clipboard contents
nnoremap <F5> viw"+p
vnoremap <F5> "+p
```

Example: Example (Neovim)

```
-- Copy current word or selection to clipboard
vim.keymap.set('n', '<F4>', '"+yiw', { desc = 'Copy word to clipboard' })
vim.keymap.set('v', '<F4>', '"+y', { desc = 'Copy selection to clipboard' })

-- Replace with clipboard contents
vim.keymap.set('n', '<F5>', 'viw"+p', { desc = 'Replace word with clipboard'
↪  })
vim.keymap.set('v', '<F5>', '"+p', { desc = 'Replace selection with
↪  clipboard' })
```

Source: vim.fandom.com

12.22 Quick External Paste Buffer Editing

Category: clipboard
Tags: paste-buffer, external-editing, clipboard

Quickly open a Vim buffer containing the current clipboard contents for editing, automatically updating the clipboard on save

Example: Example (Vim)

```
function vxp()
  # Edit paste contents
  gvim -c 'normal ggdG"*p' c:/aaa/xp
endfunction

# In vimrc
autocmd bufWritePost c:/aaa/xp normal ggVG"*y
```

Example: Example (Neovim)

```
function _G.edit_paste_buffer()
  -- Open a new buffer with clipboard contents
  vim.cmd('gvim -c "normal! ggdG\"*p" c:/aaa/xp')
end
```

```
-- Automatically copy buffer contents back to clipboard on save
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = 'c:/aaa/xp',
  callback = function()
    vim.cmd('normal! ggVG"*y')
  end
})
```

Source: vim.fandom.com

12.23 Quick System Clipboard Copy and Paste

Category: clipboard
Tags: system-clipboard, key-mapping, cross-platform

Convenient mappings for copying, cutting, and pasting using system clipboard across different modes

Example: Example (Vim)

```
" Copy to clipboard register
vmap <C-c> "*y

" Select all text
nmap <C-a> ggVG

" Normal copy/paste mappings
vmap <C-c> y<Esc>i
vmap <C-x> d<Esc>i
imap <C-v> <Esc>pi
```

Example: Example (Neovim)

```
-- Copy to system clipboard in visual mode
vim.keymap.set('v', '<C-c>', '"*y', { desc = 'Copy to system clipboard' })

-- Select entire buffer
vim.keymap.set('n', '<C-a>', 'ggVG', { desc = 'Select all text' })

-- Paste from system clipboard in insert mode
vim.keymap.set('i', '<C-v>', '<Esc>pi', { desc = 'Paste from system
↪  clipboard' })
```

Source: vim.fandom.com

12.24 Set system clipboard from Lua

Category: Clipboard
Tags: clipboard, lua, register

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

Example: Example (Vim)

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

Source: Community contributed

12.25 Swap Between Vim and System Clipboard

Category: clipboard
Tags: register-management, clipboard, cross-system

Easily transfer text between Vim's unnamed register and system clipboard (x11 clipboard)

Example: Example (Vim)

```
" Transfer between Vim register and system clipboard
nnoremap <Leader>s :let @a=@" | let @"=@+ | let @+=@a<CR>
```

Example: Example (Neovim)

```
-- Transfer between Vim register and system clipboard
vim.keymap.set('n', '<Leader>s', function()
  local unnamed = vim.fn.getreg('"')
  vim.fn.setreg('"', vim.fn.getreg('+'))
  vim.fn.setreg('+', unnamed)
end, { desc = 'Swap clipboard and unnamed register' })
```

Source: vim.fandom.com

12.26 System Clipboard Integration

Category: clipboard
Tags: system-clipboard, cross-platform, copy-paste

Access system clipboard using + and * registers in Vim/Neovim

Example: Example (Vim)

```
"+y  " copy to system clipboard
"+p  " paste from system clipboard
```

Example: Example (Neovim)

```
-- Ensure clipboard support is compiled in
vim.opt.clipboard = 'unnamedplus'  -- Use system clipboard

-- Copy and paste using system clipboard
vim.keymap.set({'n', 'v'}, '<leader>y', '"+y', { desc = 'Copy to system
↪  clipboard' })
vim.keymap.set({'n', 'v'}, '<leader>p', '"+p', { desc = 'Paste from system
↪  clipboard' })
```

Source: vim.fandom.com

12.27 System clipboard: handling yank and delete motions differently

Category: Clipboard
Tags: clipboard, copy, paste

Integrating system clipboard with Neovim is easy: All yanked and deleted content will go to the default register and also to system clipboard. That's the most common setup. But, suppose that you want yank and delete motions to behave slightly differently with respect to system clipboard. For example, you want all yanked text to go to the unnamed register and to system clipboard. But in case of delete motions, you don't want to affect system clipboard, only the default registry (quite popular setup). The setup is fairly easy. Just add the following lines to your `init.lua` configuration file: Yanked text goes to system register AND clipboard. Deleted text goes to system register only and leaves system clipboard attached.

Example: Example (Neovim)

```
vim.o.clipboard="unnamedplus"

-- Don't mention system clipboard,
-- it won't be affected wuthout the `TextYankPost` callback
vim.o.clipboard = ""

-- Copy yanked text to system clipboard.
-- This callback is involed after each yank/delete operation
vim.api.nvim_create_autocmd("TextYankPost", {
  callback = function()
    -- Only trigger on yanks
    if vim.v.event.operator == "y" and vim.v.event.regname == "" then
      -- "+" stands for system clipboard
      vim.fn.setreg("+", vim.fn.getreg('""'))
```

```
      vim.fn.setreg("+", vim.fn.getreg('"', 1)) -- also yank visual
      ↪  selection
    end
  end,
})
```

Source: Community contributed

12.28 Transfer Text Between Vim Sessions Using Clipboard

Category: clipboard
Tags: registers, cross-session, clipboard

Easily copy text between different Vim/Neovim sessions using the '+' register, which interfaces with the system clipboard

Example: Example (Vim)

```
" Yank two lines to clipboard register
"+2yy

" Paste from clipboard register in normal mode
"+p

" Paste from clipboard register in insert mode
<Ctrl-R>+
```

Example: Example (Neovim)

```
-- Yank two lines to clipboard register
vim.cmd('normal! "+2yy')

-- Paste from clipboard register in normal mode
vim.cmd('normal! "+p')

-- In insert mode, use vim.api to insert clipboard contents
vim.api.nvim_put(vim.fn.getreg('+'), 'c', true, true)
```

Source: vim.fandom.com

12.29 Use System Clipboard in Vim

Category: clipboard
Tags: clipboard, register, cross-platform

Use '+' and '*' registers to interact with system clipboard across different platforms

Example: Example (Vim)

```
" Copy entire buffer to system clipboard
gg"+yG

" Paste from system clipboard
"+p

" Set clipboard as default register
set clipboard=unnamed
" For X Window systems
set clipboard=unnamedplus
```

Example: Example (Neovim)

```
-- Copy entire buffer to system clipboard
vim.api.nvim_command('normal! gg"+yG')

-- Paste from system clipboard
vim.api.nvim_put(vim.fn.getreg('+'), 'c', true, true)

-- Set clipboard as default register
vim.opt.clipboard = 'unnamed' -- For Windows
vim.opt.clipboard = 'unnamedplus' -- For X Window systems
```

Source: vim.fandom.com

12.30 Use xclip for Efficient Linux Clipboard Operations

Category: clipboard
Tags: linux, clipboard, external-tool

Easily copy and paste between Vim and system clipboard using xclip, which is particularly useful for handling large amounts of text

Example: Example (Vim)

```
" Mapping to copy entire buffer to clipboard
:map <F7> :w !xclip<CR><CR>
" Mapping to copy visual selection to clipboard
:vmap <F7> "*y
" Mapping to paste from clipboard
:map <S-F7> :r!xclip -o<CR>
```

Example: Example (Neovim)

```
-- Copy entire buffer to clipboard
vim.keymap.set('n', '<F7>', ':w !xclip<CR>', { desc = 'Copy buffer to
↪  clipboard' })
-- Copy visual selection to clipboard
```

```
vim.keymap.set('v', '<F7>', '"*y', { desc = 'Copy selection to clipboard' })
-- Paste from clipboard
vim.keymap.set('n', '<S-F7>', ':r!xclip -o<CR>', { desc = 'Paste from
↪  clipboard' })
```

Source: vim.fandom.com

12.31 Using System Clipboard Registers

Category: clipboard
Tags: registers, clipboard, system-clipboard

Use system clipboard registers to copy and paste between Vim instances

Example: Example (Vim)

```
" Copy current line to clipboard
"*yy

" Paste from clipboard
"*p
```

Example: Example (Neovim)

```
-- Copy current line to clipboard
vim.fn.setreg('*', vim.fn.getline('.'))

-- Paste from clipboard
vim.cmd('normal "*p')
```

Source: vim.fandom.com

CHAPTER 13

Command line

13.1 Advanced Command Line Recall

Category: command_line
Tags: command-history, navigation

Use q: to open command-line window for easier command history navigation and editing without leaving home row

Example: Example (Vim)

```
" Open command-line window
q:
```

Example: Example (Neovim)

```
-- Open command-line window
vim.cmd('q:')
```

Source: vim.fandom.com

13.2 Advanced Vim/GVim Command Line Options

Category: command_line
Tags: cli-options, file-editing, startup

Powerful command-line options for opening files with specific behaviors

Example: Example (Vim)

```
# Command line option examples
gvim -u local_vimrc        # Use specific vimrc
gvim --noplugin           # Disable plugins
gvim -R important.txt     # Open read-only
gvim +10 file.txt         # Open and jump to line 10
gvim -c "/search" file.c  # Open and jump to search match
```

Example: Example (Neovim)

```
-- While these are CLI options, in Neovim you can replicate some behaviors
-- Open read-only
vim.g.started_with_read_only = true

-- Jump to specific line on open
vim.api.nvim_win_set_cursor(0, {10, 0})
```

Source: vim.fandom.com

13.3 Asynchronously Run External Commands in Windows

Category: command_line
Tags: windows, external-commands, async-execution

Run external console programs without blocking Vim, with optional pause to view output

Example: Example (Vim)

```
" Run compiled program corresponding to current source file
nnoremap <silent> <F5> :!start cmd /c "%:p:r:s,$,.exe," & pause<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for running external programs
vim.keymap.set('n', '<F5>', function()
  vim.cmd('!start cmd /c "' .. vim.fn.expand('%:p:r') .. '.exe" & pause')
end, { silent = true })
```

Source: vim.fandom.com

13.4 Base64 Decoding in Vim Using System Commands

Category: command_line
Tags: encoding, text-processing, system-commands

Decode base64 encoded strings directly within Vim using Ruby or Perl embedded interpreters

Example: Example (Vim)

```
" Ruby base64 decoding
:ruby require "base64"
:.rubydo $_=Base64.decode64 $_

" Perl base64 decoding
:perl require MIME::Base64
```

```
:.perldo $_=MIME::Base64::decode($_)
```

Example: Example (Neovim)

```
-- Lua base64 decoding (requires 'base64' luarocks package)
local base64 = require('base64')
vim.api.nvim_command(string.format(':lua vim.api.nvim_set_current_line(base
↪ 64.decode(vim.api.nvim_get_current_line()))'))
```

Source: vim.fandom.com

13.5 Capture Ex Command Output in New Tab

Category: command_line
Tags: ex-commands, output-capture, productivity

Easily capture and view output of Vim commands in a new tab for clean, temporary viewing

Example: Example (Vim)

```
function! TabMessage(cmd)
  redir => message
  silent execute a:cmd
  redir END
  if empty(message)
    echoerr "no output"
  else
    tabnew
    setlocal buftype=nofile bufhidden=wipe noswapfile nobuflisted nomodified
    silent put=message
  endif
endfunction
command! -nargs=+ -complete=command TabMessage call TabMessage(<q-args>)
```

Example: Example (Neovim)

```
function _G.tab_message(cmd)
  local output = vim.api.nvim_exec(cmd, true)
  if output == "" then
    vim.notify("No output", vim.log.levels.WARN)
    return
  end
  vim.cmd('tabnew')
  vim.api.nvim_buf_set_option(0, 'buftype', 'nofile')
  vim.api.nvim_buf_set_option(0, 'bufhidden', 'wipe')
  vim.api.nvim_buf_set_option(0, 'swapfile', false)
  vim.api.nvim_buf_set_lines(0, 0, -1, false, vim.split(output, "\n"))
end
```

```
vim.api.nvim_create_user_command('TabMessage', function(opts)
  _G.tab_message(opts.args)
end, { nargs = '+', complete = 'command' })
```

Source: vim.fandom.com

13.6 Capture External Command Output Flexibly

Category: command_line
Tags: shell-integration, system-command, text-manipulation

Provides multiple methods to capture and insert output from shell commands into Vim buffers, with flexibility for different use cases

Example: Example (Vim)

```
" Append shell command output to current line
function! GetDate(format)
  let format = empty(a:format) ? '+%A %Y-%m-%d %H:%M UTC' : a:format
  let cmd = '/bin/date -u ' . shellescape(format)
  let result = substitute(system(cmd), '[\]
|[[:cntrl:]]', '', 'g')
  call setline(line('.'), getline('.') . ' ' . result)
endfunction

" Custom command to open new buffer with command output
:command! -nargs=* -complete=shellcmd R new | setlocal buftype=nofile
↪  bufhidden=hide noswapfile | r !<args>
```

Example: Example (Neovim)

```
-- Lua equivalent for appending command output
function _G.get_date(format)
  format = format == '' and '+%A %Y-%m-%d %H:%M UTC' or format
  local cmd = string.format('/bin/date -u %s', vim.fn.shellescape(format))
  local result = vim.fn.system(cmd):gsub('[\n\r\t]', '')
  local current_line = vim.api.nvim_get_current_line()
  vim.api.nvim_set_current_line(current_line .. ' ' .. result)
end

-- Create custom command for shell output
vim.api.nvim_create_user_command('R', function(opts)
  vim.cmd('new')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'hide'
  vim.bo.swapfile = false
  vim.cmd('read !' .. opts.args)
end, { nargs = '*', complete = 'shellcmd' })
```

Source: vim.fandom.com

13.7 Chain Multiple Vim Commands with Pipe

Category: command_line
Tags: command-chaining, productivity, text-manipulation

Execute multiple Vim commands sequentially, with each command running only if the previous command succeeds

Example: Example (Vim)

```
%s/htm/html/c | %s/JPEG/jpg/c | %s/GIF/gif/c
```

Example: Example (Neovim)

```
-- For Lua, use vim.cmd to execute multiple commands
vim.cmd('s/htm/html/c | s/JPEG/jpg/c | s/GIF/gif/c')
```

Source: vim.fandom.com

13.8 Check Remote Vim Session Mode and Context

Category: command_line
Tags: remote-debugging, vim-modes, session-info

Remotely query the current Vim session's mode and working directory using client-server expressions

Example: Example (Vim)

```
vim --servername GVIM --remote-expr 'mode()'
```

Example: Example (Neovim)

```
-- Lua approach to query Vim session state
-- vim.fn.remote_expr('GVIM', 'mode()')
-- vim.fn.remote_expr('GVIM', 'getcwd()')
```

Source: vim.fandom.com

13.9 Combine Multiple Code Checking Tools in Quickfix

Category: command_line
Tags: quickfix, code-checking, error-handling

Run multiple code checking tools and combine their results in a single quickfix list, useful for comprehensive code analysis

Example: Example (Vim)

```
function! DoMake(...)
  update
  let savemp = &makeprg
  let qflist = []
  for prg in a:000
    let &makeprg = prg . ' %'
    silent make!
    let qflist += getqflist()
  endfor
  if empty(qflist)
    cclose
  else
    call setqflist(qflist)
    copen
    cfirst
  endif
  let &makeprg = savemp
endfunction

command! Pycheck call DoMake('pyflakes', 'pep8')
```

Example: Example (Neovim)

```
function _G.do_make(...)
  vim.cmd('update')
  local save_makeprg = vim.o.makeprg
  local qflist = {}

  for _, prg in ipairs({...}) do
    vim.o.makeprg = prg .. ' %'
    vim.cmd('silent make!')
    vim.list_extend(qflist, vim.fn.getqflist())
  end

  if #qflist == 0 then
    vim.cmd('cclose')
  else
    vim.fn.setqflist(qflist)
    vim.cmd('copen')
    vim.cmd('cfirst')
  end

  vim.o.makeprg = save_makeprg
end

vim.api.nvim_create_user_command('Pycheck', function()
  _G.do_make('pyflakes', 'pep8')
end, {})
```

Source: vim.fandom.com

13.10 Command Line Completion List Extraction

Category: command_line
Tags: command-completion, vim-function, utility

Extracts a list of possible command line completions using Vim's built-in completion mechanism

Example: Example (Vim)

```
function! GetCmdCompletion(cmd_prefix)
  let [cwh,ls,v] = [&cwh, &ls, @v]
  set cwh=1 ls=0
  execute 'nn <buffer> z&u :' . a:cmd_prefix . '<c-a><c-f>"vyyo<cr>'
  normal z&u
  let res = split(@v)
  let [&cwh,&ls,@v] = [cwh,ls,v]
  return res
endfunction
```

Example: Example (Neovim)

```
function _G.get_cmd_completion(cmd_prefix)
  local cwh = vim.o.cmdheight
  local ls = vim.o.laststatus
  local original_reg = vim.fn.getreg('v')

  vim.o.cmdheight = 1
  vim.o.laststatus = 0

  vim.cmd('nn <buffer> z&u :' .. cmd_prefix .. '<c-a><c-f>"vyyo<cr>')
  vim.cmd('normal z&u')

  local res = vim.fn.split(vim.fn.getreg('v'))

  vim.o.cmdheight = cwh
  vim.o.laststatus = ls
  vim.fn.setreg('v', original_reg)

  return res
end
```

Source: vim.fandom.com

13.11 Command Window Ctrl-D Completion

Category: command_line
Tags: completion, command-window, auto-completion

Add advanced completion functionality to command window using Ctrl-D

Example: Example (Vim)

```
function! s:ECWCtrlD()
  if (match(@", '^ *[a-z]\?map\s\s*\(\S\S*\)\?\s*$') >=0 )
    let s:foo = @"
    let save_more=&more
    set nomore
    execute ':redir @" |'.s:foo.'|redir END'
    let &more = save_more
    put=@"
    return
  endif
  let s:foo = substitute(@", '\(.*\)\(\S\S*\s*\)$', '\2', '')
  let s:foo = substitute(s:foo, '\s*$', '*', '')
  let @"=glob(s:foo)
  if(@" == "") | let @"='no match' | endif
  put=@"
endfunction

command -nargs=0 ECWCtrlD call s:ECWCtrlD()
```

Example: Example (Neovim)

```
local function ecw_ctrl_d()
  local line = vim.fn.getline('.')
  local cursor_pos = vim.fn.col('.')
  local text_before_cursor = line:sub(1, cursor_pos)

  local completion = vim.fn.glob(text_before_cursor .. '*')

  if completion == '' then
    completion = 'no match'
  end

  vim.fn.append('.', completion)
end

vim.api.nvim_create_user_command('ECWCtrlD', ecw_ctrl_d, {})
```

Source: vim.fandom.com

13.12 Command Window Enhanced Completion

Category: command_line
Tags: command-line, completion, search

Add advanced navigation and completion features to the command window, similar to command-line behavior

Example: Example (Vim)

```
" Arrow key navigation in command window
augroup ECW_au
  au CmdwinEnter : imap <UP> <C-O>y0<C-O>:let@/='^'.@0<CR><C-O>?<Esc><Esc>
  au CmdwinLeave : iunmap <UP>
  au CmdwinEnter : imap <DOWN> <C-O>y0<C-O>:let@/='^'.@0<CR><C-O>/<Esc><Esc>
  au CmdwinLeave : iunmap <DOWN>
  au CmdwinLeave : :let @/=""
augroup END
```

Example: Example (Neovim)

```
-- Enhanced command window navigation
local augroup = vim.api.nvim_create_augroup('ECW_au', { clear = true })

vim.api.nvim_create_autocmd('CmdwinEnter', {
  group = augroup,
  pattern = ':',
  callback = function()
    vim.keymap.set('i', '<Up>', function()
      vim.cmd('normal! y0')
      vim.fn.setreg('/', '^' .. vim.fn.getreg('"'))
      vim.cmd('normal! ?')
    end, { buffer = true })

    vim.keymap.set('i', '<Down>', function()
      vim.cmd('normal! y0')
      vim.fn.setreg('/', '^' .. vim.fn.getreg('"'))
      vim.cmd('normal! /')
    end, { buffer = true })
  end
})

vim.api.nvim_create_autocmd('CmdwinLeave', {
  group = augroup,
  callback = function()
    vim.fn.setreg('/', '')
  end
})
```

Source: vim.fandom.com

13.13 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: Example (Vim)

```
: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
```

Source: Community contributed

13.14 Command-Line Help Search Functions

Category: command_line
Tags: help, search, documentation

Bash functions to quickly search and open Vim help or helpgrep results

Example: Example (Vim)

```
vimhelp() {
  view -c "help $1" -c on -c "au! VimEnter *"
}

vimhelpgrep() {
  vim -c "helpgrep $1" -c on -c copen -c "au! VimEnter *"
}
```

Example: Example (Neovim)

```
-- Lua functions for Neovim help search
local function vim_help(topic)
  vim.cmd(string.format('help %s', topic))
  vim.cmd('only')
end

local function vim_help_grep(topic)
  vim.cmd(string.format('helpgrep %s', topic))
  vim.cmd('copen')
end
```

Source: vim.fandom.com

13.15 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: Example (Vim)

```
" 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
```

Source: Community contributed

13.16 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: Example (Vim)

```
" 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)
```

Source: Community contributed

13.17 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: Example (Vim)

```
" 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
```

Source: Community contributed

13.18 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: Example (Vim)

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

Source: Community contributed

13.19 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: Example (Vim)

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

Source: Community contributed

13.20 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: Example (Vim)

```
" 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)
```

Source: Community contributed

13.21 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: Example (Vim)

```
" 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
```

Source: Community contributed

13.22 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: Example (Vim)

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

Source: Community contributed

13.23 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: Example (Vim)

```
" 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
```

Source: Community contributed

13.24 Compile Current Buffer with Flexible Make Command

Category: command_line
Tags: compilation, build, make, quickfix

Easily compile the current file and populate quickfix list with compiler errors

Example: Example (Vim)

```
" Set make program to compile current file
set makeprg=gcc\ -o\ %<\ %

" Compile current file
:make %:r
```

Example: Example (Neovim)

```
-- Set make program for compiling current file
vim.o.makeprg = 'gcc -o %< %'

-- Compile current file and open quickfix
vim.cmd('make %:r')
```

Source: vim.fandom.com

13.25 Compile Current File with Quickfix Support

Category: command_line
Tags: compilation, quickfix, make

Set up a quick way to compile the current file and open compilation errors in the quickfix window

Example: Example (Vim)

```
" Configure make program for gcc compilation
set makeprg=gcc\ -o\ %<\ %

" Mapping to compile current file
nnoremap <F7> :update<CR>:make<CR>
```

Example: Example (Neovim)

```
-- Configure make program for gcc compilation
vim.o.makeprg = 'gcc -o %< %'
```

```
-- Mapping to compile current file
vim.keymap.set('n', '<F7>', function()
  vim.cmd.write()
  vim.cmd.make()
end, { desc = 'Compile current file' })
```

Source: vim.fandom.com

13.26 Complex Vim Command-Line Chaining

Category: command_line
Tags: command-line, vim-tricks, advanced-usage

Demonstrates how to chain multiple Vim commands using -c flag, useful for quick one-liner operations or automated tasks

Example: Example (Vim)

```
vim -c "cmd1|cmd2|cmd3"
# Example: Print paste contents to default printer
gvim -c 's/^/\=@*/|hardcopy!|q!'
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.cmd
-- Note: Direct command-line usage remains similar
vim.cmd('s/^/\=@*')
vim.cmd('hardcopy!')
vim.cmd('q!')
```

Source: vim.fandom.com

13.27 Configure Quickfix for Visual Studio Build Errors

Category: command_line
Tags: build-integration, error-handling, compiler

Set up a custom make program and mapping to parse Visual Studio compiler errors in Vim, enabling quick navigation through compilation errors

Example: Example (Vim)

```
set makeprg=/bin/dovcmake
map <F7> :make <C-R>%<CR>
```

Example: Example (Neovim)

```
vim.o.makeprg = '/bin/dovcmake'
vim.keymap.set('n', '<F7>', function()
  vim.cmd('make ' .. vim.fn.expand('%'))
end, { desc = 'Run makefile for current file' })
```

Source: vim.fandom.com

13.28 Create Convenient Shell Aliases

Category: command_line
Tags: shell-alias, workflow, productivity

Set up shell aliases to quickly open files in existing Vim instances with custom server names

Example: Example (Vim)

```
alias gvir="gvim --remote"
alias gvdev="gvim --servername dev --remote"
alias gvlib="gvim --servername lib --remote"
```

Example: Example (Neovim)

```
-- These are shell aliases, not Lua code
-- Add to your .bashrc or .zshrc
-- alias gvir='nvim --remote'
-- alias gvdev='nvim --servername dev --remote'
-- alias gvlib='nvim --servername lib --remote'
```

Source: vim.fandom.com

13.29 Create Custom Count Command

Category: command_line
Tags: custom-command, search, productivity

Add a custom command to easily count pattern occurrences

Example: Example (Vim)

```
command -nargs=1 Count :%s/<args>//gn
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Count', ':%s/<args>//gn', {})
```

Source: vim.fandom.com

13.30 Create Custom Vim Commands with Autocompletion

Category: command_line
Tags: custom-commands, autocompletion, plugin-development

Demonstrates creating a user-defined command with range support, parameter passing, and autocompletion

Example: Example (Vim)

```
" Create a custom command with autocompletion
command! -range=% -nargs=1 -complete=file MyCommand <line1>,<line2>call
 ↪  s:MyCommandFunction(<f-args>)

function! s:MyCommandFunction(...) range
  split
  execute "norm! " . a:firstline . "GV"
  execute "norm! " . a:lastline . 'G"ay'
  enew
  norm! "ap
  exe "sav! " . a:1
  q
endfunction
```

Example: Example (Neovim)

```
-- Lua equivalent for creating a custom command
vim.api.nvim_create_user_command('MyCommand', function(opts)
  local start_line = opts.line1
  local end_line = opts.line2
  local filename = opts.args

  -- Create a new buffer and copy selected lines
  local bufnr = vim.api.nvim_create_buf(true, true)
  vim.api.nvim_set_current_buf(bufnr)

  -- Get the lines from the original buffer
  local lines = vim.api.nvim_buf_get_lines(0, start_line - 1, end_line,
   ↪  false)
  vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, lines)

  -- Save the buffer to the specified filename
  vim.cmd('write! ' .. filename)
  vim.cmd('bdelete')
end, {
  nargs = 1,
  range = true,
  complete = 'file'
})
```

Source: vim.fandom.com

13.31 Custom Make Command with Quickfix

Category: command_line
Tags: custom-command, build, workflow

Create a custom :Make command that runs make and automatically opens quickfix window if errors exist

Example: Example (Vim)

```
:command -nargs=* Make make <args> | cwindow 3
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Make', function(opts)
  vim.cmd('make ' .. opts.args)
  vim.cmd('cwindow 3')
end, { nargs = '*' })
```

Source: vim.fandom.com

13.32 Disable Command-Line Abbreviations Temporarily

Category: command_line
Tags: abbreviations, command-mode, workaround

Two methods to temporarily prevent command abbreviations from expanding: using paste mode or literal character insertion

Example: Example (Vim)

```
" Method 1: Toggle paste mode
set invpaste

" Method 2: Literal character insertion
" Press <C-V> before the expansion character
```

Example: Example (Neovim)

```
-- Method 1: Toggle paste mode
vim.opt.paste = not vim.opt.paste:get()

-- Method 2: Use Vim's built-in literal insertion
-- Press <C-V> before the expansion character
```

Source: vim.fandom.com

13.33 Dynamically Replace Grep with Custom Search Program

Category: command_line
Tags: search, custom-commands, quickfix

Create flexible custom commands that temporarily replace grep with alternative search tools like cscope or lid, allowing quick exploration of code without permanent configuration changes

Example: Example (Vim)

```
fu! Mycscope(func)
  let tmp1=&grepprg
  let tmp2=&grepformat
  set grepformat=%f\ %*[a-zA-Z_0-9]\ %l\ %m
  set grepprg=cscope\ -R\ -L\ -3
  exe "grep ".a:func
  let &grepprg=tmp1
  let &grepformat=tmp2
endf
command -nargs=* CScope :silent call Mycscope("<args>")
```

Example: Example (Neovim)

```
function _G.mycscope(func)
  local tmp1 = vim.o.grepprg
  local tmp2 = vim.o.grepformat
  vim.o.grepformat = '%f %*[a-zA-Z_0-9] %l %m'
  vim.o.grepprg = 'cscope -R -L -3'
  vim.cmd('grep ' .. func)
  vim.o.grepprg = tmp1
  vim.o.grepformat = tmp2
end

vim.api.nvim_create_user_command('CScope', function(opts)
  _G.mycscope(opts.args)
end, { nargs = '*' })
```

Source: vim.fandom.com

13.34 Easily Capture Shell Command Output in Buffer

Category: command_line
Tags: shell-commands, external-tools, scripting

Provides multiple methods to insert shell command output into Vim/Neovim buffers, with flexible insertion techniques

Example: Example (Vim)

```
" Command to create a new scratch buffer with command output
:command! -nargs=* -complete=shellcmd R new | setlocal buftype=nofile
↪  bufhidden=hide noswapfile | r !<args>
```

Example: Example (Neovim)

```
-- Lua equivalent for creating a command to capture shell output
vim.api.nvim_create_user_command('R', function(ctx)
  vim.cmd('new')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'hide'
  vim.bo.swapfile = false
  vim.cmd('r !' .. ctx.args)
end, { nargs = '*', complete = 'shellcmd' })
```

Source: vim.fandom.com

13.35 Enhanced Command Window Navigation

Category: command_line
Tags: command-line, navigation, productivity

Improve command window functionality with custom mappings for easier navigation and command recall

Example: Example (Vim)

```
" Enter command window with single keystroke
nmap <Esc> q:<C-W>_
nmap q/ q/<C-W>_
nmap q? q?<C-W>_

" Custom autocommands for command window
augroup ECW_au
  au!
  au CmdwinEnter * nmap <Esc> :q<CR>
  au CmdwinLeave * nmap <Esc> q:<C-W>_
augroup END
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Esc>', 'q:<C-W>_', { desc = 'Open command window' })
vim.keymap.set('n', 'q/', 'q/<C-W>_', { desc = 'Open search command window'
↪  })
vim.keymap.set('n', 'q?', 'q?<C-W>_', { desc = 'Open reverse search command
↪  window' })

-- Create augroup for command window behavior
local augroup = vim.api.nvim_create_augroup('ECW_au', { clear = true })
```

```
vim.api.nvim_create_autocmd('CmdwinEnter', {
  group = augroup,
  callback = function()
    vim.keymap.set('n', '<Esc>', ':q<CR>', { buffer = true })
  end
})

vim.api.nvim_create_autocmd('CmdwinLeave', {
  group = augroup,
  callback = function()
    vim.keymap.set('n', '<Esc>', 'q:<C-W>_', { buffer = true })
  end
})
```

Source: vim.fandom.com

13.36 Enhanced Command-Line Completion

Category: command_line
Tags: completion, command-line, usability

Improve command-line completion with a menu and enhanced tab behavior, showing all possible completions and automatically completing to the longest common string

Example: Example (Vim)

```
set wildmenu
set wildmode=list:longest,full
```

Example: Example (Neovim)

```
vim.opt.wildmenu = true
vim.opt.wildmode = 'list:longest,full'
```

Source: vim.fandom.com

13.37 Escape Shell Command Special Characters

Category: command_line
Tags: shell, command-escaping, external-commands

Properly escape shell command special characters to prevent errors when running external commands

Example: Example (Vim)

```
" Use shellescape() for commands with special characters
let escaped_cmd = shellescape(original_cmd)
```

Example: Example (Neovim)

```
-- Use vim.fn.shellescape() to escape special characters
local escaped_cmd = vim.fn.shellescape(original_cmd)
```

Source: vim.fandom.com

13.38 Escape Shell Commands Properly

Category: command_line
Tags: shell-commands, escaping

Use shellescape() to properly escape special characters in external shell commands to prevent errors

Example: Example (Vim)

```
" Example: Escape command with special characters
let escaped_cmd = shellescape(original_cmd)
```

Example: Example (Neovim)

```
-- Use vim.fn.shellescape() to escape shell commands
local escaped_cmd = vim.fn.shellescape(original_cmd)
```

Source: vim.fandom.com

13.39 Execute Commands Across Line Ranges

Category: command_line
Tags: range-command, global-command, advanced-editing

Apply any command to a specific line range using global command with empty pattern

Example: Example (Vim)

```
:n1,n2 g/^/ command
```

Example: Example (Neovim)

```
-- Lua equivalent would typically be a function or use vim.cmd
-- Example: Execute command on lines 5-10
vim.cmd('5,10g/^/ echo "Processing line"')
```

Source: vim.fandom.com

13.40 Execute Shell Command in Explorer Directory

Category: command_line
Tags: shell-command, file-explorer, custom-function

Execute a shell command in the directory currently shown in the file explorer without changing the current working directory

Example: Example (Vim)

```
function! MyFileHandler(filename)
  let oldpath = getcwd()
  let currentdirectory = ""
  if(isdirectory(a:filename))
    let currentdirectory = strpart(a:filename, 0, strlen(a:filename) - 1)
  else
    let currentdirectory = a:filename
  endif
  let lastslash = strridx(currentdirectory, "/")
  let currentdirectory = strpart(currentdirectory, 0, lastslash)
  let usercommand = input(currentdirectory . "# ")
  if(strlen(usercommand) > 0)
    execute "cd " . currentdirectory
    execute "!" . usercommand
    execute "cd " . oldpath
  endif
endfunction

let g:explFileHandler = "MyFileHandler"
```

Example: Example (Neovim)

```
function _G.my_file_handler(filename)
  local oldpath = vim.fn.getcwd()
  local currentdirectory = ""

  if vim.fn.isdirectory(filename) == 1 then
    currentdirectory = filename:sub(1, -2)
  else
    currentdirectory = filename
  end

  local lastslash = currentdirectory:match(".*()/.+$") or 0
  currentdirectory = currentdirectory:sub(1, lastslash)
```

```
  local usercommand = vim.fn.input(currentdirectory .. "# ")

  if #usercommand > 0 then
    vim.cmd('cd ' .. currentdirectory)
    vim.cmd('!' .. usercommand)
    vim.cmd('cd ' .. oldpath)
  end
end

-- Assuming you're using explorer.vim or similar plugin
vim.g.explFileHandler = 'my_file_handler'
```

Source: vim.fandom.com

13.41 Find Perl Path Dynamically

Category: command_line
Tags: system-command, path-discovery

Quickly insert the path to perl executable using which command

Example: Example (Vim)

```
:r !which perl
```

Example: Example (Neovim)

```
vim.cmd('r !which perl')
```

Source: vim.fandom.com

13.42 Fix Common Command-Line Typos

Category: command_line
Tags: typo-correction, command-line, productivity

Automatically correct common command-line typos like accidentally holding Shift or mistyping quit/write commands

Example: Example (Vim)

```
" Correct common command typos
cabbrev Q quit
cabbrev W write
cabbrev q!@ q!
cabbrev wq!@ wq!
```

Example: Example (Neovim)

```
-- Correct common command typos in Neovim
vim.cmd('cabbrev Q quit')
vim.cmd('cabbrev W write')
vim.cmd('cabbrev q!@ q!')
vim.cmd('cabbrev wq!@ wq!')
```

Source: vim.fandom.com

13.43 Flexible Line Range Operations in Vim

Category: command_line
Tags: range, text-manipulation, search-replace

Use flexible range specifiers for powerful text editing across multiple lines, including marks, line numbers, and search patterns

Example: Example (Vim)

```
" Examples of range operations
:21,25s/old/new/g     " Substitute in lines 21-25
:%s/old/new/g         " Substitute in all lines
:'a,'bd               " Delete lines from mark a to b
:/apples/,/peaches/s/^/# /g  " Comment lines between patterns
```

Example: Example (Neovim)

```
-- Lua equivalents use ex commands
vim.cmd('21,25s/old/new/g')      -- Substitute in lines 21-25
vim.cmd('%s/old/new/g')          -- Substitute in all lines
vim.cmd("'a,'bd")                 -- Delete lines from mark a to b
vim.cmd('/apples/,/peaches/s/^/# /g')  -- Comment lines between patterns
```

Source: vim.fandom.com

13.44 Fuzzy File Name Completion

Category: command_line
Tags: completion, file-search, navigation

Use tab and Ctrl-D for dynamic file name completion and listing

Example: Example (Vim)

```
:e <space>Ctrl-D  # List all files
:e get<Tab>      # Complete files starting with 'get'
```

Example: Example (Neovim)

```
-- These are built-in Vim/Neovim command line completion features
-- Use in command mode with Ctrl-D or Tab
```

Source: vim.fandom.com

13.45 Get Shell Command for Current Directory

Category: command_line
Tags: directory, shell-integration, clipboard

Generate a shell 'cd' command for the current Vim working directory, which can be easily pasted into a terminal

Example: Example (Vim)

```
com Scd let @+="cd \"" . escape(getcwd(), "\"") . """
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Scd', function()
  local cwd = vim.fn.getcwd()
  vim.fn.setreg('+', 'cd "' .. cwd:gsub('"', '\"%') .. '"')
end, {})
```

Source: vim.fandom.com

13.46 Get VIMRUNTIME Path in Bash Scripts

Category: command_line
Tags: shell-scripting, vim-environment

Retrieve the VIMRUNTIME directory path using Vim in a bash script, which can be useful for scripting and plugin development

Example: Example (Vim)

```
vim --cmd 'echo $VIMRUNTIME' --cmd 'quit' 2> /tmp/VIMRUNTIME.txt
VIMRUNTIME=`perl -pe 's/
//g' /tmp/VIMRUNTIME.txt`
rm -f /tmp/VIMRUNTIME.txt
```

Example: Example (Neovim)

```
-- Lua equivalent for getting VIMRUNTIME
local handle = io.popen('vim --cmd "echo $VIMRUNTIME" --cmd "quit"')
```

```
local vimruntime = handle:read('*a'):gsub('[\r\n]', '')
handle:close()
```

Source: vim.fandom.com

13.47 Insert Buffer Contents into Command Line

Category: command_line
Tags: command-line, text-insertion, productivity

Quickly insert text from the current buffer into command line using different methods

Example: Example (Vim)

```
" Method 1: Use Shift-Insert
" Method 2: Use Ctrl-R Ctrl-W to insert current word
```

Example: Example (Neovim)

```
-- Insert current word in command line
vim.keymap.set('c', '<C-r><C-w>', function()
  local word = vim.fn.expand('<cword>')
  vim.api.nvim_feedkeys(word, 'n', true)
end)
```

Source: vim.fandom.com

13.48 Keep Command Window Open After Execution

Category: command_line
Tags: automation, workflow, command-window

Create a mapping to execute a command and automatically reopen the command window, maintaining context

Example: Example (Vim)

```
" Reopen command window after executing a command
autocmd CmdwinEnter * nnoremap <buffer> <F5> :let
↪  g:CmdWindowLineMark=line('.')<CR><CR>q::execute 'normal
↪  '.g:CmdWindowLineMark.'G'<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for keeping command window open
vim.api.nvim_create_autocmd('CmdwinEnter', {
```

```
  callback = function()
    vim.keymap.set('n', '<F5>', function()
      -- Store current line
      vim.g.CmdWindowLineMark = vim.fn.line('.')
      -- Execute current line
      vim.cmd('normal! \<CR>')
      -- Reopen command window and go to previous line
      vim.cmd('q:')
      vim.cmd('normal! ' .. vim.g.CmdWindowLineMark .. 'G')
    end, { buffer = true })
  end
})
```

Source: vim.fandom.com

13.49 List and Filter Loaded Vim Scripts

Category: command_line
Tags: scripting, debugging, configuration

Easily list and filter loaded Vim scripts, allowing quick inspection of loaded plugins and scripts

Example: Example (Vim)

```
function! s:Filter_lines(cmd, filter)
  let save_more = &more
  set nomore
  redir => lines
  silent execute a:cmd
  redir END
  let &more = save_more
  new
  setlocal buftype=nofile bufhidden=hide noswapfile
  put =lines
  g/^\s*$/d
  %s/^\s*\d\+:\s*//e
  if !empty(a:filter)
    execute 'v/' . a:filter . '/d'
  endif
  0
endfunction
command! -nargs=? Scriptnames call s:Filter_lines('scriptnames', <q-args>)
```

Example: Example (Neovim)

```
function _G.filter_scripts(cmd, filter)
  local lines = vim.fn.execute(cmd)
  local filtered_lines = {}

  for _, line in ipairs(vim.split(lines, '\n')) do
```

```
    if filter == '' or line:find(filter) then
      table.insert(filtered_lines, line:gsub('^%s*%d+:%s*', ''))
    end
  end

  local buf = vim.api.nvim_create_buf(false, true)
  vim.api.nvim_buf_set_lines(buf, 0, -1, false, filtered_lines)
  vim.api.nvim_buf_set_option(buf, 'buftype', 'nofile')
  vim.api.nvim_buf_set_option(buf, 'bufhidden', 'hide')
  vim.api.nvim_buf_set_option(buf, 'swapfile', false)

  vim.api.nvim_command('buffer ' .. buf)
end

vim.api.nvim_create_user_command('Scriptnames', function(opts)
  _G.filter_scripts('scriptnames', opts.args or '')
end, { nargs = '?' })
```

Source: vim.fandom.com

13.50 Manipulate Environment Variables in Vim

Category: command_line
Tags: environment, variables, shell-interaction

Easily read, modify, and insert environment variables directly within Vim

Example: Example (Vim)

```
" Display PATH variable
:echo $PATH

" Assign PATH to a Vim variable
:let myvar = $PATH

" Modify PATH for current session
:let $PATH = '/foo:/bar'

" Append to PATH
:let $PATH .= ':/foo:/bar'
```

Example: Example (Neovim)

```
-- Display PATH variable
print(vim.env.PATH)

-- Assign PATH to a variable
local myvar = vim.env.PATH

-- Modify PATH for current session
vim.env.PATH = '/foo:/bar'
```

```
-- Append to PATH
vim.env.PATH = vim.env.PATH .. ':/foo:/bar'
```

Source: vim.fandom.com

13.51 Mass File Editing with Arguments

Category: command_line
Tags: bulk-edit, search-replace, productivity

Perform find and replace operations across multiple files efficiently

Example: Example (Vim)

```
vim -c "argdo %s/ABC/DEF/g | w" *.txt
vim -c "argdo %s/FOO/BAR/g | update" `grep -l FOO *`
```

Example: Example (Neovim)

```
-- Lua equivalent using nvim-spectre or manual approach
-- Requires external grep and argument manipulation
local files = vim.fn.systemlist('grep -l FOO *')
for _, file in ipairs(files) do
  vim.cmd('edit ' .. file)
  vim.cmd(':%s/FOO/BAR/g')
  vim.cmd('write')
end
```

Source: vim.fandom.com

13.52 Master Command-Line History Navigation

Category: command_line
Tags: history, navigation, productivity

Efficiently navigate and reuse command and search history using command-line window and history keys

Example: Example (Vim)

```
" Open command-line window for commands
nnoremap : q:i

" Open command-line window for searches
nnoremap / q/i

" Open command-line window for reverse searches
```

```
nnoremap ? q?i
```

Example: Example (Neovim)

```
-- Open command-line window for commands
vim.keymap.set('n', ':', function() vim.cmd('q:') end, { desc = 'Open
↪  command history window' })

-- Open command-line window for searches
vim.keymap.set('n', '/', function() vim.cmd('q/') end, { desc = 'Open search
↪  history window' })

-- Open command-line window for reverse searches
vim.keymap.set('n', '?', function() vim.cmd('q?') end, { desc = 'Open
↪  reverse search history window' })
```

Source: vim.fandom.com

13.53 Perl Syntax Checking in Vim

Category: command_line
Tags: perl, syntax-checking, error-handling

Configures Vim to run Perl syntax checks and parse errors, allowing quick navigation to error locations

Example: Example (Vim)

```
" Compiler configuration for Perl
setlocal errorformat=%f:%l:\ %m
setlocal makeprg=perl\ -MVimCompile\ -c\ %
```

Example: Example (Neovim)

```
-- Lua equivalent for Perl syntax checking
vim.opt.errorformat = '%f:%l: %m'
vim.opt.makeprg = 'perl -MVimCompile -c %'
```

Source: vim.fandom.com

13.54 Perl-Based Command Evaluation in Vim

Category: command_line
Tags: perl, evaluation, command-extension

Add a custom command to evaluate Perl expressions directly in Vim, allowing quick calculations and code execution

Example: Example (Vim)

```
:command! -nargs=+ Evaluate :perl VIM::Msg(eval{<args>})
```

Example: Example (Neovim)

```
-- Lua equivalent would require a different approach
-- In Neovim, you might use vim.fn.system() or a Lua-based implementation
-- Note: Direct Perl integration may require additional plugin or
↪  configuration
```

Source: vim.fandom.com

13.55 Powerful Command-Line History Navigation

Category: command_line
Tags: command-history, search-history, productivity

Use command-line window to easily edit and repeat previous commands or searches with full Vim editing capabilities

Example: Example (Vim)

```
" Map : and / to open command window directly
nnoremap : q:i
nnoremap / q/i
nnoremap ? q?i
```

Example: Example (Neovim)

```
-- Open command window directly in Neovim
vim.keymap.set('n', ':', function() vim.cmd('q:i') end, { desc = 'Open
↪  command window' })
vim.keymap.set('n', '/', function() vim.cmd('q/i') end, { desc = 'Open
↪  search command window' })
vim.keymap.set('n', '?', function() vim.cmd('q?i') end, { desc = 'Open
↪  reverse search command window' })
```

Source: vim.fandom.com

13.56 Powerful Global Command for Line Matching

Category: command_line
Tags: search, pattern-matching, editing

Use :g command to apply actions to lines matching a pattern

Example: Example (Vim)

```
:g/pattern/d  " Delete all lines matching pattern
```

Example: Example (Neovim)

```
-- Can be used directly in command mode
-- For programmatic approach:
vim.cmd('g/pattern/d')
```

Source: vim.fandom.com

13.57 Powerful Range-Based Text Editing

Category: command_line
Tags: editing, ranges, search-replace

Use flexible range specifiers for precise text manipulation across lines

Example: Example (Vim)

```
:%s/old/new/g                  " Replace in all lines
:21,25s/old/new/g          " Replace in specific line range
:.,$s/old/new/g            " Replace from current line to end
:/pattern/,/endpattern/ s/old/new/g  " Replace between patterns
```

Example: Example (Neovim)

```
-- Lua equivalent uses Vim command mode
vim.cmd(':%s/old/new/g')                -- Replace in all lines
vim.cmd(':21,25s/old/new/g')            -- Replace in specific line range
vim.cmd(':.,$s/old/new/g')              -- Replace from current line to end
vim.cmd(':/pattern/,/endpattern/ s/old/new/g')  -- Replace between patterns
```

Source: vim.fandom.com

13.58 Quick Buffer and Global Commands

Category: command_line
Tags: buffer-management, global-commands, text-manipulation

Apply commands across all buffers or matching lines efficiently

Example: Example (Vim)

```
:bufdo %s/pattern/substitution/g  # Replace in all buffers
:g/pattern/d  # Delete all lines matching pattern
```

Example: Example (Neovim)

```
-- For buffer-wide substitution
vim.cmd('bufdo %s/pattern/substitution/g')

-- For deleting lines matching pattern
vim.cmd('g/pattern/d')
```

Source: vim.fandom.com

13.59 Quick Compile Current File with GCC

Category: command_line
Tags: compilation, gcc, build-tools

Configure Vim to compile the current file quickly using gcc with a single command

Example: Example (Vim)

```
set makeprg=gcc\ -o\ %<\ %
```

Example: Example (Neovim)

```
vim.opt.makeprg = 'gcc -o %< %'
```

Source: vim.fandom.com

13.60 Quick Copy and Paste with pbcopy/pbpaste

Category: command_line
Tags: macos, clipboard, file-operations

Use macOS pbcopy and pbpaste commands directly from Vim for system clipboard operations

Example: Example (Vim)

```
" Copy current line to clipboard
:.!pbcopy

" Copy specific lines to clipboard
:4,8!pbcopy

" Copy current filename to clipboard
:!echo "%:p" | pbcopy

" Paste clipboard content to current line
:r !pbpaste
```

Example: Example (Neovim)

```
-- These can typically be used directly in Neovim's command mode as well
```

Source: vim.fandom.com

13.61 Quick File Opening with Shell Command Shortcuts

Category: command_line
Tags: shell-integration, file-navigation, productivity

Quickly open the most recently modified file or files matching specific criteria using shell aliases and functions

Example: Example (Vim)

```
# Open most recent file
alias vew='vi `ls -t * | head -1`

# Open files containing a specific substring
function v() {
  vim *${1}*
}
```

Example: Example (Neovim)

```
-- Lua equivalents typically done via external shell functions
-- In Neovim, you can use telescope.nvim for similar functionality
-- Example telescope mapping
vim.keymap.set('n', '<leader>fr', require('telescope.builtin').oldfiles, {
↪  desc = 'Recent files' })
```

Source: vim.fandom.com

13.62 Quick File Peek Without Opening Buffer

Category: command_line
Tags: file-preview, quick-view, vim-tricks

View file contents directly in Vim's command line without opening a new buffer, useful for quick file inspection

Example: Example (Vim)

```
:echo join(readfile('foo.bat'), "\n")
```

Example: Example (Neovim)

```
vim.api.nvim_command(string.format('echo join(readfile("%s"), "\n")',
↪  'foo.bat'))
```

Source: vim.fandom.com

13.63 Quick Firefox URL Navigation from Vim

Category: command_line
Tags: web-development, browser-integration, productivity

Add custom commands to quickly open URLs in Firefox directly from Vim

Example: Example (Vim)

```
command! -nargs=1 Repl silent !echo \
      "repl.home();
      content.location.href = '<args>';
      repl.enter(content);
      repl.quit();" |
      nc localhost 4242

nmap <leader>mh :Repl http://
nmap <silent> <leader>ml :Repl file:///%:p<CR>
nmap <silent> <leader>md :Repl http://localhost/
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Repl', function(opts)
  vim.fn.system(string.format(
    'echo "repl.home(); content.location.href = \'%s\'; ' ..
    'repl.enter(content); repl.quit();" | nc localhost 4242',
    opts.args
  ))
end, { nargs = 1 })

vim.keymap.set('n', '<leader>mh', ':Repl http://', { desc = 'Open URL in
↪  Firefox' })
vim.keymap.set('n', '<leader>ml', ':Repl file://' .. vim.fn.expand('%:p') ..
↪  '<CR>', { desc = 'Open current file in Firefox' })
vim.keymap.set('n', '<leader>md', ':Repl http://localhost/<CR>', { desc =
↪  'Open localhost in Firefox' })
```

Source: vim.fandom.com

13.64 Quick Launch Browser from Vim

Category: command_line
Tags: windows, external-commands, productivity

Quickly open a URL in the default browser directly from Vim using the :!start explorer command

Example: Example (Vim)

```
:!start explorer http://www.vim.org/
```

Example: Example (Neovim)

```
vim.fn.system('start explorer http://www.vim.org/')
```

Source: vim.fandom.com

13.65 Quick Number Conversion with User Commands

Category: command_line
Tags: number-conversion, utility-commands

Define user commands to convert numbers between decimal and hexadecimal in visual or normal mode

Example: Example (Vim)

```
command! -nargs=? -range Dec2hex call s:Dec2hex(<line1>, <line2>, '<args>')
command! -nargs=? -range Hex2dec call s:Hex2dec(<line1>, <line2>, '<args>')
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Dec2hex', function(opts)
  local start_line = opts.line1
  local end_line = opts.line2
  local arg = opts.args
  -- Implement conversion logic
end, { nargs = '?', range = true })

vim.api.nvim_create_user_command('Hex2dec', function(opts)
  local start_line = opts.line1
  local end_line = opts.line2
  local arg = opts.args
  -- Implement conversion logic
end, { nargs = '?', range = true })
```

Source: vim.fandom.com

13.66 Quick PHP Syntax Checking in Vim/Neovim

Category: command_line
Tags: php, syntax-checking, linting

Quickly check PHP syntax directly from the editor without leaving the buffer

Example: Example (Vim)

```
map <C-B> :!php -l %<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-b>', ':!php -l %<CR>', { desc = 'Check PHP syntax'
↪  })
```

Source: vim.fandom.com

13.67 Quick Postgres Query Execution in Vim

Category: command_line
Tags: database, external-command, productivity

Execute PostgreSQL queries directly from Vim by mapping a key to run the current file as a database query

Example: Example (Vim)

```
map <F9> :!psql -d yourdb < % <BAR> less
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F9>', function()
  vim.cmd('!psql -d yourdb < % | less')
end, { desc = 'Run current file as Postgres query' })
```

Source: vim.fandom.com

13.68 Quick Vim Help in Full Window

Category: command_line
Tags: help, command-line, documentation

Quickly open Vim help for a specific topic in a full window from the command line

Example: Example (Vim)

```
#!/bin/bash
vim -c "help $1" -c only
```

Example: Example (Neovim)

```
-- Lua equivalent (shell script)
local function open_help(topic)
  vim.cmd(string.format('help %s', topic))
  vim.cmd('only')
end
```

Source: vim.fandom.com

13.69 Quick Word/WORD Paste in Command Line

Category: command_line
Tags: text-objects, command-line, navigation

Easily paste word or WORD under cursor into command line using keyboard shortcuts

Example: Example (Vim)

```
" <C-r><C-w> - Paste word under cursor
" <C-r><C-a> - Paste WORD under cursor
```

Example: Example (Neovim)

```
-- Functionality is built-in, no specific Lua implementation needed
-- Can be used directly in command and search modes
```

Source: vim.fandom.com

13.70 Quick XML Validation with xmllint

Category: command_line
Tags: xml, validation, shell-command

Quickly check XML document well-formedness directly from Vim using xmllint

Example: Example (Vim)

```
!xmllint --noout %
```

Example: Example (Neovim)

```
vim.cmd('!xmllint --noout %')
```

Source: vim.fandom.com

13.71 Quick XML Wellformedness Check

Category: command_line
Tags: xml, validation, external-tool

Validate XML file's wellformedness directly from Vim using xmllint

Example: Example (Vim)

```
!xmllint --noout %
```

Example: Example (Neovim)

```
vim.cmd('!xmllint --noout %')
```

Source: vim.fandom.com

13.72 Quickly Describe SQL Table from Vim

Category: command_line
Tags: sql, external-command, productivity

Use a custom bash script to retrieve and insert table description directly into Vim buffer

Example: Example (Vim)

```
:r !describe <tableName>
```

Example: Example (Neovim)

```
-- Equivalent Lua command
vim.api.nvim_command('r !describe ' .. table_name)
```

Source: vim.fandom.com

13.73 Quickly Find Full Command Path in Scripts

Category: command_line
Tags: shell-integration, command-lookup, productivity

Replace a command with its full system path using 'which' directly in Vim, useful for scripting and ensuring absolute command references

Example: Example (Vim)

```
!which $(cat)
```

Example: Example (Neovim)

```
-- Lua equivalent for finding command full path
local function get_command_path(cmd)
  local handle = io.popen('which ' .. cmd)
  local result = handle:read('*a'):gsub('%s+', '')
  handle:close()
  return result
end

-- Usage example
-- vim.fn.setline('.', get_command_path('cat'))
```

Source: vim.fandom.com

13.74 Quickly Inspect Vim Environment

Category: command_line
Tags: inspection, diagnostics, environment

Comprehensive set of commands to display various Vim environment details, useful for troubleshooting and understanding current configuration

Example: Example (Vim)

```
" Example commands for environment inspection
:set all      " Show all options
:map          " Show all mappings
:scriptnames  " List sourced scripts
```

Example: Example (Neovim)

```
-- Lua equivalents for environment inspection
vim.cmd('set all')      -- Show all options
vim.cmd('map')          -- Show all mappings
vim.cmd('scriptnames')  -- List sourced scripts

-- More Neovim-specific approach
print(vim.inspect(vim.opt))  -- Inspect options
print(vim.inspect(vim.keymap.list()))
```

Source: vim.fandom.com

13.75 Quickly Navigate Directories from Command Line

Category: command_line
Tags: navigation, productivity, file-operations

Custom mappings to quickly access current file's directory and navigate parent direc-

tories from command line, useful for large projects

Example: Example (Vim)

```
func! CurrentFileDir()
  return "e " . expand("%:p:h") . "/"
endfunc

func! DeleteTillSlash()
  let cmd = getcmdline()
  let cmd_edited = substitute(cmd, "\\(.*/\\).*", "\\1", "")
  if cmd == cmd_edited
    let cmd_edited = substitute(cmd, "\\(.*/\\).*/", "\\1", "")
  endif
  return cmd_edited
endfunc

cno $c e <C-\>eCurrentFileDir()<CR>
cmap <C-q> <C-\>eDeleteTillSlash()<CR>
```

Example: Example (Neovim)

```
local function current_file_dir()
  return 'e ' .. vim.fn.expand('%:p:h') .. '/'
end

local function delete_till_slash()
  local cmd = vim.fn.getcmdline()
  local cmd_edited = cmd:gsub('(.*/)[^/]*$', '%1')
  if cmd == cmd_edited then
    cmd_edited = cmd:gsub('(.*/).*/', '%1')
  end
  return cmd_edited
end

vim.api.nvim_create_user_command('CurrentFileDir', current_file_dir, {})
vim.keymap.set('c', '$c', function()
  vim.cmd('e ' .. vim.fn.expand('%:p:h') .. '/')
end)

vim.keymap.set('c', '<C-q>', function()
  local cmd = vim.fn.getcmdline()
  local new_cmd = cmd:gsub('(.*/)[^/]*$', '%1')
  vim.fn.setcmdline(new_cmd)
end)
```

Source: vim.fandom.com

13.76 Quickly Open Files with Whereis Command

Category: command_line
Tags: shell-function, file-navigation, command-line-tools

Create a shell function to quickly open files found by the whereis command in Vim/Neovim

Example: Example (Vim)

```
function vvim() { vim `whereis $1|cut -d: -f2` }
function ggvim() { gvim `whereis $1|cut -d: -f2` }
```

Example: Example (Neovim)

```
-- Bash function to be added to .bashrc or .bash_profile
-- function vvim() {
--   nvim `whereis "$1" | cut -d: -f2`
-- }

-- In Neovim, you could create a similar command
vim.api.nvim_create_user_command('VVim', function(opts)
  local file = vim.fn.system('whereis ' .. opts.args .. ' | cut -d:
  ↪  -f2'):gsub('%s+', '')
  vim.cmd('edit ' .. file)
end, { nargs = 1 })
```

Source: vim.fandom.com

13.77 Quickly Repeat Previous Command

Category: command_line
Tags: productivity, key-mapping, command-repeat

Easily re-run the last command using a leader key mapping

Example: Example (Vim)

```
nmap <Leader>. :<C-P><CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>.', function()
  vim.cmd(':<C-P><CR>')
end, { desc = 'Repeat previous command' })
```

Source: vim.fandom.com

13.78 Read and Modify Environment Variables in Vim

Category: command_line
Tags: environment, variables, shell-interaction

Easily read, display, and modify environment variables directly within Vim, allowing for

dynamic shell configuration

Example: Example (Vim)

```
" Display PATH variable
:echo $PATH

" Assign PATH to a Vim variable
:let myvar = $PATH

" Modify PATH for current session
:let $PATH = '/foo:/bar'

" Append to PATH
:let $PATH .= ':/foo:/bar'
```

Example: Example (Neovim)

```
-- Display PATH variable
print(vim.env.PATH)

-- Assign PATH to a variable
local myvar = vim.env.PATH

-- Modify PATH for current session
vim.env.PATH = '/foo:/bar'

-- Append to PATH
vim.env.PATH = vim.env.PATH .. ':/foo:/bar'
```

Source: vim.fandom.com

13.79 Remote Vim Session Control via Client-Server

Category: command_line
Tags: remote-editing, client-server, vim-management

Remotely save and close Vim sessions using the client-server protocol, useful for managing Vim instances across different machines

Example: Example (Vim)

```
vim --servername GVIM --remote-send '<Esc>:wqa<CR>'
```

Example: Example (Neovim)

```
-- Lua equivalent for remote Vim interaction
-- Requires setting up Vim's client-server functionality
-- Example command to be run in terminal
-- vim.fn.remote_send('GVIM', '<Esc>:wqa<CR>')
```

Source: vim.fandom.com

13.80 Remove Last Path Component in Command Line

Category: command_line
Tags: command-line, file-navigation, productivity

Easily remove the last path component when navigating directories in command line, making file path editing more efficient

Example: Example (Vim)

```
cnoremap <C-t> <C-\>e(<SID>RemoveLastPathComponent())<CR>
function! s:RemoveLastPathComponent()
  return substitute(getcmdline(), '\%(\ \|[\/]\@!\f\)\+[\/]\=$\|.$', '', '')
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('c', '<C-t>', function()
  local cmdline = vim.fn.getcmdline()
  local modified_cmdline = cmdline:gsub('[^/\\]+[/\\]?$', '')
  return modified_cmdline
end, { expr = true, replace_keycodes = true })
```

Source: vim.fandom.com

13.81 Repeat Ex Command on Multiple Blocks

Category: command_line
Tags: global-command, text-manipulation, bulk-editing

Use the global command to apply an Ex command (like sort) to multiple blocks of text separated by blank lines

Example: Example (Vim)

```
:g/^\s*$/;//-1sort
```

Example: Example (Neovim)

```
-- Sort each block of text in a file
vim.cmd('g/^\s*$/;//-1sort')
```

Source: vim.fandom.com

13.82 Repeat Ex Commands with Single Keystroke

Category: command_line
Tags: command-repeat, ex-mode, productivity

Quickly repeat the last Ex (colon) command using @: and repeat again with @@

Example: Example (Vim)

```
" Press @: to repeat last Ex command
" Press @@ to repeat again
```

Example: Example (Neovim)

```
-- No direct Lua equivalent, but can be mapped similarly
vim.keymap.set('n', '<leader>@', '@:', { desc = 'Repeat last Ex command' })
```

Source: vim.fandom.com

13.83 Repeat Last Ex Command Efficiently

Category: command_line
Tags: ex-commands, productivity, repeat

Use @: to repeat the last colon command, and @@ to repeat subsequent times. Useful for batch operations across files.

Example: Example (Vim)

```
" Repeat last colon command
@:
" Repeat again
@@
" Repeat multiple times
10@@
```

Example: Example (Neovim)

```
-- Repeat last command in Ex mode
vim.cmd('normal! @:')
-- Repeat again
vim.cmd('normal! @@')
-- Repeat multiple times
vim.cmd('normal! 10@@')
```

Source: vim.fandom.com

13.84 Replace Built-in Commands with Command Abbreviations

Category: command_line
Tags: command-abbreviation, customization, typo-correction

Create command abbreviations to replace or modify built-in Vim commands, useful for correcting typos or customizing command behavior

Example: Example (Vim)

```
" Function to create command abbreviations
function! CommandCabbr(abbreviation, expansion)
  execute 'cabbr ' . a:abbreviation . ' <c-r>=getcmdpos() == 1 &&
  ↪  getcmdtype() == ":" ? "' . a:expansion . '" : "' . a:abbreviation .
  ↪  '"<CR>'
endfunction
command! -nargs=+ CommandCabbr call CommandCabbr(<f-args>)

" Example: Replace 'w1' with 'w!' to force write
cabbrev w1 <c-r>=(getcmdtype()==':' && getcmdpos()==1 ? 'w!' : 'w1')<CR>
```

Example: Example (Neovim)

```
-- Function to create command abbreviations
local function command_cabbr(abbreviation, expansion)
  vim.api.nvim_create_user_command('CommandCabbr', function(opts)
    local cmd = string.format('cabbr %s <c-r>=(getcmdpos() == 1 &&
    ↪  getcmdtype() == ":" ? "%s" : "%s")<CR>',
      opts.args[1], opts.args[2], opts.args[1])
    vim.cmd(cmd)
  end, { nargs = '+' })
end

-- Example: Replace 'w1' with 'w!' to force write
vim.cmd('cabbrev w1 <c-r>=(getcmdtype()==":" && getcmdpos()==1 ? "w!" :
↪  "w1")<CR>')
```

Source: vim.fandom.com

13.85 Run Commands Within Current Function

Category: command_line
Tags: text-editing, function-scope, search-replace

Create a mapping to restrict commands like substitution to the current function body in C/C++/Java files

Example: Example (Vim)

```
:cmap ;tf ?^{??(?,/^}/
```

Example: Example (Neovim)

```
-- Lua equivalent requires more complex setup
vim.keymap.set('c', ';tf', '?^{??(?,/^}/', { desc = 'Select current function
↪  scope' })
```

Source: vim.fandom.com

13.86 Run Java Class from Current File

Category: command_line
Tags: java, execution, file-path

Quickly run the compiled Java class from the current file's directory

Example: Example (Vim)

```
map <F6> :!java -cp %:p:h %:t:r<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F6>', function()
  local classpath = vim.fn.expand('%:p:h')
  local classname = vim.fn.expand('%:t:r')
  vim.cmd('!java -cp ' .. classpath .. ' ' .. classname)
end, { desc = 'Run Java class' })
```

Source: vim.fandom.com

13.87 Selective Line Deletion with Exceptions

Category: command_line
Tags: global-command, filtering, text-manipulation

Delete lines matching a pattern while preserving specific exceptions using global command

Example: Example (Vim)

```
:global:^./help/:if (match(getline(line('.')), '^./help/de/') == -1) |
↪  delete | endif

" Alternative method:
:g#\(^\./help/\)\(de/\)\@!#d
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.api and global command
vim.cmd('g#\(^\./help/\)\(de/\)\@!#d')

-- More explicit Lua approach
for i = vim.fn.line('$'), 1, -1 do
  local line = vim.fn.getline(i)
  if line:match('^./help/') and not line:match('^./help/de/') then
    vim.cmd(i .. 'delete')
  end
end
```

Source: vim.fandom.com

13.88 Silent External Command Execution

Category: command_line
Tags: external-commands, workflow, productivity

Execute external commands silently without interrupting workflow

Example: Example (Vim)

```
command! -nargs=1 Silent
    execute 'silent !' . <q-args>
    | execute 'redraw!'
```

Example: Example (Neovim)

```
-- Create a silent command executor
vim.api.nvim_create_user_command('Silent', function(ctx)
    vim.fn.system(ctx.args)
    vim.cmd.redraw()
end, { nargs = 1 })
```

Source: vim.fandom.com

13.89 Source Vim Script Directly from Command Line

Category: command_line
Tags: cli, script-sourcing, startup

Use -S flag to source a Vim script when launching Vim, which is more concise than using -c option

Example: Example (Vim)

```
vim -S foobar.vim
```

Example: Example (Neovim)

```
vim.cmd('source foobar.vim')
```

Source: vim.fandom.com

13.90 Use Quickfix Window for Compile Errors

Category: command_line
Tags: compilation, error-handling, productivity

Use :cw to open a quickfix window that lists compilation errors and allows quick navigation between error locations

Example: Example (Vim)

```
set makeprg=javac
set makeef=c:\dev\src\errors.txt
set errorformat=%A%f:%l:\ %m,%C%m
noremap <M-1> :w<CR>:set ch=5<CR>:make -d C:\dev\classes %:p<CR>
noremap <M-2> :cp<CR>
noremap <M-3> :cn<CR>
noremap <M-4> :cl<CR>
```

Example: Example (Neovim)

```
vim.o.makeprg = 'javac'
vim.o.makeef = 'c:\\dev\\src\\errors.txt'
vim.o.errorformat = '%A%f:%l: %m,%C%m'

vim.keymap.set('n', '<M-1>', ':w<CR>:set ch=5<CR>:make -d C:\\dev\\classes
↪  %:p<CR>', { desc = 'Compile current file' })
vim.keymap.set('n', '<M-2>', ':cp<CR>', { desc = 'Previous error' })
vim.keymap.set('n', '<M-3>', ':cn<CR>', { desc = 'Next error' })
vim.keymap.set('n', '<M-4>', ':cl<CR>', { desc = 'List errors' })
```

Source: vim.fandom.com

13.91 Use Quickfix with Doxygen for Error Navigation

Category: command_line
Tags: quickfix, documentation, error-handling

Configure Vim's quickfix to navigate through Doxygen documentation generation errors, similar to compiler errors

Example: Example (Vim)

```
:set makeprg=doxygen
:make Doxyfile
```

Example: Example (Neovim)

```
vim.opt.makeprg = 'doxygen'
vim.cmd('make Doxyfile')
```

Source: vim.fandom.com

13.92 Verbose Command Execution Debugging

Category: command_line
Tags: debugging, verbosity, troubleshooting

Set verbose levels to get more detailed information about command execution

Example: Example (Vim)

```
:9verbose edit somefile.txt.gz
```

Example: Example (Neovim)

```
-- Neovim equivalent for verbose command execution
-- Use :9verbose before commands to get detailed debug information
```

Source: vim.fandom.com

CHAPTER 14

Command line (advanced)

14.1 Command line abbreviations and shortcuts

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

Create command line abbreviations for frequently used commands.

Example: Example (Vim)

```
: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
```

14.2 Command line advanced search operations

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

Perform sophisticated search operations from command line.

Example: Example (Vim)

```
: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
```

14.3 Command line advanced substitution techniques

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

Master advanced substitution patterns and techniques.

Example: Example (Vim)

```
:%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
```

14.4 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: Example (Vim)

```
: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
```

14.5 Command line completion customization

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

Customize command line completion behavior and appearance.

Example: Example (Vim)

```
: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
```

14.6 Command line conditional execution

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

Execute commands conditionally using expressions and logic.

Example: Example (Vim)

```
: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
```

14.7 Command line custom command creation

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

Create sophisticated custom commands with parameters and completion.

Example: Example (Vim)

```
" 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
```

14.8 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: Example (Vim)

```
: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
```

14.9 Command line environment variable integration

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

Work with environment variables and system integration.

Example: Example (Vim)

```
: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
```

14.10 Command line error handling

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

Handle errors gracefully in command line operations.

Example: Example (Vim)

```
: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
```

14.11 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. Use `<C-r><C-w>` in command-line mode to insert the word under the cursor, perfect for quick substitutions.

Example: Example (Vim)

```
" 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

" 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
```

14.12 Command line external command integration

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

Integrate external commands seamlessly with Vim command line.

Example: Example (Vim)

```
: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
```

14.13 Command line filename completion variations

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

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

Example: Example (Vim)

```
" 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
```

14.14 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: Example (Vim)

```
: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
```

14.15 Command line job control and async

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

Control background jobs and asynchronous operations.

Example: Example (Vim)

```
: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
```

14.16 Command line macro recording and playback

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

Record and replay command sequences for automation.

Example: Example (Vim)

```
: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
```

14.17 Command line range shortcuts

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

Use range shortcuts for efficient line selection in commands.

Example: Example (Vim)

```
:.,$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
```

14.18 Command line register manipulation

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

Access and manipulate registers from command line efficiently.

Example: Example (Vim)

```
" 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
```

14.19 Command line script execution

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

Execute scripts and source files with advanced options.

Example: Example (Vim)

```
: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
```

14.20 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: Example (Vim)

```
:%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
```

14.21 Command line terminal integration

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

Integrate terminal operations seamlessly with command line.

Example: Example (Vim)

```
: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
```

14.22 Command line window operations

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

Use command line window for advanced command editing and history.

Example: Example (Vim)

```
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 15
Community tips

15.1 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: Example (Vim)

```
" 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)
```

15.2 Buffer-specific settings

Category: Configuration
Tags: buffer, specific, settings, local

Use buffer-local settings and autocmds for file-type specific configurations and optimizations.

Example: Example (Vim)

```
: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
```

15.3 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: Example (Vim)

```
: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)
```

15.4 Command abbreviations

Category: Command Line
Tags: abbreviations, shortcuts, efficiency, typos

Use command abbreviations for frequently used commands and common typo corrections.

Example: Example (Vim)

```
:cabbrev W w
:cabbrev Wq wq
:cabbrev Q q
:cabbrev vsf vert sfind
:cabbrev ff find **/*
" Corrects common typos and creates shortcuts
```

15.5 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: Example (Vim)

```
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
```

15.6 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: Example (Vim)

```
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)
```

15.7 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: Example (Vim)

```
: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
```

15.8 Environment-aware configuration

Category: Configuration
Tags: environment, conditional, config, dotenv

Use environment variables and conditional loading for portable configurations across different machines.

Example: Example (Vim)

```
" 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
```

15.9 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: Example (Vim)

```
: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
```

15.10 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: Example (Vim)

```
: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
```

15.11 Insert mode navigation

Category: Insert
Tags: insert, navigation, movement, efficiency

Use insert mode navigation keys for efficient editing without leaving insert mode frequently.

Example: Example (Vim)

```
<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
```

15.12 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: Example (Vim)

```
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'
```

15.13 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: Example (Vim)

```
-- 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 }
})
```

15.14 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: Example (Vim)

```
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
```

15.15 Quick fold navigation

Category: Folding
Tags: fold, navigation, quick, movement

Use fold navigation commands for efficient code structure navigation and overview.

Example: Example (Vim)

```
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
```

15.16 Register operations mastery

Category: Registers
Tags: registers, operations, advanced, clipboard

Use register operations for sophisticated copy-paste workflows and text manipulation chains.

Example: Example (Vim)

```
"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
```

15.17 Session workflow optimization

Category: Session
Tags: session, workflow, project, management

Use session commands for project-based workflow management and context switching.

Example: Example (Vim)

```
: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
```

15.18 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: Example (Vim)

```
: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
```

15.19 Tab-based workflow

Category: Tabs
Tags: tabs, workflow, organization, navigation

Use tabs for logical grouping of related files and context-based editing workflows.

Example: Example (Vim)

```
: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 16
Completion

16.1 API/Module completion

Category: Completion
Tags: completion, api, modules, vim.api

Build completion for Vim/Neovim API functions and modules.

Example: Example (Neovim)

```
-- 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,
})
```

16.2 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: Example (Neovim)

```
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'
```

16.3 Cached completion for performance

Category: Completion
Tags: completion, cache, performance, optimization

Implement smart caching to avoid expensive re-computation on every keystroke.

Example: Example (Neovim)

```
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'
```

16.4 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: Example (Neovim)

```
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'
```

16.5 Completion with external command

Category: Completion
Tags: completion, external, command, integration

Integrate external tools or commands into completion sources.

Example: Example (Neovim)

```
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,
})
```

16.6 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: Example (Neovim)

```
-- 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'
```

16.7 Custom completion function basics

Category: Completion
Tags: completion, completefunc, custom, omnifunc

Create custom completion functions using `completefunc` or `omnifunc` for domain-specific completions.

Example: Example (Neovim)

```
-- 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
```

16.8 File path completion

Category: Completion
Tags: completion, files, paths, filesystem

Create custom file path completion that works in specific contexts.

Example: Example (Neovim)

```
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,
})
```

16.9 Fuzzy matching completion

Category: Completion
Tags: completion, fuzzy, matching, filter

Implement fuzzy matching for completion items for more flexible matching.

Example: Example (Neovim)

```
-- 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'
```

16.10 LSP-style completion with documentation

Category: Completion
Tags: completion, lsp, documentation, preview

Create rich completion items with documentation preview like LSP. print("Hello", "World") – Output: Hello World local mymod = require('mymodule')

Example: Example (Neovim)

```
local completion_items = {
  {
    label = "print",
    kind = "Function",
    detail = "print(...)",
    documentation = {
```

```
      kind = "markdown",
      value = [[
Print arguments to stdout.

**Parameters:**
- `...` - Values to print

**Example:**

local mymod = require('mymodule')
```

16.11 Multi-source completion chaining

Category: Completion
Tags: completion, chaining, multiple-sources, aggregation

Combine multiple completion sources into a single completion function.

Example: Example (Neovim)

```
-- 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'
```

16.12 Snippet-style completion

Category: Completion
Tags: completion, snippets, templates, user-data

Create completion items that insert templates using `user_data` for custom behavior.

Example: Example (Neovim)

```
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 17
Configuration

17.1 Add 'Edit with Vim' Context Menu in Windows

Category: configuration
Tags: windows, file-operations, context-menu

Manually add a context menu option to edit files with Vim in Windows Vista/7

Example: Example (Vim)

```
# Registry edit to add 'Edit with Vim' option
# Open regedit and navigate to HKEY_CLASSES_ROOT\*\shell
# Create new key 'Open with gVIm' with command:
# "C:\Program Files (x86)\Vim\vim72\gvim.exe" "%1"
```

Example: Example (Neovim)

```
-- For Neovim, this would typically be handled by external tools
-- Or can be scripted with vim.fn.system() for registry edits
-- Recommended to use Windows-specific tools or installers
```

Source: vim.fandom.com

17.2 Add Context Menu Option to Open Vim in Directory

Category: configuration
Tags: windows, file-explorer, integration

Add a right-click context menu option in Windows Explorer to open Vim in the current directory

Example: Example (Vim)

```
REGEDIT4

[HKEY_CLASSES_ROOT\Folder\shell\VimHere]
@="Open Vim here"

[HKEY_CLASSES_ROOT\Folder\shell\VimHere\command]
```

```
@="C:\\Program Files\\Vim\\vim72\\gvim.exe ."
```

Example: Example (Neovim)

```
-- For Neovim, this is typically handled by external tools or scripts
-- You might use a system-specific script or AutoHotkey to achieve similar
 ↪  functionality
-- Recommend using a cross-platform solution like creating a shell script or
 ↪  using file manager extensions
```

Source: vim.fandom.com

17.3 Add Modeline for Perl Batch Files

Category: configuration
Tags: modeline, filetype, perl

Add a vim modeline to specify filetype for individual Perl batch files

Example: Example (Vim)

```
# Add this as the first line of the file
@rem vim: set ft=perl:
```

Example: Example (Neovim)

```
-- Add this as the first line of the file
-- @rem vim: set ft=perl:
```

Source: vim.fandom.com

17.4 Add Vim Context Menu in Nemo File Manager

Category: configuration
Tags: file-integration, linux, system-config

Create a right-click menu action to open files directly in Vim from Nemo file manager in Linux Mint Cinnamon

Example: Example (Vim)

```
[Nemo Action]
Active=true
Name=Open with Vim
Comment=Open %N with Vim in a terminal
Exec=gnome-terminal --geometry=80x49 --working-directory=%P -e "vim %F"
Icon-Name=vim
```

```
Selection=s
Extensions=nodirs;
Dependencies=vim;
EscapeSpaces=true;
```

Example: Example (Neovim)

```
-- Lua equivalent: Create ~/.local/share/nemo/actions/vim.nemo_action
-- Use vim.fn.writefile() to create the file programmatically
local nemo_action_content = {
  '[Nemo Action]',
  'Active=true',
  'Name=Open with Vim',
  'Comment=Open %N with Vim in a terminal',
  'Exec=gnome-terminal --geometry=80x49 --working-directory=%P -e "vim %F"',
  'Icon-Name=vim',
  'Selection=s',
  'Extensions=nodirs;',
  'Dependencies=vim;',
  'EscapeSpaces=true;'
}
vim.fn.writefile(nemo_action_content,
 ↪  vim.fn.expand('~/.local/share/nemo/actions/vim.nemo_action'))
```

Source: vim.fandom.com

17.5 Add Vim to Windows Explorer Context Menu

Category: configuration
Tags: windows, file-operations, integration

Adds 'Edit with Vim' option to right-click context menu in Windows File Explorer for quick file editing

Example: Example (Vim)

```
Windows Registry Editor Version 5.00

[HKEY_CLASSES_ROOT\*\Shell\Vim]
@="Edit with &Vim"
"Icon"="\"C:\\Program Files\\Vim\\vim80\\gvim.exe\""

[HKEY_CLASSES_ROOT\*\Shell\Vim\command]
@="\"C:\\Program Files\\Vim\\vim80\\gvim.exe\" \"%1\""
```

Example: Example (Neovim)

```
-- Note: This is a Windows Registry modification
-- Equivalent in Neovim would typically be handled by OS-specific scripts
-- Recommend using vim.fn.system() to run registry import if needed
```

Source: vim.fandom.com

17.6 Advanced CTags Configuration for ANT Files

Category: configuration
Tags: ctags, xml, navigation

Create custom regex patterns to extract meaningful tags from ANT XML build files

Example: Example (Vim)

```
--regex-ant=/^[ \t]*<[ \t]*project.*name="([^<"&]+)".*>/\1/p,project/i
--regex-ant=/^[ \t]*<[ \t]*target.*name="([^<"&]+)".*>/\1/t,target/i
--regex-ant=/^[ \t]*<[ \t]*property.*name="([^<"&]+)".*>/\1/r,property/i
```

Example: Example (Neovim)

```
-- CTags configuration can be placed in a separate ctags configuration file
-- This is typically a system-wide or user-specific configuration
```

Source: vim.fandom.com

17.7 Advanced Syntax Sync for Large Files

Category: configuration
Tags: syntax-highlighting, performance, autocmd

Configure syntax synchronization to improve highlighting accuracy for specific file types

Example: Example (Vim)

```
syntax sync minlines=200
autocmd FileType c let c_minlines=500
```

Example: Example (Neovim)

```
vim.cmd('syntax sync minlines=200')
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'c',
  callback = function()
    vim.b.c_minlines = 500
  end
})
```

Source: vim.fandom.com

17.8 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: Example (Vim)

```
" 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
```

Source: Community contributed

17.9 Alternative Bell Suppression Method

Category: configuration
Tags: ui, settings, noise-reduction

Another method to completely disable bells across different Vim/Neovim environments

Example: Example (Vim)

```
set belloff=all
```

Example: Example (Neovim)

```
vim.opt.belloff = 'all'
```

Source: vim.fandom.com

17.10 Alternative CapsLock to Control Mapping

Category: configuration
Tags: key-mapping, productivity

Map CapsLock to Control key for improved keyboard efficiency

Example: Example (Neovim)

```
-- X windows mapping to turn CapsLock into Control
xmodmap -e 'keycode 66 = Control_L'
xmodmap -e 'clear Lock'
xmodmap -e 'add Control = Control_L'
```

Source: vim.fandom.com

17.11 Alternative CapsLock to Escape Mapping

Category: configuration
Tags: key-mapping, system-config

Create a custom .Xmodmap file to permanently remap CapsLock to Escape

Example: Example (Neovim)

```
-- Create ~/.Xmodmap with these contents:
-- remove Lock = Caps_Lock
-- keysym Caps_Lock = Escape
-- add Lock = Caps_Lock
```

Source: vim.fandom.com

17.12 Alternative Escape Key Mapping

Category: configuration
Tags: key-mapping, productivity

Use the Windows/Super key as an additional Escape key for easier mode switching

Example: Example (Vim)

```
xmodmap -e 'keysym Super_L = Escape'
```

Example: Example (Neovim)

```
-- System-level key remapping
-- Can be added to system startup scripts
```

Source: vim.fandom.com

17.13 Always Display Status Line

Category: configuration
Tags: ui, status-line, interface

Configures Vim/Neovim to always show the status line, providing constant visibility of file and mode information

Example: Example (Vim)

```
set laststatus=2
```

Example: Example (Neovim)

```
vim.opt.laststatus = 2
```

Source: vim.fandom.com

17.14 Auto tab completion

Category: Configuration
Tags: completion, tab, autocomplete

Configure TAB to autocomplete words while preserving normal TAB functionality.

Example: Example (Vim)

```
" Vimscript:
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"
```

Example: Example (Neovim)

```
-- Lua:
local function tab_or_complete()
  local col = vim.fn.col('.')
  if col > 1 and vim.fn.getline('.'):sub(col - 2, col):match('^%w') then
    return '<C-N>'
  else
    return '<Tab>'
  end
end

vim.keymap.set('i', '<Tab>', tab_or_complete, { expr = true, desc = 'Tab or
↪  complete' })
vim.opt.dictionary = '/usr/dict/words'
```

Source: Community contributed

17.15 Auto-Reload Vimrc on Save

Category: configuration
Tags: auto-reload, vimrc, configuration

Automatically source your Vimrc file whenever it is saved, eliminating the need to manually reload configuration

Example: Example (Vim)

```
autocmd BufWritePost .vimrc source %
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = '.vimrc',
  callback = function()
    vim.cmd('source %')
  end
})
```

Source: vim.fandom.com

17.16 Auto-continue Doxygen Comments in C/C++

Category: configuration
Tags: comments, documentation, c-cpp

Automatically continue Doxygen-style comments when pressing enter or creating new lines in C/C++ files

Example: Example (Vim)

```
set comments=s0:*\ -,m0:*\ \ ,ex0:*/,s1:/*,mb:*,ex:*/,b0:///,0://
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'c', 'cpp'},
  callback = function()
    vim.opt.comments:append('b0:///,0://')
  end
})
```

Source: vim.fandom.com

17.17 Auto-detect CPU cores for parallel make builds

Category: configuration
Tags: build, performance, shell-integration

Automatically detect the number of CPU cores and configure make to use parallel jobs, improving build performance

Example: Example (Vim)

```
function! SetMakeprg()
  if !empty($NUMBER_OF_PROCESSORS)
    let n = $NUMBER_OF_PROCESSORS + 0
  elseif filereadable('/proc/cpuinfo')
    let n = system('grep -c ^processor /proc/cpuinfo') + 0
  elseif executable('/usr/sbin/psrinfo')
    let n = system('/usr/sbin/psrinfo -p')
  else
    let n = 1
  endif
  let &makeprg = 'make' . (n > 1 ? (' -j'.(n + 1)) : '')
endfunction
call SetMakeprg()
```

Example: Example (Neovim)

```
local function set_makeprg()
  local n = 1
  if vim.env.NUMBER_OF_PROCESSORS and vim.env.NUMBER_OF_PROCESSORS ~= ''
  ↪  then
    n = tonumber(vim.env.NUMBER_OF_PROCESSORS)
  elseif vim.fn.filereadable('/proc/cpuinfo') == 1 then
    n = tonumber(vim.fn.system('grep -c ^processor /proc/cpuinfo'))
  elseif vim.fn.executable('/usr/sbin/psrinfo') == 1 then
    n = tonumber(vim.fn.system('/usr/sbin/psrinfo -p'))
  end

  vim.o.makeprg = n > 1 and 'make -j' .. (n + 1) or 'make'
end

set_makeprg()
```

Source: vim.fandom.com

17.18 Autoload Cscope Database Recursively

Category: configuration
Tags: cscope, project-navigation, auto-configuration

Automatically find and load cscope database by searching up the directory tree, making cross-referencing easier in large projects

Example: Example (Vim)

```
function! LoadCscope()
  let db = findfile("cscope.out", ".;")
  if (!empty(db))
    let path = strpart(db, 0, match(db, "/cscope.out$"))
    set nocscopeverbose
    exe "cs add " . db . " " . path
    set cscopeverbose
  elseif $CSCOPE_DB != ""
    cs add $CSCOPE_DB
  endif
endfunction
au BufEnter /* call LoadCscope()
```

Example: Example (Neovim)

```
local function load_cscope()
  local db = vim.fn.findfile("cscope.out", ".;")
  if db ~= "" then
    local path = db:sub(1, db:find("/cscope.out$") - 1)
    vim.o.cscopeverbose = false
    vim.cmd("cs add " .. db .. " " .. path)
    vim.o.cscopeverbose = true
  elseif os.getenv("CSCOPE_DB") ~= "" then
    vim.cmd("cs add " .. os.getenv("CSCOPE_DB"))
  end
end

vim.api.nvim_create_autocmd("BufEnter", {
  pattern = "/*",
  callback = load_cscope
})
```

Source: vim.fandom.com

17.19 Automated Patch Backup with Patchmode

Category: configuration
Tags: backup, file-management, development

Automatically create backup files when preparing patches by setting patchmode option

Example: Example (Vim)

```
:set patchmode=.orig
```

Example: Example (Neovim)

```
-- Set patchmode to automatically create .orig backup files
vim.opt.patchmode = '.orig'
```

Source: vim.fandom.com

17.20 Automatically Add Modeline to Files

Category: configuration
Tags: key-mapping, modeline, file-settings

Quickly append a modeline to the current file based on your current Vim settings, making it easy to standardize file-specific configurations.

Example: Example (Vim)

```
function! AppendModeline()
  let l:modeline = printf(" vim: set ts=%d sw=%d tw=%d %set :",
        \ &tabstop, &shiftwidth, &textwidth, &expandtab ? '' : 'no')
  let l:modeline = substitute(&commentstring, "%s", l:modeline, "")
  call append(line("$"), l:modeline)
endfunction
nnoremap <silent> <Leader>ml :call AppendModeline()<CR>
```

Example: Example (Neovim)

```
function _G.append_modeline()
  local modeline = string.format(
    " vim: set ts=%d sw=%d tw=%d %set :",
    vim.o.tabstop, vim.o.shiftwidth, vim.o.textwidth,
    vim.o.expandtab and '' or 'no'
  )
  -- Assuming commentstring is set correctly
  modeline = string.format(vim.o.commentstring, modeline)
  vim.api.nvim_buf_set_lines(0, -1, -1, false, {modeline})
end

vim.keymap.set('n', '<Leader>ml', _G.append_modeline, { desc = 'Append
↪  modeline' })
```

Source: vim.fandom.com

17.21 Automatically Add Python Paths to Vim Path

Category: configuration
Tags: python, path, file-navigation

Dynamically add Python system paths to Vim's path, enabling easy navigation to imported modules using gf or Ctrl-W Ctrl-F

Example: Example (Vim)

```
python << EOF
import os
import sys
import vim
for p in sys.path:
    if os.path.isdir(p):
        vim.command(r"set path+=%s" % (p.replace(" ", r"\ ")))
EOF
```

Example: Example (Neovim)

```
local function add_python_paths()
  local sys_path = vim.fn.py3eval('sys.path')
  for _, p in ipairs(sys_path) do
    if vim.fn.isdirectory(p) == 1 then
      vim.opt.path:append(p:gsub(' ', '\ '))
    end
  end
end

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = add_python_paths
})
```

Source: vim.fandom.com

17.22 Automatically Continue Doxygen Comments

Category: configuration
Tags: comments, documentation, c++

Configure Vim to automatically continue Doxygen-style comments when pressing Enter or creating a new line

Example: Example (Vim)

```
set comments=s0:*\ -,m0:*\ \ ,ex0:*/,s1:/*,mb:*,ex:*/,b0:///,0://
```

Example: Example (Neovim)

```
vim.opt.comments = 's0:* -,m0:* ,ex0:*/,s1:/*,mb:*,ex:*/,b0:///,0://'
```

Source: vim.fandom.com

17.23 Automatically Reload Changed Files

Category: configuration
Tags: file-management, auto-reload, workspace

Configure Vim to automatically detect and reload files changed outside the editor

Example: Example (Vim)

```
set autoread

" In visualstudioinvoke.vim
set autoread
```

Example: Example (Neovim)

```
vim.opt.autoread = true

-- In init.lua or a specific project config
vim.api.nvim_create_autocmd({'FocusGained', 'BufEnter'}, {
  pattern = '*',
  command = 'checktime'
})
```

Source: vim.fandom.com

17.24 Automatically Save Global Variables to viminfo

Category: configuration
Tags: variables, persistence, backup

Automatically save global uppercase variables across Vim sessions, including lists and dictionaries, by converting them to strings and restoring them on startup

Example: Example (Vim)

```
fun! WriteVars()
    sil exe "norm! :let g:\<c-a>'"
    let i=0
    for name in split(v)
        if name[2:]==#toupper(name[2:]) && eval("type(".name.")")>1
            let g:VARSAV_{i}=substitute("let".name."=".eval("string(".name.
            ↪  ")"),"\n",'\'."\
".'\''","g")
            let i+=1
        en
    endfor
    let g:VARSAVES=i
endfun
```

```
au VimLeavePre * call WriteVars()

" To restore: :call RestoreVars()
```

Example: Example (Neovim)

```
local function write_vars()
    local vars = vim.fn.split(vim.fn.execute('let g:'))
    for _, name in ipairs(vars) do
        if name:sub(3):upper() == name:sub(3) and
           type(vim.g[name]) == 'table' then
            -- Convert complex variables to storable strings
            vim.g['VARSAV_' .. name] = string.format('vim.g.%s = %s', name,
            ↪  vim.inspect(vim.g[name]))
        end
    end
    vim.g.VARSAVES = #vars
end

vim.api.nvim_create_autocmd('VimLeavePre', {
    callback = write_vars
})
```

Source: vim.fandom.com

17.25 Automatically Save Session and Viminfo on Exit

Category: configuration
Tags: session-management, state-preservation, workspace

Automatically save Vim session and viminfo when exiting, allowing you to restore workspace state when reopening

Example: Example (Vim)

```
au VimLeave * exe 'if strlen(v:this_session) | exe "wviminfo! " .
↪  v:this_session . ".viminfo" | else | exe "wviminfo! " .
↪  "~/.vim/session/" . g:myfilename . ".session.viminfo" | endif'
au VimLeave * exe 'if strlen(v:this_session) | exe "mksession! " .
↪  v:this_session | else | exe "mksession! " . "~/.vim/session/" .
↪  g:myfilename . ".session" | endif'
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('VimLeave', {
  callback = function()
    local session = vim.v.this_session
    if session ~= '' then
      vim.cmd('wviminfo! ' .. session .. '.viminfo')
      vim.cmd('mksession! ' .. session)
```

```
    else
      local filename = vim.g.myfilename or 'default'
      vim.cmd('wviminfo! ~/.vim/session/' .. filename .. '.session.viminfo')
      vim.cmd('mksession! ~/.vim/session/' .. filename .. '.session')
    end
  end
})
```

Source: vim.fandom.com

17.26 Automatically Source Session on Startup

Category: configuration
Tags: auto-commands, startup, workflow

Automatically load a session file when Vim/Neovim starts, streamlining project resumption

Example: Example (Vim)

```
au VimEnter Session.vim :source %:p
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('VimEnter', {
  pattern = 'Session.vim',
  callback = function()
    vim.cmd('source ' .. vim.fn.expand('%:p'))
  end
})
```

Source: vim.fandom.com

17.27 Bright Background Colors in Linux Console

Category: configuration
Tags: terminal, color-configuration, console

Configure Vim to use 16 background colors in Linux framebuffer console, enabling bright background and foreground colors

Example: Example (Vim)

```
if &term =~ "linux"
  if has("terminfo")
    set t_Co=16
    set t_AB=<Esc>[%?%p1%{7}%>%t5%p1%{8}%-%e25%p1%;m<Esc>[4%dm
    set t_AF=<Esc>[%?%p1%{7}%>%t1%p1%{8}%-%e22%p1%;m<Esc>[3%dm
```

```
  endif
endif
```

Example: Example (Neovim)

```
if vim.o.term:match("linux") then
  if vim.fn.has("terminfo") == 1 then
    vim.o.t_Co = 16
    vim.o.t_AB = "\x1b[%?%p1%{7}%>%t5%p1%{8}%-%e25%p1%;m\x1b[4%dm"
    vim.o.t_AF = "\x1b[%?%p1%{7}%>%t1%p1%{8}%-%e22%p1%;m\x1b[3%dm"
  end
end
```

Source: vim.fandom.com

17.28 Build Vim on Ubuntu with GUI Support

Category: configuration
Tags: build, linux, ubuntu

Install development packages and compile Vim with full GUI features

Example: Example (Vim)

```
./configure --with-features=huge --enable-gui=gnome2
make
sudo make install
```

Example: Example (Neovim)

```
-- Lua equivalent focuses on build process configuration
-- Same shell commands apply
-- Requires installing dev packages first
```

Source: vim.fandom.com

17.29 Build Vim with Custom Compilation Info

Category: configuration
Tags: build, compilation, customization

Use –with-compiledby to set custom compilation information displayed in :version

Example: Example (Vim)

```
./configure --with-compiledby='Your Name <email@example.com>'
```

Example: Example (Neovim)

```
-- Lua equivalent is configuration during build process
-- Pass --with-compiledby flag during compilation
```

Source: vim.fandom.com

17.30 Building Vim with Extensive Features

Category: configuration
Tags: build, compilation, development

Configure and build Vim with comprehensive feature support, including GUI and multiple language integrations

Example: Example (Vim)

```
./configure --with-features=huge --enable-gui=gnome2
```

Example: Example (Neovim)

```
-- Lua doesn't directly handle compilation, but demonstrates configuration
↪  philosophy
vim.g.build_features = 'huge'
vim.g.gui_support = 'gnome2'
```

Source: vim.fandom.com

17.31 Building Vim with Python Support on Windows

Category: configuration
Tags: build, windows, python, compilation

Configure and compile Vim with optional Python interface support using Visual Studio

Example: Example (Vim)

```
:: Configuration for building Vim with Python support
set DYNAMIC_PYTHON=yes
set PYTHON=C:/Python27
set PYTHON_VER=27
```

Example: Example (Neovim)

```
-- Lua equivalent for Neovim build configuration
-- Note: Build configuration is typically done via command line or build
↪  scripts
```

```
-- In Neovim, Python support is often handled differently via :checkhealth
```

Source: vim.fandom.com

17.32 Cache and Reset Vim Options Dynamically

Category: configuration
Tags: options, settings, dynamic-configuration

Provides a flexible way to set, track, and reset Vim options, preventing unintended changes by plugins

Example: Example (Vim)

```
let s:option_preferences = []
function! ResetOption(options)
  if empty(a:options)
    let options = s:option_preferences
  else
    let options = a:options
  endif
  for name in options
    let name0 = 'g:'. name .'_default'
    if exists(name0)
      exec 'let &'. name .' = '. name0
    endif
  endfor
endfunction

command! -nargs=* ResetOption :call ResetOption([<f-args>])
command! -nargs=+ SetOption let s:tmlargs=[<f-args>]
 \ | for arg in s:tmlargs[1:-1]
 \ |   if arg =~ '^[+-]\?='
 \ |     exec 'set '.s:tmlargs[0] . arg
 \ |   else
 \ |     exec 'let &'.s:tmlargs[0] .'='. arg
 \ |   endif
 \ | endfor
 \ | call add(s:option_preferences, s:tmlargs[0])
 \ | exec 'let g:'. s:tmlargs[0] .'_default = &'. s:tmlargs[0]
```

Example: Example (Neovim)

```
local M = {}
M.option_preferences = {}

function M.reset_option(options)
  options = options or vim.tbl_keys(M.option_preferences)
  for _, name in ipairs(options) do
    if M.option_preferences[name] then
      vim.opt[name] = M.option_preferences[name]
```

```
    end
  end
end

function M.set_option(name, value)
  local current_value = vim.opt[name]:get()
  M.option_preferences[name] = current_value
  vim.opt[name] = value
end

-- Add commands for easier use
vim.api.nvim_create_user_command('SetOption', function(opts)
  local name = opts.fargs[1]
  local value = opts.fargs[2]
  M.set_option(name, value)
end, { nargs = '+' })

vim.api.nvim_create_user_command('ResetOption', function(opts)
  M.reset_option(opts.fargs)
end, { nargs = '*' })

return M
```

Source: vim.fandom.com

17.33 Cache and Restore Vim Option Preferences

Category: configuration
Tags: options, settings, reset

Provides a flexible way to save, modify, and reset Vim options, preventing unintended changes by plugins or temporary modifications

Example: Example (Vim)

```
let s:option_preferences = []
function! ResetOption(options)
  if empty(a:options)
    let options = s:option_preferences
  else
    let options = a:options
  endif
  for name in options
    let name0 = 'g:'. name .'_default'
    if exists(name0)
      exec 'let &'. name .' = '. name0
    endif
  endfor
endfunction

command! -nargs=* ResetOption :call ResetOption([<f-args>])
command! -nargs=+ SetOption let s:tmlargs=[<f-args>]
```

```
  \ | for arg in s:tmlargs[1:-1]
  \ |   if arg =~ '^[+-]\?='
  \ |     exec 'set '.s:tmlargs[0] . arg
  \ |   else
  \ |     exec 'let &'.s:tmlargs[0] .'='. arg
  \ |   endif
  \ | endfor
  \ | call add(s:option_preferences, s:tmlargs[0])
  \ | exec 'let g:'. s:tmlargs[0] .'_default = &'. s:tmlargs[0]
```

Example: Example (Neovim)

```
local option_preferences = {}

function _G.reset_option(options)
  options = options or vim.tbl_keys(option_preferences)
  for _, name in ipairs(options) do
    if option_preferences[name] then
      vim.opt[name] = option_preferences[name]
    end
  end
end

vim.api.nvim_create_user_command('ResetOption', function(opts)
  reset_option(opts.fargs)
end, { nargs = '*' })

vim.api.nvim_create_user_command('SetOption', function(opts)
  local option_name = opts.fargs[1]
  local value = opts.fargs[2]

  if value then
    option_preferences[option_name] = vim.opt[option_name]:get()
    vim.opt[option_name] = value
  end
end, { nargs = '+' })
```

Source: vim.fandom.com

17.34 Chain Commands in Vimrc Mappings

Category: configuration
Tags: key-mapping, custom-mapping, configuration

Use <bar> to chain commands in Vim configuration and key mappings

Example: Example (Vim)

```
map <F6> <ESC>:echo "test" <bar> :echo "test2"
```

Example: Example (Neovim)

```
-- In Lua, use vim.keymap.set with multiple commands
vim.keymap.set('n', '<F6>', function()
  vim.cmd('echo "test"')
  vim.cmd('echo "test2"')
end)
```

Source: vim.fandom.com

17.35 Check Vim Installation Details

Category: configuration
Tags: system-config, troubleshooting, vim-info

Examine Vim configuration and system-wide customizations before installation

Example: Example (Vim)

```
:version
:scriptnames
```

Example: Example (Neovim)

```
-- Print Vim version information
print(vim.version())

-- List loaded scripts
for _, script in ipairs(vim.fn.getscriptinfo()) do
  print(script.name)
end
```

Source: vim.fandom.com

17.36 Check Vim Installation Features

Category: configuration
Tags: system-info, troubleshooting

How to verify Vim installation details and system-wide customizations

Example: Example (Vim)

```
:version
:scriptnames
```

Example: Example (Neovim)

```
-- Check Vim version and loaded scripts
vim.cmd('version')
vim.cmd('scriptnames')
```

Source: vim.fandom.com

17.37 Check and Set Buffer Filetype

Category: configuration
Tags: filetype, buffer-management

Quick methods to check and manually set the filetype of the current buffer

Example: Example (Vim)

```
" Check current filetype
:set ft?

" Set filetype manually
:set ft=python
```

Example: Example (Neovim)

```
-- Check current filetype
print(vim.o.filetype)

-- Set filetype manually
vim.o.filetype = 'python'
```

Source: vim.fandom.com

17.38 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: Example (Vim)

```
: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
```

Source: Community contributed

17.39 Choose the Right Vim Build for Your System

Category: configuration
Tags: installation, platform-specific, versions

Select the appropriate Vim build based on your Windows architecture and feature requirements

Example: Example (Vim)

```
" Check system architecture
" 32-bit vs 64-bit considerations
```

Example: Example (Neovim)

```
-- Consider 64-bit Vim for:
-- 1. Editing files >4GB
-- 2. Better performance
-- 3. Compatibility with 64-bit systems
```

Source: vim.fandom.com

17.40 Choosing the Right Vim Version

Category: configuration
Tags: installation, version-management

Select Vim version based on stability and feature requirements - nightly builds vs stable releases

Example: Example (Vim)

```
" Consider:
" - Nightly builds for newest features
" - Stable builds for production
" - Check feature support with :version
```

Example: Example (Neovim)

```
-- Recommend checking vim-win32-installer for Windows
-- Use package managers or official sources for other platforms
-- Verify feature support with vim.fn.has()
```

Source: vim.fandom.com

17.41 Clone Vim Source Using Mercurial

Category: configuration
Tags: version-control, source-code, vim-development

Demonstrates how to clone the official Vim source code repository using Mercurial, which allows developers to track and update Vim's source code directly

Example: Example (Vim)

```
# Clone Vim repository
hg clone http://hg.256bit.org/vim vim
```

Example: Example (Neovim)

```
-- Lua equivalent for shell command
-- Use os.execute or vim.fn.system to run shell commands
vim.fn.system('hg clone http://hg.256bit.org/vim vim')
```

Source: vim.fandom.com

17.42 Commenting Configuration Files

Category: configuration
Tags: documentation, comments

Use double quotes to add comments in Vim configuration files, helping document and explain configuration settings

Example: Example (Vim)

```
" This is a comment in vimrc
map zm :let @/=expand("<cword>") <BAR> split <BAR> execute 'normal n'<CR>
```

Example: Example (Neovim)

```
-- This is a comment in init.lua
-- Use comments to explain complex mappings or configurations
```

Source: vim.fandom.com

17.43 Compile Vim with Python Support Separately

Category: configuration
Tags: compilation, python, performance

Create two Vim executables - one with standard options and another with Python sup-

port for more efficient startup and specialized editing

Example: Example (Vim)

```
./configure --enable-pythoninterp
```

Example: Example (Neovim)

```
-- Use compile-time configuration for separate Vim builds
-- Compile with: ./configure --enable-pythoninterp
-- Set different configurations for different use cases
```

Source: vim.fandom.com

17.44 Completely Disable Syntax Highlighting and Search Highlights

Category: configuration
Tags: syntax, display, search

Disable all syntax highlighting and search result highlights for a minimal editor view

Example: Example (Vim)

```
syntax off
set nohlsearch
set t_Co=0
```

Example: Example (Neovim)

```
vim.cmd('syntax off')
vim.opt.hlsearch = false
vim.o.t_Co = 0
```

Source: vim.fandom.com

17.45 Completely Silence Vim Bells

Category: configuration
Tags: ui, settings, noise-reduction

Comprehensive method to disable all bell/flash notifications in Vim

Example: Example (Vim)

```
set belloff=all
```

Example: Example (Neovim)

```
vim.opt.belloff = 'all'
```

Source: vim.fandom.com

17.46 Comprehensive List of Special Keys

Category: configuration
Tags: key-mapping, special-keys, vim-reference

Reference for special key names in Vim, useful for creating cross-platform key mappings

Example: Example (Vim)

```
" Navigation Keys:
" <Insert>, <Del>, <Home>, <End>, <PageUp>, <PageDown>

" Numeric Keypad Keys:
" <kDivide>, <kMultiply>, <kHome>, <kEnd>, etc.
```

Example: Example (Neovim)

```
-- Special key names can be used directly in Neovim mappings
-- Example mapping:
vim.keymap.set('n', '<Home>', ':echo "Home key pressed"', {})
```

Source: vim.fandom.com

17.47 Conditionally Source Runtime Scripts

Category: configuration
Tags: script-loading, runtime, error-handling

Safely source runtime scripts only if they exist, preventing errors in different environments

Example: Example (Vim)

```
if filereadable($VIMRUNTIME . "/macros/matchit.vim")
 source $VIMRUNTIME/macros/matchit.vim
endif
```

Example: Example (Neovim)

```
local matchit_path = vim.fn.expand('$VIMRUNTIME/macros/matchit.vim')
if vim.fn.filereadable(matchit_path) == 1 then
  vim.cmd.source(matchit_path)
```

```
end
```

Source: vim.fandom.com

17.48 Configure 256-Color Terminal Support in Vim

Category: configuration
Tags: terminal, color-scheme, 256-colors

Verify and configure 256-color support in your terminal and Vim configuration to use advanced color schemes

Example: Example (Vim)

```
" Check terminal color support
:echo $TERM
:set t_Co?

" Manually set 256 colors if needed
:set t_Co=256
:color jellybeans
```

Example: Example (Neovim)

```
-- Check terminal color support
vim.pretty_print(vim.env.TERM)
print(vim.o.t_Co)

-- Manually set 256 colors if needed
vim.o.t_Co = 256
vim.cmd.colorscheme('jellybeans')
```

Source: vim.fandom.com

17.49 Configure Automatic Indentation Methods

Category: configuration
Tags: indentation, auto-indent, code-formatting

Use different automatic indentation methods like autoindent, smartindent, and cindent for various file types

Example: Example (Vim)

```
# Basic auto-indentation
set autoindent

# Smart indentation for C-like files
```

```
set smartindent

# Precise C-style indentation
set cindent
```

Example: Example (Neovim)

```
-- Basic auto-indentation
vim.opt.autoindent = true

-- Smart indentation for C-like files
vim.opt.smartindent = true

-- Precise C-style indentation
vim.opt.cindent = true
```

Source: vim.fandom.com

17.50 Configure Backspace Behavior in Insert Mode

Category: configuration
Tags: insert-mode, editing, key-behavior

Allow backspace to delete previously entered characters, autoindent, and across line breaks

Example: Example (Vim)

```
set backspace=indent,eol,start
# Or
set backspace=2
```

Example: Example (Neovim)

```
vim.opt.backspace = {'indent', 'eol', 'start'}
```

Source: vim.fandom.com

17.51 Configure COBOL Syntax Highlighting in Neovim

Category: configuration
Tags: syntax, filetype, legacy-code

Resolve COBOL syntax highlighting issues by setting a legacy code flag to prevent overly aggressive syntax matching

Example: Example (Vim)

```
let cobol_legacy_code=1
```

Example: Example (Neovim)

```
vim.g.cobol_legacy_code = 1
```

Source: vim.fandom.com

17.52 Configure Clipboard Behavior

Category: configuration
Tags: clipboard, registers, settings

Configure Vim/Neovim to automatically use system clipboard

Example: Example (Vim)

```
set cb+=unnamed
set go+=a
```

Example: Example (Neovim)

```
-- Enable system clipboard by default
vim.opt.clipboard = 'unnamed,unnamedplus'
```

Source: vim.fandom.com

17.53 Configure Diff Options Flexibly

Category: configuration
Tags: diff, comparison, file-operations

Dynamically configure diff options to handle case-insensitive or whitespace-insensitive comparisons

Example: Example (Vim)

```
function! MyDiff()
  let opt = ''
  if &diffopt =~ 'icase' | let opt = opt . '-i ' | endif
  if &diffopt =~ 'iwhite' | let opt = opt . '-b ' | endif
endfunction
```

Example: Example (Neovim)

```
function _G.MyDiff()
  local opt = ''
  if vim.opt.diffopt:get():find('icase') then
    opt = opt .. '-i '
  end
  if vim.opt.diffopt:get():find('iwhite') then
    opt = opt .. '-b '
  end
  return opt
end
```

Source: vim.fandom.com

17.54 Configure Errorformat for Intel Fortran Compiler

Category: configuration
Tags: compiler, error-handling, fortran

Set up custom errorformat to parse Intel Fortran compiler error messages correctly, enabling better error navigation in Vim/Neovim

Example: Example (Vim)

```
set efm=%E%.%#rror:\ %f\,\ line\ %l:\ %m,\%-C%.%#,\%-Z\%p^
let isf="@,48-57,/,.,-,_,+,#,^,,$,%,~,="
```

Example: Example (Neovim)

```
-- Configure errorformat for Intel Fortran compiler
vim.opt.errorformat:append('%E%.%#rror: %f, line %l: %m,%-C%.%#,%-Z%p^')
vim.opt.isfname = '@,48-57,/,.,-,_,+,#,^,,$,%,~,='
```

Source: vim.fandom.com

17.55 Configure External Diff Program

Category: configuration
Tags: diff, external-tools, system-integration

Properly set up an external diff program for Vim, especially on Windows systems

Example: Example (Vim)

```
set diffexpr=

" Optional: Custom diff function if needed
function MyDiff()
```

```
  let opt = ''
  if &diffopt =~ 'icase' | let opt = opt . '-i ' | endif
  if &diffopt =~ 'iwhite' | let opt = opt . '-b ' | endif
  silent execute '!diff -a ' . opt . v:fname_in . ' ' . v:fname_new . ' > '
  ↪ . v:fname_out
endfunction
```

Example: Example (Neovim)

```
-- Clear diff expression
vim.o.diffexpr = ''

-- Optional: Custom diff function in Lua
local function my_diff()
  local opt = ''
  if vim.opt.diffopt:get():match('icase') then opt = opt .. '-i ' end
  if vim.opt.diffopt:get():match('iwhite') then opt = opt .. '-b ' end
  vim.fn.system(string.format('diff -a %s %s %s > %s', opt, v:fname_in,
  ↪ v:fname_new, v:fname_out))
end
```

Source: vim.fandom.com

17.56 Configure External Diff Tool

Category: configuration
Tags: diff, external-tool, system-integration

Properly set up an external diff tool, particularly for Windows systems, to ensure reliable file comparison

Example: Example (Vim)

```
set diffexpr=

" Optional: Custom diff configuration
function MyDiff()
  let opt = ''
  if &diffopt =~ 'icase' | let opt = opt . '-i ' | endif
  if &diffopt =~ 'iwhite' | let opt = opt . '-b ' | endif
endfunction
```

Example: Example (Neovim)

```
-- Clear default diff expression
vim.o.diffexpr = ''

-- Optional custom diff function
local function MyDiff()
  local opt = ''
```

```
  if vim.opt.diffopt:get():match('icase') then
    opt = opt .. '-i '
  end
  if vim.opt.diffopt:get():match('iwhite') then
    opt = opt .. '-b '
  end
  return opt
end
```

Source: vim.fandom.com

17.57 Configure Fonts in Neovim on macOS

Category: configuration
Tags: ui, font-configuration, macos

Set custom fonts and font size for Neovim with optional antialiasing

Example: Example (Vim)

```
set guifont=Monaco:h14
set antialias
```

Example: Example (Neovim)

```
-- Set font in Neovim (requires Neovim GUI like Neovide)
vim.opt.guifont = "Monaco:h14"

-- Optional: Enable/disable font antialiasing
-- Note: Specific antialiasing might require GUI-specific configuration
```

Source: vim.fandom.com

17.58 Configure GUI Font in Neovim

Category: configuration
Tags: gui, font, setup

Conditionally set GUI font across different platforms with fallback options

Example: Example (Vim)

```
if has('gui_running')
  set guioptions-=T  " no toolbar
  colorscheme elflord
  set lines=60 columns=108 linespace=0
  if has('gui_win32')
    set guifont=DejaVu_Sans_Mono:h10:cANSI
```

```
    else
      set guifont=DejaVu\ Sans\ Mono\ 10
    endif
  endif
```

Example: Example (Neovim)

```
if vim.g.neovide or vim.g.gui_running then
  vim.opt.guioptions:remove('T')  -- no toolbar
  vim.cmd('colorscheme elflord')
  vim.opt.lines = 60
  vim.opt.columns = 108
  vim.opt.linespace = 0

  if vim.fn.has('win32') == 1 then
    vim.opt.guifont = 'DejaVu_Sans_Mono:h10:cANSI'
  else
    vim.opt.guifont = 'DejaVu Sans Mono 10'
  end
end
```

Source: vim.fandom.com

17.59 Configure IPython Editor in Vim

Category: configuration
Tags: python, editor-config, external-tools

Set Vim (specifically gvim) as the default editor for IPython's %edit command

Example: Example (Vim)

```
" In ipython_config.py
c.TerminalInteractiveShell.editor = 'gvim -f'
```

Example: Example (Neovim)

```
-- For modern IPython configuration
-- Typically done in Python config file, not directly in Neovim
-- But you can set external editor in IPython settings
```

Source: vim.fandom.com

17.60 Configure Indentation Styles in Vim/Neovim

Category: configuration
Tags: indentation, formatting, code-style

Configure different indentation strategies for spaces, tabs, or mixed indentation across file types

Example: Example (Vim)

```
" Spaces-only indentation
set expandtab
set shiftwidth=2
set softtabstop=2

" Tabs-only indentation
set shiftwidth=2
set tabstop=2

" Mixed tabs and spaces
set shiftwidth=2
set softtabstop=2
```

Example: Example (Neovim)

```
-- Spaces-only indentation
vim.opt.expandtab = true
vim.opt.shiftwidth = 2
vim.opt.softtabstop = 2

-- Tabs-only indentation
vim.opt.shiftwidth = 2
vim.opt.tabstop = 2

-- Mixed tabs and spaces
vim.opt.shiftwidth = 2
vim.opt.softtabstop = 2
```

Source: vim.fandom.com

17.61 Configure Man Page Viewing

Category: configuration
Tags: man-pages, documentation

Disable SGR for better man page rendering in Vim/Neovim

Example: Example (Vim)

```
let $GROFF_NO_SGR=1
```

Example: Example (Neovim)

```
vim.env.GROFF_NO_SGR = '1'
```

Source: vim.fandom.com

17.62 Configure Mouse Behavior

Category: configuration
Tags: mouse, ui, settings

Adjust mouse interactions in Vim/Neovim, limiting or disabling specific mouse functions

Example: Example (Vim)

```
set mouse=nicr  # Disable visual mode mouse selection
set mouse=      # Completely disable mouse
```

Example: Example (Neovim)

```
vim.opt.mouse = 'nicr'  -- Limit mouse interactions
-- or
vim.opt.mouse = ''  -- Disable mouse completely
```

Source: vim.fandom.com

17.63 Configure Number Format Options

Category: configuration
Tags: number-format, editing-options

Customize how Vim handles number increments/decrements by modifying nrformats option

Example: Example (Vim)

```
" View current number format options
:set nrformats?

" Add alpha to number formats
:set nrformats+=alpha

" Remove octal format
:set nrformats-=octal
```

Example: Example (Neovim)

```
-- View current number format options
vim.pretty_print(vim.o.nrformats)

-- Add alpha to number formats
vim.o.nrformats = vim.o.nrformats .. ',alpha'

-- Remove octal format
```

```
vim.o.nrformats = vim.o.nrformats:gsub('octal,?', '')
```

Source: vim.fandom.com

17.64 Configure Path for File Searching

Category: configuration
Tags: file-operations, path-configuration

Set Vim's path option to easily find and open include files or project files

Example: Example (Vim)

```
" Add multiple directories to search path
set path+=src/**
set path+=/usr/include
```

Example: Example (Neovim)

```
-- Configure path for file searching
vim.opt.path:append('src/**')
vim.opt.path:append('/usr/include')
```

Source: vim.fandom.com

17.65 Configure SSH Tunneling for SVN

Category: configuration
Tags: ssh, version-control, subversion

Set up SSH tunneling for Subversion to work with TortoiseSVN on non-standard ports

Example: Example (Vim)

```
# In Subversion config file
ssh = $SVN_SSH "C:/Program Files/tortoisesvn/bin/tortoiseplink.exe"
```

Example: Example (Neovim)

```
-- This is typically set in the Subversion config file, not directly in
↪  Neovim
-- Lua equivalent would involve configuring SSH settings externally
```

Source: vim.fandom.com

17.66 Configure Syntax Highlighting Options

Category: configuration
Tags: syntax-highlighting, customization, filetype

Many syntax files have built-in configuration options you can enable by setting variables before loading the syntax file

Example: Example (Vim)

```
" Example for Perl syntax configuration
let perl_fold=1
let perl_fold_blocks=1
```

Example: Example (Neovim)

```
-- Lua equivalent for configuring syntax options
vim.g.perl_fold = 1
vim.g.perl_fold_blocks = 1
```

Source: vim.fandom.com

17.67 Configure Terminal Color Highlighting

Category: configuration
Tags: terminal, color-scheme, telnet

Configure Vim to enable color highlighting in terminal environments like SecureCRT, handling color support for non-GUI terminals

Example: Example (Vim)

```
if !has("gui_running")
  set t_Co=8
  set t_Sf=^[[3%p1%dm
  set t_Sb=^[[4%p1%dm
endif
```

Example: Example (Neovim)

```
if not vim.g.gui_running then
  vim.opt.t_Co = 8
  -- Note: Terminal color escape sequences might require additional
  ↪  configuration
  -- Consult your specific terminal emulator's documentation
end
```

Source: vim.fandom.com

17.68 Configure Terminal Colors on Solaris

Category: configuration
Tags: terminal, color-scheme, solaris

Set terminal type and color configuration for Vim on Solaris systems to enable syntax highlighting and color support

Example: Example (Vim)

```
if &term =~ "xterm"
  if has("terminfo")
    set t_Co=8
    set t_Sf=ESC[3%p1%dm
    set t_Sb=ESC[4%p1%dm
  else
    set t_Co=8
    set t_Sf=ESC[3%dm
    set t_Sb=ESC[4%dm
  endif
endif
```

Example: Example (Neovim)

```
-- Configure terminal colors for Solaris
if vim.o.term:match("xterm") then
  if vim.fn.has("terminfo") == 1 then
    vim.o.t_Co = 8
    vim.o.t_Sf = "ESC[3%p1%dm"
    vim.o.t_Sb = "ESC[4%p1%dm"
  else
    vim.o.t_Co = 8
    vim.o.t_Sf = "ESC[3%dm"
    vim.o.t_Sb = "ESC[4%dm"
  end
end
```

Source: vim.fandom.com

17.69 Configure Terminal Cursor for Different Modes

Category: configuration
Tags: terminal-config, cursor-shape, mode-indicators

Set different cursor shapes and colors for various Vim modes in terminal environments, helping to visually distinguish between normal, insert, and replace modes

Example: Example (Vim)

```
if &term =~ '^xterm\|rxvt'
  " solid underscore in insert mode
  let &t_SI .= "\<Esc>[4 q"
  " solid block in normal mode
  let &t_EI .= "\<Esc>[2 q"
endif
```

Example: Example (Neovim)

```
-- Terminal cursor shape configuration
if vim.fn.has('unix') then
  vim.cmd[[
    let &t_SI = "\<Esc>[4 q"
    let &t_EI = "\<Esc>[2 q"
  ]]
  -- Alternative Lua approach (requires more complex escape sequence
  ↪  handling)
  vim.api.nvim_create_autocmd('ModeChanged', {
    pattern = '*:i',
    callback = function()
      vim.cmd('set guicursor=i:ver25')
    end
  })
  vim.api.nvim_create_autocmd('ModeChanged', {
    pattern = 'i:*',
    callback = function()
      vim.cmd('set guicursor=n:block')
    end
  })
end
```

Source: vim.fandom.com

17.70 Configure Unicode and UTF-8 Encoding

Category: configuration
Tags: encoding, unicode, internationalization

Set up Vim/Neovim to handle Unicode text correctly, ensuring proper internal representation and file encoding

Example: Example (Vim)

```
if has("multi_byte")
  if &encoding !~? '^u'
    if &termencoding == ""
      let &termencoding = &encoding
    endif
    set encoding=utf-8
  endif
```

```
  setglobal fileencoding=utf-8
  set fileencodings=ucs-bom,utf-8,latin1
endif
```

Example: Example (Neovim)

```
if vim.fn.has('multi_byte') == 1 then
  if not string.lower(vim.o.encoding):match('^u') then
    if vim.o.termencoding == '' then
      vim.o.termencoding = vim.o.encoding
    end
    vim.o.encoding = 'utf-8'
  end
  vim.o.fileencoding = 'utf-8'
  vim.o.fileencodings = 'ucs-bom,utf-8,latin1'
end
```

Source: vim.fandom.com

17.71 Configure Vim Color Support on HP-UX

Category: configuration
Tags: terminal, color-support, compilation

Use curses library instead of termlib to enable proper color support when building Vim on HP-UX systems

Example: Example (Vim)

```
./configure --with-tlib=curses
```

Example: Example (Neovim)

```
-- For Neovim compilation, use similar configure flag
-- Relevant when building from source on specific Unix systems
-- Note: Most modern Neovim installations won't need this
```

Source: vim.fandom.com

17.72 Configure Vim Language and Encoding Support

Category: configuration
Tags: internationalization, language-support, encoding

Set up language and encoding configurations for multilingual Vim usage, specifically for Greek language support

Example: Example (Vim)

```
" Set language
let $LANG='el'
:lan mes el

" Set menu language
:menut English Greek
```

Example: Example (Neovim)

```
-- Set language
vim.env.LANG = 'el'
vim.cmd('lan mes el')

-- Set menu language
vim.cmd('menut English Greek')
```

Source: vim.fandom.com

17.73 Configuring External Diff Program

Category: configuration
Tags: diff, external-tools, system-integration

Clear diffexpr to use default system diff program, resolving Windows-specific diff issues

Example: Example (Vim)

```
set diffexpr=
```

Example: Example (Neovim)

```
vim.o.diffexpr = ''
```

Source: vim.fandom.com

17.74 Convert Note Files to Vim Help Documents

Category: configuration
Tags: help, documentation, custom-docs

Create custom help documents from personal note files, making them searchable and accessible within Vim using help commands

Example: Example (Vim)

```
" Vim modeline for help files
" vim: filetype=help foldmethod=marker foldmarker=<<<,>>> modifiable
↪  noreadonly

" Generate help tags
:helptags ~/.vim/doc
```

Example: Example (Neovim)

```
-- In Lua, you can use vim.cmd to generate help tags
vim.cmd('helptags ~/.vim/doc')

-- Autocmd to regenerate help tags after writing
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = '~/.vim/doc/*',
  callback = function()
    vim.cmd('helptags ~/.vim/doc')
  end
})
```

Source: vim.fandom.com

17.75 Create Consistent Color Scheme Between Vim and GVim

Category: configuration
Tags: color-scheme, ui, customization

Define a custom color scheme that provides consistent appearance across terminal and GUI Vim, ensuring visual uniformity

Example: Example (Vim)

```
" Vim color file for consistent colors
set background=dark
hi clear
if exists("syntax_on")
  syntax reset
endif

" Custom color definitions
hi Normal guifg=White guibg=Black
hi Comment guifg=Blue guibg=Black
hi Constant guifg=#BB0000 guibg=Black
```

Example: Example (Neovim)

```
-- Lua equivalent for consistent color configuration
vim.opt.background = 'dark'
```

```
-- Create custom highlight groups
vim.api.nvim_set_hl(0, 'Normal', { fg = 'White', bg = 'Black' })
vim.api.nvim_set_hl(0, 'Comment', { fg = 'Blue', bg = 'Black' })
vim.api.nvim_set_hl(0, 'Constant', { fg = '#BB0000', bg = 'Black' })
```

Source: vim.fandom.com

17.76 Create Custom Color Scheme Extending Existing Scheme

Category: configuration
Tags: color-scheme, customization, ui

Easily modify an existing Vim color scheme by overriding specific highlight groups without copying the entire scheme

Example: Example (Vim)

```
"Clear existing highlights
hi clear
if exists("syntax_on")
  syntax reset
endif

"Load base colorscheme
runtime colors/blue.vim

"Override scheme name
let g:colors_name = "my-blue"

"Customize specific highlight groups
hi clear StatusLine
hi clear StatusLineNC
hi StatusLine guifg=black guibg=white
hi StatusLineNC guifg=LightCyan guibg=blue gui=bold
```

Example: Example (Neovim)

```
-- Clear existing highlights
vim.cmd('hi clear')
if vim.g.syntax_on then
  vim.cmd('syntax reset')
end

-- Load base colorscheme
vim.cmd('runtime colors/blue.vim')

-- Override scheme name
vim.g.colors_name = 'my-blue'

-- Customize specific highlight groups
vim.cmd('hi clear StatusLine')
```

```
vim.cmd('hi clear StatusLineNC')
vim.cmd('hi StatusLine guifg=black guibg=white')
vim.cmd('hi StatusLineNC guifg=LightCyan guibg=blue gui=bold')
```

Source: vim.fandom.com

17.77 Create Custom Syntax Highlighting for Files

Category: configuration
Tags: syntax-highlighting, file-types, custom-config

Create a custom syntax file to define syntax highlighting for unique file types or domain-specific formats

Example: Example (Vim)

```
" Vim syntax file template
if exists("b:current_syntax")
  finish
endif

" Define syntax elements
syn keyword customKeywords keyword1 keyword2
syn match customMatch 'regexp'
syn region customRegion start='x' end='y'

" Link syntax groups to highlight groups
hi def link customKeywords Keyword
hi def link customMatch Number
hi def link customRegion String

let b:current_syntax = 'custom'
```

Example: Example (Neovim)

```
-- Neovim syntax file template in Lua
if vim.b.current_syntax then
  return
end

-- Define syntax elements using Treesitter or native syntax
vim.treesitter.highlighter.new({
  highlights = {
    ['keyword'] = '@keyword',
    ['match'] = '@number',
    ['region'] = '@string'
  }
})

vim.b.current_syntax = 'custom'
```

Source: vim.fandom.com

17.78 Create Desktop-Specific Vim Instances

Category: configuration
Tags: multi-desktop, server-mode, vim-instance

Create separate Vim server instances for each desktop workspace

Example: Example (Vim)

```
vim --servername desktop_0 --remote-tab-silent file.txt
```

Example: Example (Neovim)

```
-- Lua function to create desktop-specific Vim server
local function create_desktop_server()
  local desktop = vim.fn.system('xprop -root -notype _NET_CURRENT_DESKTOP |
  ↪  perl -pe "s/.*?= (\d+)/$1/"'):gsub('\n', '')
  vim.fn.serverstart('desktop_' .. desktop)
end
```

Source: vim.fandom.com

17.79 Create TODO Tag Lists with Ctags

Category: configuration
Tags: ctags, task-management, code-navigation

Generate and list TODO tags automatically across code files using ctags

Example: Example (Vim)

```
# Add to ~/.ctags
--regex-java=/\/\/TODO(.*)/todo\1/

# In vimrc
command TODO tselect /^todo djh
```

Example: Example (Neovim)

```
-- Lua equivalent requires external ctags configuration
-- Create a Lua function to list TODOs
vim.api.nvim_create_user_command('TODO', function()
  vim.cmd('tselect /^todo')
end, {})
```

Source: vim.fandom.com

17.80 Create Vim Translation Files

Category: configuration
Tags: internationalization, localization, translation

Steps to extract, translate, and install message translation files for Vim

Example: Example (Vim)

```
" No direct Vimscript implementation
" Relies on external tools like xgettext and msgfmt
```

Example: Example (Neovim)

```
-- No direct Lua implementation
-- Translation process involves external GNU gettext utilities
```

Source: vim.fandom.com

17.81 Create and Install Language Translation Files

Category: configuration
Tags: internationalization, translation, locale

Process for generating .po and .mo files to support custom language translations in Vim

Example: Example (Vim)

```
" Key steps:
" 1. Extract translatable strings
" 2. Translate vim.po file
" 3. Convert to vim.mo
" 4. Install in language directory
```

Example: Example (Neovim)

```
-- Translation process requires external tools like xgettext and msgfmt
-- Typically handled outside of Lua/Neovim configuration
```

Source: vim.fandom.com

17.82 Cross-Platform GUI Font Configuration

Category: configuration
Tags: font, gui, cross-platform

Provides a robust, cross-platform method for setting GUI fonts in Vim, with platform-

specific configurations

Example: Example (Vim)

```
if has("gui_running")
  if has("gui_gtk2") || has("gui_gtk3")
    set guifont=Courier\ New\ 11
  elseif has("gui_photon")
    set guifont=Courier\ New:s11
  elseif has("gui_kde")
    set guifont=Courier\ New/11/-1/5/50/0/0/0/1/0
  elseif has("x11")
    set guifont=-*-courier-medium-r-normal-*-*-180-*-*-m-*-*
  else
    set guifont=Courier_New:h11:cDEFAULT
  endif
endif
```

Example: Example (Neovim)

```
if vim.g.gui_running then
  if vim.fn.has('gui_gtk2') == 1 or vim.fn.has('gui_gtk3') == 1 then
    vim.opt.guifont = 'Courier\ New 11'
  elseif vim.fn.has('gui_photon') == 1 then
    vim.opt.guifont = 'Courier\ New:s11'
  elseif vim.fn.has('gui_kde') == 1 then
    vim.opt.guifont = 'Courier\ New/11/-1/5/50/0/0/0/1/0'
  elseif vim.fn.has('x11') == 1 then
    vim.opt.guifont = '-*-courier-medium-r-normal-*-*-180-*-*-m-*-*'
  else
    vim.opt.guifont = 'Courier_New:h11:cDEFAULT'
  end
end
```

Source: vim.fandom.com

17.83 Cross-Terminal Cursor Shape Detection

Category: configuration
Tags: terminal, cursor, cross-platform

Configure cursor shape and color for different terminals like xterm, rxvt, and tmux with mode-specific escape sequences

Example: Example (Vim)

```
if &term =~ "xterm\|rxvt"
  let &t_SI = "\<Esc>]12;orange\x7"
  let &t_EI = "\<Esc>]12;red\x7"
endif
```

Example: Example (Neovim)

```
if vim.fn.match(vim.o.term, "xterm\|rxvt") ~= -1 then
  vim.o.t_SI = "\<Esc>]12;orange\x7"
  vim.o.t_EI = "\<Esc>]12;red\x7"
end
```

Source: vim.fandom.com

17.84 Custom Filetype Detection for Unique File Extensions

Category: configuration
Tags: filetype, autocmd, file-detection

Create custom filetype detection rules for specific file extensions or file content patterns

Example: Example (Vim)

```
augroup filetypedetect
  au! BufNewFile,BufRead *.csv setf csv
  au! BufNewFile,BufRead *.cmd if getline(1) =~ '^/\*' | setf rexx | else |
   ↪  setf dosbatch | endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('CustomFiletypeDetect', { clear = true })

vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.csv',
  callback = function()
    vim.bo.filetype = 'csv'
  end,
  group = 'CustomFiletypeDetect'
})

vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.cmd',
  callback = function()
    local first_line = vim.fn.getline(1)
    if first_line:match('^/\*') then
      vim.bo.filetype = 'rexx'
    else
      vim.bo.filetype = 'dosbatch'
    end
  end,
  group = 'CustomFiletypeDetect'
})
```

Source: vim.fandom.com

17.85 Custom Make Command for Specific Compiler

Category: configuration
Tags: compiler, build-system, error-handling

Customize make command and error format for NEC V850 CA850 compiler, allowing seamless error navigation

Example: Example (Vim)

```
" Use build.exe for making
set makeprg=build\ -script\ script.bld\ -XO\ build.err

" Find CA850 errors and warnings
set errorformat=%f\ %l\ %.%#rror:\ %t%n:%m,%f\ %l\ %.%#arning:\
↪  %t%n:%m,%+Eld850:\ %.%#rror:\ %t%n:%m
```

Example: Example (Neovim)

```
-- Set custom make program
vim.opt.makeprg = 'build -script script.bld -XO build.err'

-- Configure error format for specific compiler
vim.opt.errorformat:append({
  '%f %l %.%#rror: %t%n:%m',
  '%f %l %.%#arning: %t%n:%m',
  '%+Eld850: %.%#rror: %t%n:%m'
})
```

Source: vim.fandom.com

17.86 Custom Syntax Highlighting for File Types

Category: configuration
Tags: syntax, file-type, customization

Easily define custom syntax highlighting for specific file types or override existing highlighting

Example: Example (Vim)

```
" To highlight *.foo files like HTML
" Add to your vimrc or create a custom syntax file
```

Example: Example (Neovim)

```
-- Reference :help new-filetype
-- Use vim.filetype.add() to define custom file type associations
```

Source: vim.fandom.com

17.87 Custom Syntax Highlighting for Mixed Languages

Category: configuration
Tags: syntax-highlighting, custom-syntax, multi-language

Create a custom syntax file that supports multiple language contexts within a single file, useful for preprocessor or template languages

Example: Example (Vim)

```
" Vim syntax file for mixed language support
" Load multiple syntax contexts
syn include @ep3Perl    syntax/perl.vim
syn include @ep3Verilog syntax/verilog.vim

" Define regions with different syntax
syn region ep3PerlChunk matchgroup=Delimiter start="@perl_begin"
 ↪  end="@perl_end" contains=@ep3Perl,ep3Vline
```

Example: Example (Neovim)

```
-- Lua equivalent for custom syntax highlighting
-- Note: More complex syntax definition typically requires Vimscript
-- But you can use treesitter for advanced multi-language support
require('nvim-treesitter.configs').setup({
  ensure_installed = {'perl', 'verilog'},
  highlight = { enable = true }
})
```

Source: vim.fandom.com

17.88 Custom Syntax Highlighting for Specific Filetypes

Category: configuration
Tags: syntax, filetype, highlighting

Create custom syntax highlighting rules for specific file extensions by defining a custom syntax file and filetype detection

Example: Example (Vim)

```
" In ~/.vim/syntax/en.vim
syn region enConstant start=/</hs=e+1 end=/>/he=s-1
syn region enType start=/`/hs=e+1 end=/`/he=s-1

" In filetype.vim
```

```
au BufNewFile,BufRead *.en setf en
```

Example: Example (Neovim)

```
-- In ~/.config/nvim/ftdetect/en.lua
vim.cmd('augroup en_filetype')
vim.cmd('autocmd BufNewFile,BufRead *.en setfiletype en')
vim.cmd('augroup END')

-- In ~/.config/nvim/after/syntax/en.lua
local api = vim.api

api.nvim_create_autocmd('Syntax', {
  pattern = '*.en',
  callback = function()
    vim.cmd[[syn region enConstant start=/</hs=e+1 end=/>/he=s-1]]
    vim.cmd[[syn region enType start=/`/hs=e+1 end=/`/he=s-1]]
  end
})
```

Source: vim.fandom.com

17.89 Customize Completion Sources

Category: configuration
Tags: completion, options, customization

Configure where keywords are searched for completion using the 'complete' option, allowing more flexible word completion across files and sources

Example: Example (Vim)

```
" Customize completion sources
set complete+=i  " Include current and included files
set complete+=t  " Include tags
```

Example: Example (Neovim)

```
-- Configure completion sources
vim.opt.complete:append('i')  -- Include current and included files
vim.opt.complete:append('t')  -- Include tags
```

Source: vim.fandom.com

17.90 Customize Cursor Color and Shape Across Modes

Category: configuration
Tags: ui, cursor, terminal-config

Configure cursor appearance dynamically across different Vim modes, changing color and shape to provide visual mode indication

Example: Example (Vim)

```
" Change cursor color in different modes
highlight Cursor guifg=white guibg=black
highlight iCursor guifg=white guibg=steelblue
set guicursor=n-v-c:block-Cursor
set guicursor+=i:ver100-iCursor
```

Example: Example (Neovim)

```
-- Lua equivalent for cursor configuration
vim.cmd('highlight Cursor guifg=white guibg=black')
vim.cmd('highlight iCursor guifg=white guibg=steelblue')
vim.opt.guicursor = 'n-v-c:block-Cursor,i:ver100-iCursor'
```

Source: vim.fandom.com

17.91 Customize Errorformat for Different Compilers

Category: configuration
Tags: compiler, quickfix, error-handling

Configure Vim's errorformat to parse compiler and tool error messages, enabling quick navigation to error locations

Example: Example (Vim)

```
" Example: Add eslint error format
set errorformat+=%f:line %l\,col %c\,%trror -\ %m
set errorformat+=%f:line %l\,col %c\,%tarning -\ %m
```

Example: Example (Neovim)

```
-- Lua equivalent for errorformat configuration
vim.o.errorformat = vim.o.errorformat .. ',%f:line %l\,col %c\,%trror -\ %m'
vim.o.errorformat = vim.o.errorformat .. ',%f:line %l\,col %c\,%tarning -\
↪  %m'
```

Source: vim.fandom.com

17.92 Customize File Path Detection

Category: configuration
Tags: file-operations, configuration, path-handling

Modify 'isfname' and 'path' settings to improve file detection, especially with complex paths or spaces

Example: Example (Vim)

```
" Add space as valid filename character
set isfname+=32

" Add path for include files
set path+=C:/DOCUME~1/MYUSER~1/MYDOCU~1
```

Example: Example (Neovim)

```
-- Customize isfname to include space character
vim.opt.isfname:append(32)

-- Add custom paths for file searching
vim.opt.path:append('C:/DOCUME~1/MYUSER~1/MYDOCU~1')
```

Source: vim.fandom.com

17.93 Customize File Path Detection with isfname

Category: configuration
Tags: file-handling, customization

Modify 'isfname' option to include characters like spaces in file paths, expanding file detection capabilities

Example: Example (Vim)

```
" Allow spaces in filenames
set isfname+=32
```

Example: Example (Neovim)

```
-- Allow spaces in filenames
vim.opt.isfname:append(32)
```

Source: vim.fandom.com

17.94 Customize Perl Documentation Lookup

Category: configuration
Tags: perl, documentation, key-mapping

Remap 'K' in Perl files to use perldoc for function lookups instead of man pages

Example: Example (Vim)

```
au FileType perl setlocal keywordprg=perldoc\ -T\ -f
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'perl',
  callback = function()
    vim.opt_local.keywordprg = 'perldoc -T -f'
  end
})
```

Source: vim.fandom.com

17.95 Customize Python Triple-Quoted Comments

Category: configuration
Tags: syntax-highlighting, python, customization

Modify Python syntax highlighting to treat docstrings after a colon as comments instead of strings

Example: Example (Vim)

```
syn region pythonComment
      \ start=+\%(:\n\s*\)\@<=\z('''\|"""\)+ end=+\z1+ keepend
      \ contains=pythonEscape,pythonTodo,@Spell
```

Example: Example (Neovim)

```
vim.cmd[[syn region pythonComment
      \ start=+\%(:\n\s*\)\@<=\z('''\|"""\)+ end=+\z1+ keepend
      \ contains=pythonEscape,pythonTodo,@Spell]]
```

Source: vim.fandom.com

17.96 Customize Syntax Highlighting Colors

Category: configuration
Tags: color-scheme, syntax-highlighting, ui

Modify specific syntax highlighting colors for better readability

Example: Example (Vim)

```
hi Comment ctermfg=DarkGrey guifg=DarkGrey
hi Search guibg=LightBlue
highlight ErrorMsg guibg=White guifg=Red
```

Example: Example (Neovim)

```
-- Customize syntax highlighting colors
vim.cmd('hi Comment ctermfg=DarkGrey guifg=DarkGrey')
vim.cmd('hi Search guibg=LightBlue')
vim.cmd('highlight ErrorMsg guibg=White guifg=Red')
```

Source: vim.fandom.com

17.97 Customized Vim Configuration for Productivity

Category: configuration
Tags: settings, editor-config, productivity

A comprehensive vimrc configuration with useful defaults and settings to improve editing experience

Example: Example (Vim)

```
set nocompatible
set history=50
set undolevels=100
syntax on
set ignorecase
set tabstop=4
set shiftwidth=4
set background=light
set backspace=2
set noerrorbells
set laststatus=2
set statusline=%<%F%h%m%r%h%w%y\ %{&ff}\ line:%l\ col:%c%V
```

Example: Example (Neovim)

```
vim.opt.compatible = false
vim.opt.history = 50
vim.opt.undolevels = 100
vim.opt.syntax = 'on'
vim.opt.ignorecase = true
vim.opt.tabstop = 4
vim.opt.shiftwidth = 4
vim.opt.background = 'light'
vim.opt.backspace = '2'
vim.opt.errorbells = false
vim.opt.laststatus = 2
```

```
vim.opt.statusline = '%<%F%h%m%r%h%w%y\ %{&ff}\ line:%l\ col:%c%V'
```

Source: vim.fandom.com

17.98 Cycle Numeric Options with Flexible Range

Category: configuration
Tags: options, cycle, numeric-settings

Create a function to cycle numeric options within a specified range, useful for dynamically adjusting settings like tabstop or foldcolumn

Example: Example (Vim)

```
function CycleNum(option,min,inc,max)
  exec ('let tz_value = (((&'.a:option.'-'.a:min.')+'.a:inc.')%(('.a:max.'-
  ↪  '.a:min.')+'.a:inc.'))+'.a:min)
  if (tz_value < a:min) " in case inc<0
    let tz_value = tz_value+a:max
  endif
  exec ('setlocal '.a:option.'='.tz_value)
endfunction

" Example usage
noremap <silent> <F7> :call CycleNum("foldcolumn",0,2,6)<BAR>set
↪  foldcolumn?<CR>
```

Example: Example (Neovim)

```
function _G.cycle_num(option, min, inc, max)
  local current = vim.o[option]
  local next_val = ((current - min + inc) % (max - min + inc)) + min
  if next_val < min then
    next_val = next_val + max
  end
  vim.o[option] = next_val
  print(string.format('%s is now %d', option, next_val))
end

-- Example usage
vim.keymap.set('n', '<F7>', function() _G.cycle_num('foldcolumn', 0, 2, 6)
↪  end)
```

Source: vim.fandom.com

17.99 Debug Unexpected Vim Option Settings

Category: configuration
Tags: troubleshooting, configuration, debugging

Systematic approach to identifying and resolving unexpected Vim configuration issues by methodically isolating the source of problems

Example: Example (Vim)

```
" Debugging commands
:scriptnames  " List loaded scripts
:verbose set option?  " Check option source
gvim -u NONE -U NONE  " Start with default settings
gvim --noplugin  " Disable all plugins
```

Example: Example (Neovim)

```
-- Lua equivalents for debugging
-- List loaded scripts
vim.cmd('scriptnames')

-- Check option source (e.g., tabstop)
vim.cmd('verbose set tabstop?')

-- Binary search technique for .vimrc
function debug_vimrc()
  -- Use marks to narrow down problematic section
  -- Example pseudo-code for binary search through config
end
```

Source: vim.fandom.com

17.100 Detect Filetype by File Contents

Category: configuration
Tags: file-types, detection

Programmatically detect file type by examining first few lines of content

Example: Example (Vim)

```
if getline(1) =~ '-!'
  set ft=xcl
elseif getline(1) =~ '\/\*'
  set ft=c
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufRead', {
  callback = function()
    local first_line = vim.fn.getline(1)
    if first_line:match('-!') then
      vim.opt.filetype = 'xcl'
```

```
    elseif first_line:match('\/\*') then
      vim.opt.filetype = 'c'
    end
  end
})
```

Source: vim.fandom.com

17.101 Detect GUI Running in Vim/Neovim

Category: configuration
Tags: gui, environment-detection

Reliable ways to check if running in GUI mode for conditional configuration

Example: Example (Vim)

```
" Check if GUI is running
if has("gui_running")
  " GUI-specific settings
endif

" Alternative GUI check
if has("gui")
  " Another GUI-specific block
endif
```

Example: Example (Neovim)

```
-- Check if GUI is running in Lua
if vim.g.neovide or vim.fn.has('gui_running') == 1 then
  -- GUI-specific Neovim configuration
end
```

Source: vim.fandom.com

17.102 Diagnose Backspace Key Mapping Issues

Category: configuration
Tags: troubleshooting, key-mapping, insert-mode

Check for problematic backspace mappings in insert mode and remove them if needed

Example: Example (Vim)

```
" Check backspace mappings
:verbose imap <BS>
:verbose imap ^H
```

```
" Remove problematic mapping
:iunmap <BS>
```

Example: Example (Neovim)

```
-- Check backspace mappings
vim.cmd('verbose imap <BS>')
vim.cmd('verbose imap ^H')

-- Remove problematic mapping
vim.keymap.del('i', '<BS>')
```

Source: vim.fandom.com

17.103 Diagnose Vim Menu Bar Placement Issues

Category: configuration
Tags: gnome, troubleshooting, ui

Fix menu and toolbar overlay by modifying Gnome placement settings

Example: Example (Vim)

```
" Edit ~/.gnome2/Vim configuration
" Change Dock setting from:
" Dock=Toolbar\0,0,0,0\Menubar\0,0,0,0
" To:
" Dock=Toolbar\0,1,0,0\Menubar\0,0,0,0
```

Example: Example (Neovim)

```
-- Lua cannot directly edit this file, but can help diagnose
vim.notify('Check ~/.gnome2/Vim configuration for menu placement issues',
↪  vim.log.levels.WARN)
```

Source: vim.fandom.com

17.104 Diagnose and Clear Problematic Backspace Mappings

Category: configuration
Tags: key-mapping, troubleshooting, insert-mode

Check and remove unwanted backspace key mappings that might interfere with normal editing

Example: Example (Vim)

```
" Check backspace mappings
:verbose imap <BS>
" Remove problematic mapping
:iunmap <BS>
```

Example: Example (Neovim)

```
-- Check backspace mappings
vim.cmd('verbose imap <BS>')
-- Remove problematic mapping
vim.keymap.del('i', '<BS>')
```

Source: vim.fandom.com

17.105 Disable ANSI Escape Sequences in Man Pages

Category: configuration
Tags: man-pages, terminal, display

Prevent garbage characters when viewing man pages by disabling SGR (ANSI escape sequences)

Example: Example (Vim)

```
let $GROFF_NO_SGR=1
```

Example: Example (Neovim)

```
vim.env.GROFF_NO_SGR = '1'
```

Source: vim.fandom.com

17.106 Disable Auto-Indentation for Specific Filetypes

Category: configuration
Tags: indentation, filetype, customization

Disable automatic indentation for specific file types while keeping it enabled for others

Example: Example (Vim)

```
let b:did_indent = 1
```

Example: Example (Neovim)

```
-- For Neovim, create a specific indent file for the filetype
-- Example for HTML: ~/.config/nvim/indent/html.lua
vim.g.did_indent = 1
```

Source: vim.fandom.com

17.107 Disable Automatic Comment Continuation

Category: configuration
Tags: comments, formatting, autocmd

Prevent automatic comment insertion when continuing lines

Example: Example (Vim)

```
autocmd FileType * setlocal formatoptions-=c formatoptions-=r
↪  formatoptions-=o
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = '*',
  callback = function()
    vim.opt_local.formatoptions:remove({'c', 'r', 'o'})
  end
})
```

Source: vim.fandom.com

17.108 Disable Automatic Comment Insertion

Category: configuration
Tags: comments, formatting, productivity

Prevent Vim from automatically inserting comment characters when creating new lines in comments

Example: Example (Vim)

```
" Disable auto-comment for all file types
autocmd FileType * setlocal formatoptions-=c formatoptions-=r
↪  formatoptions-=o
```

Example: Example (Neovim)

```
-- Disable auto-comment for all file types
vim.api.nvim_create_autocmd('FileType', {
  pattern = '*',
  callback = function()
    vim.opt_local.formatoptions:remove({'c', 'r', 'o'})
  end
})
```

Source: vim.fandom.com

17.109 Disable Default Filetype Plugins

Category: configuration
Tags: filetype, plugins, customization

Disable all default filetype plugins or prepare for custom ftplugin handling

Example: Example (Vim)

```
autocmd BufReadPre,BufNewFile * let b:did_ftplugin = 1
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufReadPre', 'BufNewFile'}, {
  callback = function()
    vim.b.did_ftplugin = 1
  end
})
```

Source: vim.fandom.com

17.110 Disable IME Input Mode in Windows

Category: configuration
Tags: windows, internationalization, input-method

Prevent automatic IME (Input Method Editor) activation in Vim when using non-English locales

Example: Example (Vim)

```
set iminsert=0

# Optional: Disable IME completely
set imdisable
```

Example: Example (Neovim)

```
-- Disable IME input mode
vim.opt.iminsert = 0

-- Optional: Completely disable IME
vim.g.imdisable = true
```

Source: vim.fandom.com

17.111 Disable Mouse Visual Mode Selection

Category: configuration
Tags: mouse, ui, interaction

Prevent accidental visual mode activation when dragging mouse, allowing normal mouse interactions

Example: Example (Vim)

```
noremap <LeftDrag> <LeftMouse>
noremap! <LeftDrag> <LeftMouse>
```

Example: Example (Neovim)

```
vim.keymap.set({'n', 'v', 'i', 'c'}, '<LeftDrag>', '<LeftMouse>', { noremap
↪  = true, silent = true })
```

Source: vim.fandom.com

17.112 Disable Python Omnicomplete Preview Window

Category: configuration
Tags: python, completion, omnicomplete

Prevents the Pydoc preview window from appearing during Python code completion

Example: Example (Vim)

```
set completeopt-=preview

autocmd FileType python set omnifunc=pythoncomplete#Complete
```

Example: Example (Neovim)

```
vim.opt.completeopt:remove('preview')

vim.api.nvim_create_autocmd('FileType', {
```

```
    pattern = 'python',
    callback = function()
      vim.bo.omnifunc = 'pythoncomplete#Complete'
    end
  })
```

Source: vim.fandom.com

17.113 Disable Terminal Flow Control for Vim

Category: configuration
Tags: terminal, settings, workflow

Prevent terminal from intercepting Ctrl-S, allowing it to work as a Vim shortcut

Example: Example (Vim)

```
" Disable terminal flow control
stty -ixon  # Add to .bashrc
```

Example: Example (Neovim)

```
-- For bash/zsh config
-- Add 'stty -ixon' to your shell's startup file
-- Alternatively, in Neovim terminal settings:
vim.g.terminal_disable_flow_control = true
```

Source: vim.fandom.com

17.114 Disable Vim Bells and Visual Flashes

Category: configuration
Tags: ui, sound, annoyance-removal

Completely disable error bells and visual flashes in Vim/Neovim, which can be distracting during editing

Example: Example (Vim)

```
set noerrorbells visualbell t_vb=
autocmd GUIEnter * set visualbell t_vb=
```

Example: Example (Neovim)

```
vim.opt.errorbells = false
vim.opt.visualbell = true
vim.opt.belloff = 'all'
```

```
-- Alternative with autocmd
vim.api.nvim_create_autocmd('GUIEnter', {
  callback = function()
    vim.opt.visualbell = true
    vim.opt.t_vb = ''
  end
})
```

Source: vim.fandom.com

17.115 Discover Special Key Names in Vim

Category: configuration
Tags: key-mapping, key-codes, vim-tips

Learn how to identify special key names in Vim, which is crucial for creating custom key mappings

Example: Example (Vim)

```
" In insert mode:
i
Ctrl-K + Special Key (e.g., Ctrl-Left)
" This will reveal the key name like <C-Left>
```

Example: Example (Neovim)

```
-- Use vim.inspect() to help discover key names
-- In insert mode, Ctrl-K followed by a special key reveals its name
-- For programmatic key discovery, use vim.keycode() function
```

Source: vim.fandom.com

17.116 Display Indentation Level in Status Line

Category: configuration
Tags: status-line, indentation, custom-function

Add a custom function to show the current indentation level in the status line, which helps track indentation depth

Example: Example (Vim)

```
set statusline=<Whatever your status is>\t%{ShowTab()}\ %P
fu ShowTab()
  let TabLevel = (indent('.') / &ts )
  if TabLevel == 0
```

```
    let TabLevel='*'
  endif
  return TabLevel
endf
```

Example: Example (Neovim)

```
vim.opt.statusline = vim.o.statusline .. '%{v:lua.ShowTab()}'

_G.ShowTab = function()
  local tabLevel = math.floor(vim.fn.indent('.') / vim.o.tabstop)
  return tabLevel == 0 and '*' or tostring(tabLevel)
end
```

Source: vim.fandom.com

17.117 Display Non-ASCII Characters in Console

Category: configuration
Tags: character-display, encoding, console-support

Configure Vim to correctly display non-ASCII characters like umlauts in console environments

Example: Example (Vim)

```
set isprint=@,128-255
```

Example: Example (Neovim)

```
vim.opt.isprint:append({128, 255})
```

Source: vim.fandom.com

17.118 Download Vim for Windows

Category: configuration
Tags: windows, installation, download

Multiple options for downloading Vim on Windows, including 32-bit and 64-bit builds

Example: Example (Vim)

```
" Recommended download sources:
" 1. Nightly builds: https://github.com/vim/vim-win32-installer
" 2. Stable builds: http://sourceforge.net/projects/cream/files/Vim/
```

Example: Example (Neovim)

```
-- Use recommended download sources
-- Nightly builds: https://github.com/vim/vim-win32-installer
-- Stable builds: http://sourceforge.net/projects/cream/files/Vim/
```

Source: vim.fandom.com

17.119 Dynamic Color Scheme for Context Awareness

Category: configuration
Tags: autocmd, colorscheme, ui

Automatically change color scheme based on file type, directory, or Vim instance to improve visual context awareness

Example: Example (Vim)

```
" Change color scheme by file type
au BufEnter * if (exists("b:colors_name")) | let
 ↪  b:current_colors=colors_name
 \ | execute "colorscheme " . b:colors_name | endif
au BufLeave * if (exists("b:current_colors")) | execute "colorscheme " .
 ↪  b:current_colors | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    if vim.b.colors_name then
      vim.g.current_colors = vim.g.colors_name
      vim.cmd('colorscheme ' .. vim.b.colors_name)
    end
  end
})

vim.api.nvim_create_autocmd('BufLeave', {
  callback = function()
    if vim.g.current_colors then
      vim.cmd('colorscheme ' .. vim.g.current_colors)
    end
  end
})
```

Source: vim.fandom.com

17.120 Dynamic Cscope Database Generation

Category: configuration
Tags: project-management, code-navigation, build-tools

Add Makefile targets to automatically generate Cscope databases for code navigation, ensuring your project's symbol database stays up-to-date

Example: Example (Vim)

```
# Makefile target for generating cscope database
cscope: csclean cscope.out

cscope.out:
        cscope -bv ./*.[ch] ./*.cpp ./if_perl.xs auto/*.h
```

Example: Example (Neovim)

```
-- For Lua, you would typically use a separate shell script or :make command
-- Example Lua function to trigger cscope database generation
function generate_cscope_db()
  vim.fn.system('cscope -bv ./*.[ch] ./*.cpp')
  vim.cmd('cs reset')
end
```

Source: vim.fandom.com

17.121 Dynamic Doxygen Comment Configuration

Category: configuration
Tags: comments, autocmd, file-types

Flexible method to configure comment styles for different file types without overwriting existing settings

Example: Example (Vim)

```
autocmd FileType c,cpp set comments-=://
autocmd FileType c,cpp set comments+=:///
autocmd FileType c,cpp set comments+=://
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'c', 'cpp'},
  callback = function()
    -- Remove existing // comments
    local current_comments = vim.opt.comments:get()
    table.remove(current_comments, '://')

    -- Add new comment styles
    vim.opt.comments = current_comments
    vim.opt.comments:append(':///'):append(':/')
  end
})
```

Source: vim.fandom.com

17.122 Dynamic File-Specific Vim Settings with Modelines

Category: configuration
Tags: file-settings, per-file-config, vim-options

Modelines allow you to set Vim options specific to individual files, overriding global settings. Useful for project-specific formatting or editor preferences.

Example: Example (Vim)

```
# vim: set expandtab ts=4 sw=4:
# Add this near top/bottom of file to set tab behavior
```

Example: Example (Neovim)

```
-- In Neovim, modelines work similarly
-- Add to file: # vim: set expandtab ts=4 sw=4:
-- Or configure in init.lua
vim.o.modeline = true
vim.o.modelines = 5  -- Check first/last 5 lines for modeline
```

Source: vim.fandom.com

17.123 Dynamic Include Path for C++ Intellisense

Category: configuration
Tags: c++, path-configuration, development

Dynamically set the include path for C++ header files relative to the current file's directory, enabling easier header file resolution

Example: Example (Vim)

```
let $INCLUDE = expand("%:p:h") . ";" . $INCLUDE
```

Example: Example (Neovim)

```
-- Set include path dynamically for C++ projects
vim.env.INCLUDE = vim.fn.expand('%:p:h') .. ';' .. (vim.env.INCLUDE or '')
```

Source: vim.fandom.com

17.124 Dynamic PHP Function Documentation Lookup

Category: configuration
Tags: documentation, external-tools, mapping

Set up a custom keywordprg to quickly look up PHP function documentation directly from Vim using an external script

Example: Example (Vim)

```
set keywordprg=~/.vim/php_doc

" In .vimrc
autocmd FileType php set keywordprg=~/.vim/php_doc
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'php',
  callback = function()
    vim.opt.keywordprg = '~/.vim/php_doc'
  end
})
```

Source: vim.fandom.com_on_line_help)

17.125 Dynamic Syntax Sync for Performance

Category: configuration
Tags: syntax, performance, optimization

Improve syntax highlighting speed by searching backwards a configurable number of lines

Example: Example (Vim)

```
syntax sync minlines=200
```

Example: Example (Neovim)

```
vim.opt.synmaxline = 200
```

Source: vim.fandom.com

17.126 Dynamically Detect and Handle Screen Terminal

Category: configuration
Tags: terminal, environment, conditional-config

Detect when running in Screen and adjust Vim terminal settings accordingly

Example: Example (Vim)

```
if match($TERM, "screen")!=-1
  set term=xterm
  let g:GNU_Screen_used = 1
endif
```

Example: Example (Neovim)

```
if vim.env.TERM:match("screen") then
  vim.o.term = "xterm"
  vim.g.GNU_Screen_used = true
end
```

Source: vim.fandom.com

17.127 Dynamically Manage Vim Path Settings

Category: configuration
Tags: path-management, include-paths, project-configuration

Create a flexible function to dynamically switch between different include path configurations for code completion and file searching

Example: Example (Vim)

```
function Feral_Path(whichpaths)
  if strlen(a:whichpaths)
    let old_path = &path
    set path=
    let paths_to_use = toupper(a:whichpaths)
    while strlen(paths_to_use)
      if exists("g:feral_path_{paths_to_use[0]}")
        execute "set path+=" . substitute(g:feral_path_{paths_to_use[0]}, "\
        ↪  ", "\\\\ ", "g")
        let paths_to_use = strpart(paths_to_use,1)
      else
        let paths_to_use = strpart(paths_to_use,1)
      endif
    endwhile
    if strlen(&path) == 0
      execute "set path=" . old_path
    endif
```

```
  endif
endfunction

command -nargs=1 FP call Feral_Path(<q-args>)
```

Example: Example (Neovim)

```
function _G.feral_path(whichpaths)
  if #whichpaths > 0 then
    local old_path = vim.o.path
    vim.o.path = ''
    local paths_to_use = whichpaths:upper()

    for i = 1, #paths_to_use do
      local var_name = 'g:feral_path_' .. paths_to_use:sub(i,i)
      if vim.g[var_name] then
        vim.o.path = vim.o.path .. ',' .. vim.g[var_name]:gsub(' ', '\\ ')
      end
    end

    if vim.o.path == '' then
      vim.o.path = old_path
    end
  end
end

vim.api.nvim_create_user_command('FP', function(opts)
  _G.feral_path(opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

17.128 Easy vimrc File Management

Category: configuration
Tags: configuration, file-management, vim-setup

Quickly open and source your vimrc file using environment variables

Example: Example (Vim)

```
" Open vimrc
:e $MYVIMRC

" Source vimrc without restarting
:so $MYVIMRC
```

Example: Example (Neovim)

```
-- Open vimrc
vim.cmd.edit(vim.env.MYVIMRC)

-- Source vimrc
vim.cmd.source(vim.env.MYVIMRC)
```

Source: vim.fandom.com

17.129 Enable 256 Color Support in Vim/Neovim

Category: configuration
Tags: color, terminal, display

Configures Vim/Neovim to use 256 color mode, improving color rendering in terminal environments

Example: Example (Vim)

```
set t_Co=256
set t_AB=^[[48;5;%dm
set t_AF=^[[38;5;%dm
```

Example: Example (Neovim)

```
-- Enable 256 color support
vim.opt.termguicolors = true

-- Optional: Additional terminal color configuration
vim.g.terminal_color_count = 256
```

Source: vim.fandom.com

17.130 Enable Filetype Detection and Plugins

Category: configuration
Tags: file-type, plugin-management

Ensure filetype detection and plugin loading are enabled for language-specific configurations

Example: Example (Vim)

```
" Enable filetype detection and indent plugins
filetype plugin indent on
```

Example: Example (Neovim)

```
-- Enable filetype detection and indent plugins
vim.cmd('filetype plugin indent on')
```

Source: vim.fandom.com

17.131 Enable Mouse Support in Terminal Vim

Category: configuration
Tags: mouse, terminal, interaction

Enable full mouse support in terminal Vim, allowing mouse interactions like selection and scrolling

Example: Example (Vim)

```
set mouse=a
```

Example: Example (Neovim)

```
vim.opt.mouse = 'a'
```

Source: vim.fandom.com

17.132 Enable Omni Completion for Smart Autocompletion

Category: configuration
Tags: completion, autocomplete, programming

Enable language-aware autocompletion that provides intelligent suggestions based on context

Example: Example (Vim)

```
filetype plugin on
set omnifunc=syntaxcomplete#Complete
```

Example: Example (Neovim)

```
vim.cmd('filetype plugin on')
vim.opt.omnifunc = 'syntaxcomplete#Complete'
```

Source: vim.fandom.com

17.133 Enable Windows Shortcuts in GVim

Category: configuration
Tags: windows, shortcut-keys, window-management

Configure winaltkeys to enable standard Windows window management shortcuts in Vim

Example: Example (Vim)

```
set winaltkeys=yes

" Now you can use:
" Alt+Space followed by 'x' to maximize
" Alt+Space followed by 'n' to minimize
" Alt+Space followed by 'r' to restore
```

Example: Example (Neovim)

```
-- For Neovim, this can be set in init.lua
vim.g.winaltkeys = 'yes'

-- Note: Behavior may vary slightly in Neovim compared to classic Vim
```

Source: vim.fandom.com

17.134 Enable Windows Shortcuts in gVim

Category: configuration
Tags: windows, shortcut-keys, window-management

Configure Alt+Space shortcuts for window control in Vim on Windows

Example: Example (Vim)

```
set winaltkeys=yes
```

Example: Example (Neovim)

```
vim.g.winaltkeys = 'yes'
```

Source: vim.fandom.com

17.135 Enable Windows-like Selection Mode

Category: configuration
Tags: selection, key-model

Configure Vim to use Windows-like text selection behavior

Example: Example (Vim)

```
set keymodel=startsel,stopsel
```

Example: Example (Neovim)

```
vim.opt.keymodel = {'startsel', 'stopsel'}
```

Source: vim.fandom.com

17.136 Enhance Vim Usability with Essential Settings

Category: configuration
Tags: usability, search, interface

Set up core Vim settings to improve editing experience, including case-insensitive search, mouse support, and visual enhancements

Example: Example (Vim)

```
" Essential usability settings
set ignorecase
set smartcase
set mouse=a
set number
set ruler
set hlsearch
```

Example: Example (Neovim)

```
-- Enhance Vim usability
vim.opt.ignorecase = true
vim.opt.smartcase = true
vim.opt.mouse = 'a'
vim.opt.number = true
vim.opt.ruler = true
vim.opt.hlsearch = true
```

Source: vim.fandom.com

17.137 Enhanced COBOL Syntax Highlighting for Modern COBOL

Category: configuration
Tags: syntax, filetype, language-support

Use an updated cobol.vim syntax file that supports modern COBOL features including

object-oriented syntax

Example: Example (Neovim)

```
-- Manually place the updated syntax file in your Neovim syntax directory
-- Can be found in the comments section of the original wiki page
```

Source: vim.fandom.com

17.138 Enhanced Linux Console Color Configuration

Category: configuration
Tags: terminal, color-scheme, console

Configure 16 background colors in Linux console with bright color support, improving visual appearance and readability

Example: Example (Vim)

```
if &term =~ "linux"
  if has("terminfo")
    set t_Co=16
    set t_AB=<Esc>[%?%p1%{7}%>%t5%p1%{8}%-%e25%p1%;m<Esc>[4%dm
    set t_AF=<Esc>[%?%p1%{7}%>%t1%p1%{8}%-%e22%p1%;m<Esc>[3%dm
  endif
endif
```

Example: Example (Neovim)

```
if vim.env.TERM:match('linux') then
  vim.opt.t_Co = 16
  -- Note: Translating terminal codes in Lua might require additional
  ↪  terminal configuration
  -- Consult your specific terminal emulator's documentation
end
```

Source: vim.fandom.com

17.139 Ensure Vim Compatibility Mode is Off

Category: configuration
Tags: compatibility, settings

Disable Vi compatibility mode to enable modern Vim features and fix common issues like backspace behavior

Example: Example (Vim)

```
set nocompatible
```

Example: Example (Neovim)

```
vim.o.compatible = false
```

Source: vim.fandom.com

17.140 Environment Variable Management for Custom Vim

Category: configuration
Tags: environment, vim-setup, custom-build

Set VIM, VIMRUNTIME, and PATH environment variables after compiling a custom Vim build to manage multiple executable versions

Example: Example (Vim)

```
# After make, set custom environment variables
export VIM=/path/to/custom/vim
export VIMRUNTIME=/path/to/custom/runtime
export PATH=$PATH:/path/to/custom/bin
```

Example: Example (Neovim)

```
-- Lua equivalent for managing environment variables
vim.env.VIM = '/path/to/custom/vim'
vim.env.VIMRUNTIME = '/path/to/custom/runtime'
vim.env.PATH = vim.env.PATH .. ':/path/to/custom/bin'
```

Source: vim.fandom.com

17.141 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: Example (Vim)

```
-- In init.lua:
if os.getenv("MACHINE") == "work" then
  -- Work-specific configuration
```

```
  vim.opt.colorcolumn = "80"
else
  -- Personal configuration
  vim.opt.colorcolumn = "120"
end
```

Source: Community contributed

17.142 Escape Spaces in Font Names

Category: configuration
Tags: font, escaping, gui

Properly escape font names with spaces when setting guifont

Example: Example (Vim)

```
set guifont=Monospace\ 10
```

Example: Example (Neovim)

```
vim.opt.guifont = 'Monospace\ 10'
```

Source: vim.fandom.com

17.143 Essential Vim Configuration Settings

Category: configuration
Tags: settings, usability, startup

A set of recommended Vim configuration options that improve overall editor usability and behavior

Example: Example (Vim)

```
" Core configuration settings
set nocompatible
set hidden
set wildmenu
set showcmd
set hlsearch
set ignorecase
set smartcase
set backspace=indent,eol,start
set autoindent
```

Example: Example (Neovim)

```
-- Equivalent Neovim configuration
vim.opt.compatible = false
vim.opt.hidden = true
vim.opt.wildmenu = true
vim.opt.showcmd = true
vim.opt.hlsearch = true
vim.opt.ignorecase = true
vim.opt.smartcase = true
vim.opt.backspace = {'indent', 'eol', 'start'}
vim.opt.autoindent = true
```

Source: vim.fandom.com

17.144 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: Example (Vim)

```
" Vimscript:
: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
```

Example: Example (Neovim)

```
-- Lua:
-- Run python after save
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = '*.py',
  command = '!python %',
  desc = 'Run python after save'
})

-- Clear all BufRead autocmds - use augroup for better control
vim.api.nvim_clear_autocmds({ event = 'BufRead' })

-- Trigger BufRead event
vim.api.nvim_exec_autocmds('BufRead', { pattern = '*' })

-- Python-specific settings
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.opt_local.tabstop = 4
```

```
  end,
  desc = 'Python-specific settings'
})
```

Source: Community contributed

17.145 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: Example (Vim)

```
" Vimscript:
: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
```

Example: Example (Neovim)

```
-- Lua:
vim.cmd('highlight Comment ctermfg=green')  -- set comment color
vim.cmd('syntax on')   -- enable syntax highlighting
vim.cmd('syntax off')  -- disable syntax highlighting
vim.cmd('colorscheme desert')  -- change color scheme
vim.cmd('highlight clear')  -- clear all highlighting

-- Or using Lua API for highlight:
vim.api.nvim_set_hl(0, 'Comment', { ctermfg = 'green' })
```

Source: Community contributed

17.146 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: Example (Vim)

```
" Vimscript:
: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
```

Example: Example (Neovim)

```
-- Lua:
vim.keymap.set('n', '<F2>', ':w<CR>', { desc = 'Save file' })
vim.keymap.set('i', '<F3>', '<Esc>:w<CR>', { desc = 'Save file from insert
↪  mode' })
vim.cmd('abbrev teh the')  -- abbreviation for typo
vim.keymap.del('n', '<F2>')  -- remove mapping
vim.cmd('unabbrev teh')  -- remove abbreviation
```

Source: Community contributed

17.147 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: Example (Vim)

```
" Vimscript:
:source ~/.vimrc        " load configuration file
:runtime! plugin/**/*.vim  " load all plugins
:scriptnames            " list all loaded scripts
:source %               " reload current file as script
```

Example: Example (Neovim)

```
-- Lua:
vim.cmd('source ~/.vimrc')  -- load configuration file
vim.cmd('runtime! plugin/**/*.vim')  -- load all plugins
vim.cmd('scriptnames')  -- list all loaded scripts
vim.cmd('source %')  -- reload current file as script

-- Or using Lua API:
dofile(vim.fn.expand('~/.vimrc'))  -- load Vimscript file
-- For Lua files:
dofile(vim.fn.expand('~/.config/nvim/init.lua'))
```

Source: Community contributed

17.148 Extend Keyword Characters for Namespaces

Category: configuration
Tags: keyword, syntax, xml

Add special characters like colon to keyword definition for XML/namespace support

Example: Example (Vim)

```
set iskeyword+=:
```

Example: Example (Neovim)

```
vim.opt.iskeyword:append(':')
```

Source: vim.fandom.com

17.149 Extend iskeyword for XML Namespaces

Category: configuration
Tags: syntax, autocomplete, xml

Add colon to iskeyword to support XML namespaces without breaking other settings

Example: Example (Vim)

```
set iskeyword+=:
```

Example: Example (Neovim)

```
vim.opt.iskeyword:append(':')
```

Source: vim.fandom.com

17.150 Fine-tune Terminal Color Palette

Category: configuration
Tags: color, syntax-highlighting, terminal-setup

Provides specific color hex values for customizing terminal color mappings to enhance syntax highlighting

Example: Example (Neovim)

```
-- Example color customization
vim.g.terminal_color_0 = '#4D4D4D'  -- ANSI Black
vim.g.terminal_color_1 = '#98FB98'   -- ANSI Green
```

Source: vim.fandom.com

17.151 Fix Arrow Keys in Remote Shell Terminals

Category: configuration
Tags: terminal, key-mapping, remote-editing

Solve arrow key issues in remote shell environments by configuring Vim/Neovim to correctly interpret terminal key codes

Example: Example (Vim)

```
" Configure terminal key codes
set t_ku=^[OA
set t_kd=^[OB
set t_kr=^[OC
set t_kl=^[OD
```

Example: Example (Neovim)

```
-- Lua equivalent for terminal key configuration
vim.opt.t_ku = '^[OA'
vim.opt.t_kd = '^[OB'
vim.opt.t_kr = '^[OC'
vim.opt.t_kl = '^[OD'
```

Source: vim.fandom.com

17.152 Fix Arrow Keys in Remote Shell Vim

Category: configuration
Tags: terminal, remote-shell, key-mapping

Resolve arrow key issues in Vim when using remote shells or terminals that display unexpected characters

Example: Example (Vim)

```
" Solution 2: Dynamically set terminal key codes
set t_ku=^[OA
set t_kd=^[OB
set t_kr=^[OC
set t_kl=^[OD
```

Example: Example (Neovim)

```
-- Lua equivalent for terminal key code configuration
vim.o.t_ku = '\x1b0A'
vim.o.t_kd = '\x1b0B'
vim.o.t_kr = '\x1b0C'
vim.o.t_kl = '\x1b0D'
```

Source: vim.fandom.com

17.153 Fix Arrow Keys in Visual Mode on Windows

Category: configuration
Tags: visual-mode, windows-compatibility

Prevents arrow keys from exiting visual mode in Windows Vim/Neovim environments

Example: Example (Vim)

```
set keymodel-=stopsel
```

Example: Example (Neovim)

```
vim.o.keymodel = vim.o.keymodel:gsub('stopsel', '')
```

Source: vim.fandom.com

17.154 Fix Backspace Behavior in Insert Mode

Category: configuration
Tags: insert-mode, editing, key-mapping

Configure backspace to work like most modern text editors, allowing deletion across line breaks, indentation, and insert start points

Example: Example (Vim)

```
set backspace=indent,eol,start
```

Example: Example (Neovim)

```
vim.opt.backspace = {'indent', 'eol', 'start'}
```

Source: vim.fandom.com

17.155 Fix Backspace Key in PuTTY Terminal

Category: configuration
Tags: terminal, key-mapping, remote-connection

Resolve backspace key issues when connecting to Linux servers via PuTTY by changing keyboard settings

Example: Example (Vim)

```
" In PuTTY settings:
" 1. Set Terminal Keyboard to 'linux'
" 2. Set Backspace Key to 'Control-H'
```

Example: Example (Neovim)

```
-- Neovim terminal configuration
-- Typically handled in PuTTY settings, not directly in Neovim
-- Ensure terminal emulator is configured correctly
```

Source: vim.fandom.com

17.156 Fix Backspace in PuTTY Terminal Vim

Category: configuration
Tags: terminal, putty, keyboard-config

Resolve backspace key issues when connecting to Linux servers via PuTTY by adjusting terminal keyboard settings

Example: Example (Vim)

```
" In PuTTY Configuration:
" - Set Keyboard to 'linux' instead of default 'xterm'
" - Set Backspace Key to 'Control-H'
```

Example: Example (Neovim)

```
-- Configuration is done in PuTTY settings, not in Neovim
-- 1. Change Terminal Keyboard to 'linux'
-- 2. Change Backspace Key setting to 'Control-H'
```

Source: vim.fandom.com

17.157 Fix Broken Syntax Highlighting

Category: configuration
Tags: syntax, performance, highlighting

Provides multiple methods to fix syntax highlighting when it becomes incorrect, with options for performance and accuracy

Example: Example (Vim)

```
" Method 1: Sync from start of file
autocmd BufEnter * :syntax sync fromstart

" Method 2: Quick mapping to fix highlighting
noremap <F12> <Esc>:syntax sync fromstart<CR>
inoremap <F12> <C-o>:syntax sync fromstart<CR>

" Method 3: Sync with a specific number of lines
syntax sync minlines=200
```

Example: Example (Neovim)

```
-- Method 1: Sync from start of file
vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '*',
  callback = function()
    vim.cmd('syntax sync fromstart')
  end
})

-- Method 2: Quick mapping to fix highlighting
vim.keymap.set('n', '<F12>', ':syntax sync fromstart<CR>', { desc = 'Fix
↪  syntax highlighting' })
vim.keymap.set('i', '<F12>', '<C-o>:syntax sync fromstart<CR>', { desc =
↪  'Fix syntax highlighting' })

-- Method 3: Sync with a specific number of lines
vim.o.synmaxline = 200
```

Source: vim.fandom.com

17.158 Fix Crontab Editing in Vim/Neovim

Category: configuration
Tags: crontab, file-editing, system-config

Ensure crontab can be edited in-place by setting backup options correctly

Example: Example (Vim)

```
autocmd FileType crontab setlocal nobackup nowritebackup

# Alternative approach
au BufEnter /private/tmp/crontab.* setl backupcopy=yes
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'crontab',
  callback = function()
    vim.opt_local.backup = false
    vim.opt_local.writebackup = false
  end
})

# Alternative approach
vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '/private/tmp/crontab.*',
  callback = function()
    vim.opt_local.backupcopy = 'yes'
  end
})
```

Source: vim.fandom.com

17.159 Fix Cygwin Executable Path in Windows Taskbar Vim

Category: configuration
Tags: windows, system-path, environment

Resolve issues with Cygwin executable paths when launching Vim from Windows 7 taskbar by setting a non-blank 'Start in' directory

Example: Example (Vim)

```
# No direct Vimscript implementation - this is a Windows shortcut
↪  configuration tip
```

Example: Example (Neovim)

```
-- Lua tip: Ensure proper path configuration in Windows environment
-- Modify taskbar shortcut 'Start in' property to
↪  'C:\Users\username\Documents'
```

Source: vim.fandom.com

17.160 Fix Esc Key and Terminal Keymappings in Screen

Category: configuration
Tags: terminal, screen, key-mapping

Resolve key mapping issues when using Vim inside GNU Screen, ensuring proper Esc key and special key functionality

Example: Example (Vim)

```
if match($TERM, "screen")!=-1
  set term=xterm
endif

" Fix special keys
map ^[[1~ <Home>
map ^[[4~ <End>
imap ^[[1~ <Home>
imap ^[[4~ <End>
```

Example: Example (Neovim)

```
if vim.env.TERM:match("screen") then
  vim.opt.term = "xterm"
end

-- Fix special keys
vim.keymap.set({'n', 'i'}, '<Home>', '<Home>', { noremap = true })
vim.keymap.set({'n', 'i'}, '<End>', '<End>', { noremap = true })
```

Source: vim.fandom.com

17.161 Fix Escape Key Timing in Screen Terminal

Category: configuration
Tags: terminal, screen, key-mapping

Resolves Escape key detection issues in GNU Screen by setting a short timeout for key sequences

Example: Example (Vim)

```
# In .screenrc
maptimeout 5
```

Example: Example (Neovim)

```
-- No direct Lua equivalent, this is a Screen configuration
-- Add to ~/.screenrc
```

Source: vim.fandom.com

17.162 Fix Meta/Alt Keys in Terminal Vim

Category: configuration
Tags: terminal, key-mapping, meta-keys

Configures Vim to properly handle Meta (Alt) key sequences in terminal emulators that generate escape sequences

Example: Example (Vim)

```
" Fix meta-keys which generate <Esc>a .. <Esc>z
let c='a'
while c <= 'z'
  exec "set <M-".toupper(c).">=\e".c
  exec "imap \e".c." <M-".toupper(c).">"
  let c = nr2char(1+char2nr(c))
endw
```

Example: Example (Neovim)

```
-- Fix meta-keys in Neovim terminal
for i = string.byte('a'), string.byte('z') do
  local char = string.char(i)
  local upper_char = string.upper(char)
  vim.api.nvim_set_keymap('i', '\027' .. char, '<M-' .. upper_char .. '>', {
  ↪  noremap = true, silent = true })
end
```

Source: vim.fandom.com

17.163 Fix Missing GVim Menu Bar in GNOME

Category: configuration
Tags: gnome, ui, troubleshooting

Resolve GNOME menu bar display issues in GVim by modifying Gnome settings or using specific launch commands

Example: Example (Vim)

```
:set guioptions+=m  " Enable menu bar
:set guioptions-=m  " Disable menu bar
```

Example: Example (Neovim)

```
-- For menu bar toggle in Neovim
vim.o.guioptions = vim.o.guioptions:gsub('m', '')  -- Toggle menu bar

-- Ubuntu Unity workaround
os.execute('UBUNTU_MENUPROXY="" nvim')
```

Source: vim.fandom.com

17.164 Fix Modeline File Encoding Detection

Category: configuration
Tags: encoding, file-operations, autocmd

Automatically re-read file with correct encoding when modeline specifies a different encoding

Example: Example (Vim)

```
function! CheckFileEncoding()
  if exists('b:fenc_at_read') && &fileencoding != b:fenc_at_read
    exec 'e! ++enc=' . &fileencoding
    unlet b:fenc_at_read
  endif
endfunction

au BufRead     *.txt let b:fenc_at_read=&fileencoding
au BufWinEnter *.txt call CheckFileEncoding()
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufRead', {
  pattern = '*.txt',
  callback = function()
    vim.b.fenc_at_read = vim.o.fileencoding
  end
})

vim.api.nvim_create_autocmd('BufWinEnter', {
  pattern = '*.txt',
  callback = function()
    if vim.b.fenc_at_read and vim.o.fileencoding ~= vim.b.fenc_at_read then
      vim.cmd('e! ++enc=' .. vim.o.fileencoding)
      vim.b.fenc_at_read = nil
    end
  end
})
```

Source: vim.fandom.com

17.165 Fix PuTTY Numeric Keypad Behavior

Category: configuration
Tags: terminal, putty, key-mapping

Disable PuTTY's application keypad mode to ensure numeric keypad works correctly in Vim

Example: Example (Vim)

```
" No direct Vim configuration needed
" Configure in PuTTY settings
```

Example: Example (Neovim)

```
-- Requires manual PuTTY configuration:
-- 1. Open PuTTY Configuration
-- 2. Go to Terminal > Features
-- 3. Check 'Disable application keypad mode'
-- 4. Save session settings
```

Source: vim.fandom.com

17.166 Fix Shell Command Issues in FreeBSD GVim

Category: configuration
Tags: shell, freebsd, compatibility

Resolve shell command execution problems in GVim on FreeBSD by changing the shell to ksh

Example: Example (Vim)

```
set sh=ksh
```

Example: Example (Neovim)

```
vim.opt.shell = 'ksh'
```

Source: vim.fandom.com

17.167 Fix Syntax Highlighting Globally

Category: configuration
Tags: syntax, performance, autocmd

Ensure consistent syntax highlighting by rescanning buffer from start, useful for com-

plex files

Example: Example (Vim)

```
autocmd BufEnter * :syntax sync fromstart
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    vim.cmd('syntax sync fromstart')
  end
})
```

Source: vim.fandom.com

17.168 Fix Syntax Highlighting Reliably

Category: configuration
Tags: syntax, performance, highlighting

Synchronize syntax highlighting for large or complex files, ensuring accurate highlighting across the entire document

Example: Example (Vim)

```
" Option 1: Sync from start of file
autocmd BufEnter * :syntax sync fromstart

" Option 2: Set sync range
syntax sync minlines=200

" Quick mapping to force syntax sync
noremap <F12> <Esc>:syntax sync fromstart<CR>
inoremap <F12> <C-o>:syntax sync fromstart<CR>
```

Example: Example (Neovim)

```
-- Option 1: Sync from start of file
vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '*',
  callback = function()
    vim.cmd('syntax sync fromstart')
  end
})

-- Option 2: Set sync range
vim.o.synmaxline = 200

-- Quick mapping to force syntax sync
```

```
vim.keymap.set('n', '<F12>', ':syntax sync fromstart<CR>', { desc = 'Resync
↪  Syntax Highlighting' })
vim.keymap.set('i', '<F12>', '<C-o>:syntax sync fromstart<CR>', { desc =
↪  'Resync Syntax Highlighting' })
```

Source: vim.fandom.com

17.169 Fix Syntax Highlighting with Sync Methods

Category: configuration
Tags: syntax, performance, highlighting

Improve syntax highlighting accuracy and performance by configuring sync methods for different file types

Example: Example (Vim)

```
" Sync syntax highlighting from start of file
autocmd BufEnter * :syntax sync fromstart

" For languages with C-style comments
autocmd FileType javascript syn sync ccomment javaScriptComment

" Set minimum lines to search backwards
syntax sync minlines=200
```

Example: Example (Neovim)

```
-- Sync syntax highlighting from start of file
vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '*',
  callback = function()
    vim.cmd('syntax sync fromstart')
  end
})

-- For languages with C-style comments
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'javascript',
  callback = function()
    vim.cmd('syn sync ccomment javaScriptComment')
  end
})

-- Set minimum lines to search backwards
vim.opt.synmaxline = 200
```

Source: vim.fandom.com

17.170 Fix Temp File Errors on Windows

Category: configuration
Tags: windows, environment, file-operations

Resolve Vim temporary file creation issues by explicitly setting the temp directory with write permissions

Example: Example (Vim)

```
let $TMP="c:/tmp"
set directory=.,$TMP,$TEMP
```

Example: Example (Neovim)

```
vim.env.TMP = "c:/tmp"
vim.opt.directory = { ".", vim.env.TMP, vim.env.TEMP }
```

Source: vim.fandom.com

17.171 Fix Temporary File Errors on Windows

Category: configuration
Tags: windows, file-operations, environment

Resolve Vim temporary file creation errors by setting a custom temp directory with write permissions

Example: Example (Vim)

```
let $TMP="c:/tmp"
set directory=.,$TMP,$TEMP
```

Example: Example (Neovim)

```
vim.env.TMP = "c:/tmp"
vim.opt.directory = { ".", vim.env.TMP, vim.env.TEMP }
```

Source: vim.fandom.com

17.172 Fix Temporary File Issues in Windows Vim

Category: configuration
Tags: windows, temp-files, environment

Configure temporary file directory to resolve file creation and read errors in Vim on Win-

dows

Example: Example (Vim)

```
let $TMP="c:/tmp"
set directory=.,$TMP,$TEMP
```

Example: Example (Neovim)

```
vim.env.TMP = "c:/tmp"
vim.opt.directory = { ".", vim.env.TMP, vim.env.TEMP }
```

Source: vim.fandom.com

17.173 Fix Terminal Arrow Key Navigation Issues

Category: configuration
Tags: terminal, key-mapping, troubleshooting

Resolve arrow key navigation problems in terminal Vim/Neovim by adjusting terminal settings

Example: Example (Vim)

```
:set term=builtin_ansi
:set timeout ttimeoutlen=100 timeoutlen=5000
```

Example: Example (Neovim)

```
-- Set terminal type to resolve key navigation issues
vim.o.term = 'builtin_ansi'

-- Adjust timeout settings for better key responsiveness
vim.o.timeout = true
vim.o.ttimeoutlen = 100
vim.o.timeoutlen = 5000
```

Source: vim.fandom.com

17.174 Fix Trackpoint Scrolling in Vim

Category: configuration
Tags: windows, input-devices, trackpoint

Modify Synaptics driver configuration to enable proper scrolling in Vim split windows

Example: Example (Vim)

```
; Add to tp4table.dat file in Synaptics driver directory
*,*,gvim.exe,*,*,*,WheelStd,0,9
```

Example: Example (Neovim)

```
-- Note: This is a driver-level configuration
-- Recommend editing tp4table.dat manually or using Synaptics driver
↪  settings
```

Source: vim.fandom.com

17.175 Fix Vim Configuration in Cygwin

Category: configuration
Tags: cygwin, setup, environment

Resolve .vimrc configuration issues in Cygwin by either storing settings in .virc or installing Vim via setup.exe

Example: Example (Vim)

```
# Option 1: Use .virc instead of .vimrc
# Option 2: Add alias to shell config
echo "alias vi=vim" >> ~/.bashrc
```

Example: Example (Neovim)

```
-- Create alias in shell config
-- In Lua, this would typically be done in shell configuration
-- Can also be managed through Neovim lua config if needed
```

Source: vim.fandom.com

17.176 Fix Vim File Association Issues

Category: configuration
Tags: windows, file-operations, troubleshooting

Resolve file association problems after Vim upgrade by updating registry keys

Example: Example (Vim)

```
# Edit registry key for Vim application
# Navigate to HKEY_CLASSES_ROOT\Applications\gvim.exe
# Ensure edit\command and open\command have correct path
```

Example: Example (Neovim)

```
-- For Neovim, use vim.fn.system() to modify registry if needed
-- Recommended to use official installers or Windows tools
-- Example registry modification:
-- vim.fn.system('reg add
↪  "HKEY_CLASSES_ROOT\\Applications\\gvim.exe\\shell\\edit\\command" /ve /d
↪  "C:\\vim\\vim82\\gvim.exe %1"')
```

Source: vim.fandom.com

17.177 Flexible CSV Delimiter Configuration

Category: configuration
Tags: csv, text-processing, configuration

Easily change the delimiter used in CSV files, supporting different data formats

Example: Example (Vim)

```
" Change CSV delimiter
:Delimiter       " show delimiter
:Delimiter ;     " set delimiter to ';'
:Delimiter \t    " set delimiter to tab
```

Example: Example (Neovim)

```
-- Create a command to manage CSV delimiter
vim.api.nvim_create_user_command('Delimiter', function(opts)
  if #opts.args == 0 then
    print('Current delimiter: ' .. (vim.g.csv_delimiter or ','))
  else
    vim.g.csv_delimiter = opts.args
    print('Delimiter set to: ' .. opts.args)
  end
end, { nargs = '?' })
```

Source: vim.fandom.com

17.178 Flexible Compiler Plugin Management

Category: configuration
Tags: compiler, plugin, language-tools

Create compiler plugins that can be easily switched for different build/validation tools within the same filetype

Example: Example (Vim)

```
" In ~/.vim/compiler/xmllint.vim
if exists('current_compiler')
  finish
endif
let current_compiler = 'xmllint'
setlocal makeprg=xmllint\ --valid\ --noout\ %
setlocal errorformat=%f:%l:\ %m

" In ~/.vim/ftplugin/xml.vim
compiler xmllint
```

Example: Example (Neovim)

```
-- In ~/.config/nvim/compiler/xmllint.lua
if vim.g.current_compiler then
  return
end

vim.g.current_compiler = 'xmllint'

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'xml',
  callback = function()
    vim.opt_local.makeprg = 'xmllint --valid --noout %'
    vim.opt_local.errorformat = '%f:%l: %m'
  end
})
```

Source: vim.fandom.com

17.179 Flexible Error Message Parsing

Category: configuration
Tags: compiler, error-handling, quickfix

Use sophisticated errorformat patterns to extract file, line, column, and error type information from various tool outputs

Example: Example (Vim)

```
" Parse CMake multi-line error messages
let &efm = ' %#%f:%l %#(%m)'
let &efm .= ',%E' . 'CMake Error at %f:%l (%m):'
let &efm .= ',%Z' . 'Call Stack (most recent call first):'
let &efm .= ',%C' . ' %m'
```

Example: Example (Neovim)

```
-- Lua equivalent for complex errorformat
vim.o.errorformat = table.concat({
  ' %#%f:%l %#(%m)',
  ',%E' .. 'CMake Error at %f:%l (%m):',
  ',%Z' .. 'Call Stack (most recent call first):',
  ',%C' .. ' %m'
}, '')
```

Source: vim.fandom.com

17.180 Flexible File Path Configuration

Category: configuration
Tags: file-operations, path-management

Configure Vim's path and filename handling to work with complex file systems and naming conventions

Example: Example (Vim)

```
" Add space to valid filename characters
set isfname+=32

" Add custom paths for file searching
set path+=C:/DOCUME~1/MYUSER~1/MYDOCU~1
```

Example: Example (Neovim)

```
-- Lua equivalents for path and filename configuration
vim.opt.isfname:append(32)  -- Add space to valid filename chars
vim.opt.path:append('C:/DOCUME~1/MYUSER~1/MYDOCU~1')
```

Source: vim.fandom.com

17.181 Flexible Indentation Strategies in Neovim

Category: configuration
Tags: indentation, formatting, settings

Configure Vim/Neovim's indentation to use spaces, tabs, or mixed modes based on project requirements

Example: Example (Vim)

```
# Indentation without tabs
set expandtab
set shiftwidth=2
```

```
set softtabstop=2

# Indentation with hard tabs
set shiftwidth=2
set tabstop=2

# Mixed tabs and spaces
set shiftwidth=2
set softtabstop=2
```

Example: Example (Neovim)

```
-- Indentation without tabs
vim.opt.expandtab = true
vim.opt.shiftwidth = 2
vim.opt.softtabstop = 2

-- Indentation with hard tabs
vim.opt.shiftwidth = 2
vim.opt.tabstop = 2

-- Mixed tabs and spaces
vim.opt.shiftwidth = 2
vim.opt.softtabstop = 2
```

Source: vim.fandom.com

17.182 Flexible LaTeX Compilation with Makefiles

Category: configuration
Tags: latex, build-system, make

Dynamically set make program for LaTeX compilation, using project-specific or generic Makefile

Example: Example (Vim)

```
setlocal errorformat=%f:%l:\ %m,%f:%l-%\d%\+:\ %m
if filereadable('Makefile')
  setlocal makeprg=make
else
  exec "setlocal makeprg=make\ -f\ ~/academic/tools/latex.mk\ " .
  ↪ substitute(bufname("%"),"tex$","pdf", "")
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'tex',
  callback = function()
```

```
    vim.opt_local.errorformat = '%f:%l: %m,%f:%l-%d+: %m'
    if vim.fn.filereadable('Makefile') == 1 then
      vim.opt_local.makeprg = 'make'
    else
      vim.opt_local.makeprg = string.format('make -f
      ↪  ~/academic/tools/latex.mk %s',
        vim.fn.expand('%:r') .. '.pdf')
    end
  end
})
```

Source: vim.fandom.com

17.183 Flexible LaTeX Compilation with Rubber and Make

Category: configuration
Tags: latex, compilation, make, project-setup

Dynamically set LaTeX compilation method based on project structure

Example: Example (Vim)

```
setlocal errorformat=%f:%l:\ %m,%f:%l-%\d%\+:\ %m
if filereadable('Makefile')
  setlocal makeprg=make
else
  exec "setlocal makeprg=make\ -f\ ~/academic/tools/latex.mk\ " .
  ↪  substitute(bufname("%"),"tex$","pdf", "")
endif
```

Example: Example (Neovim)

```
vim.api.nvim_buf_set_option(0, 'errorformat', '%f:%l: %m,%f:%l-%d+: %m')
if vim.fn.filereadable('Makefile') == 1 then
  vim.bo.makeprg = 'make'
else
  vim.bo.makeprg = string.format('make -f ~/academic/tools/latex.mk %s',
  ↪  vim.fn.expand('%:r') .. '.pdf')
end
```

Source: vim.fandom.com

17.184 Flexible Number Format Handling

Category: configuration
Tags: number-formats, settings

Configure number formats to support octal, hex, and alphabetic increments

Example: Example (Vim)

```
" Add alpha support to number formats
:set nrformats+=alpha
" View current number formats
:set nrformats?
```

Example: Example (Neovim)

```
-- Add alpha support to number formats
vim.opt.nrformats:append('alpha')
-- View current number formats
print(vim.o.nrformats)
```

Source: vim.fandom.com

17.185 Flexible Option Flag Management

Category: configuration
Tags: options, configuration, flags

Advanced methods to add, remove, or modify option flags

Example: Example (Vim)

```
" Add flag to end of option
set wildmode+=lastused

" Remove flag
set wildmode-=lastused

" Add flag to beginning
set wildmode^=full
```

Example: Example (Neovim)

```
-- Note: In Neovim, use vim.opt with += -= ^= similar to Vim
vim.opt.wildmode:append('lastused')
vim.opt.wildmode:remove('lastused')
vim.opt.wildmode:prepend('full')
```

Source: vim.fandom.com

17.186 Flexible Tab to Spaces Conversion

Category: configuration
Tags: indentation, formatting, whitespace

Configure Vim/Neovim to convert tabs to spaces with customizable settings for different

file types

Example: Example (Vim)

```
" Python-specific tab settings
autocmd FileType python set tabstop=4 shiftwidth=4 expandtab

" Makefile-specific setting
autocmd FileType make setlocal noexpandtab
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.opt.tabstop = 4
    vim.opt.shiftwidth = 4
    vim.opt.expandtab = true
  end
})

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'make',
  callback = function()
    vim.opt.expandtab = false
  end
})
```

Source: vim.fandom.com

17.187 Force File Encoding to Latin1

Category: configuration
Tags: encoding, file-handling, charset

Ensure a file is always interpreted as Latin1 encoding by adding a line with upper-ASCII bytes

Example: Example (Vim)

```
# zim: set fenc=latin1 nomod : ÷÷÷÷÷÷

" Or from command line
:setlocal fenc=latin1
```

Example: Example (Neovim)

```
-- Lua equivalent for setting file encoding
vim.opt.fileencoding = 'latin1'
```

```
-- Alternative when opening file
vim.cmd('e ++enc=latin1 file.txt')
```

Source: vim.fandom.com

17.188 Force Syntax Highlighting for Custom File Extensions

Category: configuration
Tags: syntax-highlighting, file-types, autocmd

Automatically set syntax highlighting for files with non-standard extensions using autocmd

Example: Example (Vim)

```
autocmd BufNewFile,BufRead *.meals set syntax=json
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufNewFile', 'BufRead'}, {
  pattern = '*.meals',
  callback = function()
    vim.cmd('set syntax=json')
  end
})
```

Source: vim.fandom.com

17.189 GUI-Specific Configuration

Category: configuration
Tags: configuration, gui, display

Conditionally set GUI-specific options when running gvim/neovide

Example: Example (Vim)

```
if has('gui_running')
  set guioptions-=T  " no toolbar
  colorscheme elflord
endif
```

Example: Example (Neovim)

```
if vim.g.gui_running then
  vim.opt.guioptions:remove('T')  -- no toolbar
  vim.cmd.colorscheme('elflord')
```

```
end
```

Source: vim.fandom.com

17.190 GUI-Specific Vim Configuration

Category: configuration
Tags: gui, colorscheme, ui

Demonstrates how to apply settings specifically for GUI Vim (gvim) using a conditional check

Example: Example (Vim)

```
if has('gui_running')
  set guioptions-=T  " no toolbar
  colorscheme elflord
endif
```

Example: Example (Neovim)

```
if vim.g.gui_running then
  vim.opt.guioptions:remove('T')  -- no toolbar
  vim.cmd('colorscheme elflord')
end
```

Source: vim.fandom.com

17.191 Generate Project CTags Quickly

Category: configuration
Tags: ctags, project-management, development

Quick mapping to generate ctags for the current project, enabling better code navigation and completion

Example: Example (Vim)

```
" Generate ctags for the project
map <C-F12> :!ctags -R --sort=yes --c++-kinds=+pl --fields=+iaS --extra=+q
↪  .<CR>
```

Example: Example (Neovim)

```
-- Generate ctags for the project
```

```
vim.keymap.set('n', '<C-F12>', ':!ctags -R --sort=yes --c++-kinds=+pl
↪  --fields=+iaS --extra=+q .<CR>', { desc = 'Generate project ctags' })
```

Source: vim.fandom.com

17.192 Handle Character Encoding When Switching to UTF-8

Category: configuration
Tags: encoding, internationalization, character-set

Safely handle character encoding transitions by explicitly setting script and file encodings when moving from single-byte to multi-byte character sets

Example: Example (Vim)

```
set encoding=iso-8859-1
" Your vimrc contents here
set encoding=utf-8

" Additional recommendation
set fileencodings=iso-8859-1
```

Example: Example (Neovim)

```
-- Lua equivalent for handling encoding
vim.o.encoding = 'iso-8859-1'
-- Your neovim config here
vim.o.encoding = 'utf-8'

-- Set file encodings
vim.o.fileencodings = 'iso-8859-1'
```

Source: vim.fandom.com

17.193 Handle Encoding Transitions Safely

Category: configuration
Tags: encoding, character-set, internationalization

Safely manage character encoding transitions, especially when moving from single-byte to UTF-8 encodings

Example: Example (Vim)

```
set encoding=iso-8859-1
" Your vimrc contents
set encoding=utf-8
```

Example: Example (Neovim)

```
-- Lua equivalent for managing encoding
vim.o.encoding = 'iso-8859-1'
-- Your configuration
vim.o.encoding = 'utf-8'
```

Source: vim.fandom.com

17.194 Handle Filenames with Spaces

Category: configuration
Tags: file-handling, settings

Modify isfname option to handle filenames with spaces

Example: Example (Vim)

```
" Allow spaces in filenames
set isfname+=32
```

Example: Example (Neovim)

```
-- Allow spaces in filenames
vim.opt.isfname:append(32)
```

Source: vim.fandom.com

17.195 Handle Terminal Encoding for Non-Unicode Terminals

Category: configuration
Tags: encoding, terminal, internationalization

Intelligently detect and set terminal encoding for non-UTF-8 terminals, ensuring proper character display in Vim

Example: Example (Vim)

```
if has("multi_byte")
  set encoding=utf-8
  if $TERM == "linux" || $TERM_PROGRAM == "GLterm"
    set termencoding=latin1
  endif
  if $TERM == "xterm" || $TERM == "xterm-color"
    let propv = system("xprop -id $WINDOWID -f WM_LOCALE_NAME 8s ' $0'
    ↪  -notype WM_LOCALE_NAME")
    if propv !~ "WM_LOCALE_NAME .*UTF.*8"
      set termencoding=latin1
    endif
```

```
  endif
endif
```

Example: Example (Neovim)

```
if vim.fn.has('multi_byte') == 1 then
  vim.opt.encoding = 'utf-8'
  if vim.env.TERM == 'linux' or vim.env.TERM_PROGRAM == 'GLterm' then
    vim.opt.termencoding = 'latin1'
  end
  if vim.env.TERM == 'xterm' or vim.env.TERM == 'xterm-color' then
    local propv = vim.fn.system("xprop -id $WINDOWID -f WM_LOCALE_NAME 8s '
    ↪  $0' -notype WM_LOCALE_NAME")
    if not propv:match("WM_LOCALE_NAME .*UTF.*8") then
      vim.opt.termencoding = 'latin1'
    end
  end
end
```

Source: vim.fandom.com

17.196 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: Example (Vim)

```
:set hidden        " allow unsaved buffer switching
:set nohidden      " require saving before switching (default)
" Now you can use :edit, :next, etc. without saving first
```

Source: Community contributed

17.197 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: Example (Vim)

```
" Vimscript - 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
```

Example: Example (Neovim)

```
-- Lua - Smart Home key mapping:
vim.keymap.set('n', '<Home>', function()
  local col = vim.fn.col('.')
  return col == 1 and '^' or '0'
end, { expr = true, desc = 'Smart Home' })

vim.keymap.set('i', '<Home>', function()
  local col = vim.fn.col('.')
  return col == 1 and '<C-o>^' or '<C-o>0'
end, { expr = true, desc = 'Smart Home' })

-- Alternative version using function:
local function smart_home()
  local curcol = vim.fn.col('.')
  vim.cmd('normal! ^')
  if vim.fn.col('.') == curcol then
    vim.cmd('normal! 0')
  end
end

vim.keymap.set('n', '<Home>', smart_home, { silent = true, desc = 'Smart
↪  Home' })
```

Source: Community contributed

17.198 Ignore Whitespace Differences in Diff Mode

Category: configuration
Tags: diff, whitespace, comparison

Ignore trivial whitespace changes when using Vim's diff mode, reducing visual noise

Example: Example (Vim)

```
set diffopt+=iwhite
```

Example: Example (Neovim)

```
vim.opt.diffopt:append('iwhite')
```

Source: vim.fandom.com

17.199 Improve Vim Learning and Productivity

Category: configuration
Tags: learning, productivity, workflow

Practical advice for learning Vim efficiently and expanding your skills

Example: Example (Vim)

```
# Create .vimrc
touch $HOME/.vimrc
```

Example: Example (Neovim)

```
-- Create initial Neovim config
vim.fn.mkdir(vim.fn.expand('~/.config/nvim'), 'p')
-- Tip: Learn features gradually as needed
-- Use :help to explore Vim capabilities
```

Source: vim.fandom.com

17.200 Improve XML/XSLT Attribute Completion

Category: configuration
Tags: completion, xml, filetype

Modify iskeyword setting to enable better completion for XML/XSLT attributes with hyphens and namespaces

Example: Example (Vim)

```
if has("autocmd")
  autocmd FileType {xml,xslt} setlocal
  ↪  iskeyword=@,-,\:,48-57,_,128-167,224-235
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd("FileType", {
  pattern = {"xml", "xslt"},
  callback = function()
    vim.opt_local.iskeyword:append("-,:,0-9,_")
  end
})
```

Source: vim.fandom.com

17.201 Improved Java Compilation Error Handling

Category: configuration
Tags: java, error-handling, quickfix

Configure Vim to handle Java compiler errors with a more robust error format

Example: Example (Vim)

```
set makeprg=javac\ %
set errorformat=%A:%f:%l:\ %m,%-Z%p^,%-C%.%#
```

Example: Example (Neovim)

```
vim.o.makeprg = 'javac %'
vim.o.errorformat = '%A:%f:%l: %m,%-Z%p^,%-C%.%#'

-- Optional: Auto-open quickfix window on compilation
vim.api.nvim_create_autocmd('QuickFixCmdPost', {
  pattern = '[^l]*',
  command = 'cwindow'
})
```

Source: vim.fandom.com

17.202 Increase Menu Item Limit

Category: configuration
Tags: menu-customization, ui-configuration

Increase the number of menu items before collapsing

Example: Example (Vim)

```
set menuitems=50
```

Example: Example (Neovim)

```
-- Lua equivalent
vim.o.menuitems = 50
```

Source: vim.fandom.com

17.203 Increase Redraw Time for Large Files

Category: configuration
Tags: performance, large-files

Increase redraw time to ensure syntax highlighting works on complex or large files

Example: Example (Vim)

```
" Increase redraw time
set redrawtime=10000
```

Example: Example (Neovim)

```
-- Increase redraw time
vim.opt.redrawtime = 10000
```

Source: vim.fandom.com

17.204 Install Full-Featured Vim on Linux

Category: configuration
Tags: linux, package-management, installation

Install full-featured Vim with scripting and GUI support on different Linux distributions

Example: Example (Vim)

```
# Debian/Ubuntu
# apt-get install vim-gtk

# Fedora
# yum install vim-enhanced vim-X11
```

Example: Example (Neovim)

```
-- Use distribution package manager
-- Debian/Ubuntu: vim.fn.system('apt-get install vim-gtk')
-- Fedora: vim.fn.system('yum install vim-enhanced vim-X11')
```

Source: vim.fandom.com

17.205 Install Multiple Vim Versions Simultaneously

Category: configuration
Tags: multi-version, vim-setup, environment

Strategy for installing and managing multiple Vim/Neovim versions on the same system by using PATH and version-specific directories

Example: Example (Vim)

```
# Example environment setup
set VIM=C:\PROGRA~1\vim
path %VIM%\vim63;%PATH%
```

Example: Example (Neovim)

```
-- Lua equivalent focuses on environment management
-- Use vim.env to manage environment variables
vim.env.VIM = '/usr/share/vim'
-- Configure runtime paths dynamically based on version
vim.opt.runtimepath:append(vim.env.VIM .. '/vim63')
```

Source: vim.fandom.com

17.206 Integrate PHP Manual into Vim Help

Category: configuration
Tags: documentation, php, help

Download and integrate PHP manual as Vim help files for instant function reference

Example: Example (Vim)

```
" Download PHP manual
" Put php_manual.txt in vim/doc folder
:helptags /path/to/vim/doc
```

Example: Example (Neovim)

```
-- For Neovim, add PHP doc path to runtimepath
vim.opt.runtimepath:append('path/to/php/doc')

-- Generate help tags
vim.cmd('helptags path/to/php/doc')
```

Source: vim.fandom.com

17.207 Integrate Vim as Diff Tool for TortoiseCVS

Category: configuration
Tags: version-control, diff-tool, windows

Set up Vim/GVim as an external diff tool for TortoiseCVS on Windows

Example: Example (Vim)

```
@echo off
"C:\Program Files\Vim\vim63\GVim.exe" -d %1 %2
```

Example: Example (Neovim)

```
-- For TortoiseCVS diff configuration
-- Navigate to TortoiseCVS Preferences
-- Set Diff Application to full path of nvim or gvim
-- Use parameters: -d "%1" "%2"
```

Source: vim.fandom.com

17.208 Internationalize Vim Messages and Menus

Category: configuration
Tags: internationalization, localization, translation

Comprehensive guide to translating Vim messages and menus using GNU gettext utility, allowing custom language support

Example: Example (Vim)

```
" Set language in Vim
:language el  " Set to Greek example
```

Example: Example (Neovim)

```
-- Set language in Neovim
vim.cmd('language el')  -- Set to Greek example
```

Source: vim.fandom.com

17.209 Investigate Mappings and Abbreviations

Category: configuration
Tags: mappings, debugging, configuration

Quick ways to list and investigate defined mappings, abbreviations, and their sources

Example: Example (Vim)

```
" List commands
:map              " List all mappings
:ab               " List abbreviations
:verbose map x    " List mappings starting with 'x'
:verbose ab x     " List abbreviations starting with 'x'
```

Example: Example (Neovim)

```
-- Lua equivalents for listing mappings
-- Note: Neovim provides more modern APIs for this
vim.cmd('map')  -- List mappings
vim.cmd('ab')   -- List abbreviations

-- More idiomatic Neovim approach might use
local mappings = vim.api.nvim_get_keymap('n')  -- Get normal mode mappings
```

Source: vim.fandom.com

17.210 J2ME Development Vim Configuration

Category: configuration
Tags: java, mobile-development, compiler-integration

Set up Vim mappings and configurations for J2ME mobile development, including compilation, verification, and emulation shortcuts

Example: Example (Vim)

```
set makeprg=jikes\ -nowarn\ -classpath\ tmpclasses\ -d\ tmpclasses\
↪  -sourcepath\ src\ -Xstdout\ +E\ %
set errorformat=%f:%l:%c:%*\d:%*\d:%*\s%m

map <M-1> :make<CR>:cw5<CR>
map <M-2> :!preverify -classpath
↪  c:\wtk22\lib\cldcapi10.jar;c:\wtk22\lib\midpapi10.jar -d bin
↪  tmpclasses<CR>
map <M-3> :!c:\wtk22\bin\emulator.exe -classpath bin
↪  -Xdescriptor:zhanguo.jad<CR>

cmap %/ <C-R>=expand("%:p:h")."/"<CR>
```

Example: Example (Neovim)

```
-- J2ME Development Configuration
vim.o.makeprg = 'jikes -nowarn -classpath tmpclasses -d tmpclasses
↪  -sourcepath src -Xstdout +E %'
vim.o.errorformat = '%f:%l:%c:%*%d:%*%d:%*s%m'

-- Mappings for compilation, verification, and emulation
```

```
vim.keymap.set('n', '<M-1>', ':make<CR>:cw5<CR>', { desc = 'Compile J2ME
↪  project' })
vim.keymap.set('n', '<M-2>', ':!preverify -classpath
↪  c:\wtk22\lib\cldcapi10.jar;c:\wtk22\lib\midpapi10.jar -d bin
↪  tmpclasses<CR>', { desc = 'Preverify J2ME classes' })
vim.keymap.set('n', '<M-3>', ':!c:\wtk22\bin\emulator.exe -classpath bin
↪  -Xdescriptor:zhanguo.jad<CR>', { desc = 'Run J2ME emulator' })

-- Expand current file's directory
vim.keymap.set('c', '%/', function()
  return vim.fn.expand('%:p:h') .. '/'
end, { expr = true })
```

Source: vim.fandom.com

17.211 Java Build Error Handling in Vim

Category: configuration
Tags: error-handling, build-system, java, compilation

Configure Vim to parse Java/Ant build errors and navigate to exact error locations using errorformat and custom scripts

Example: Example (Vim)

```
" Set up errorformat for Java/Ant build errors
set errorformat=
\%-G%.%#build.xml:%.%#,
\%f:%l:\ %#%m,
\%ECaused\ by:%[%^:]%#:%\=\ %\=%m

" Key mappings for error navigation
nmap <F10> :clist<CR>
nmap <F11> :cprev<CR>
nmap <F12> :cnext<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for error handling and navigation
vim.opt.errorformat = {
  '%-G%.%#build.xml:%.%#',
  '%f:%l: %#%m',
  '%ECaused by:%[%^:]%#:%=\ %m'
}

-- Key mappings for error navigation
vim.keymap.set('n', '<F10>', ':clist<CR>', { desc = 'List errors' })
vim.keymap.set('n', '<F11>', ':cprev<CR>', { desc = 'Previous error' })
vim.keymap.set('n', '<F12>', ':cnext<CR>', { desc = 'Next error' })
```

Source: vim.fandom.com

17.212 Language-Specific Compiler and Error Handling

Category: configuration
Tags: compiler, language-support, error-handling

Set custom makeprg and errorformat for different programming languages using compiler plugins, enabling language-specific build and error parsing

Example: Example (Vim)

```
" XML Lint Compiler Plugin
if exists("current_compiler")
  finish
endif
let current_compiler = "xmllint"
setlocal makeprg=xmllint\ --valid\ --noout\ %
setlocal errorformat=%f:%l:\ %m

" Filetype plugin to set default compiler
compiler xmllint
```

Example: Example (Neovim)

```
-- XML Lint Compiler Plugin
if vim.g.current_compiler then
  return
end

vim.g.current_compiler = "xmllint"
vim.opt_local.makeprg = "xmllint --valid --noout %"
vim.opt_local.errorformat = "%f:%l: %m"

-- Set default compiler for XML files
vim.cmd.compiler("xmllint")
```

Source: vim.fandom.com

17.213 Launch Vim Inside Visual Studio MDI

Category: configuration
Tags: integration, external-tools, ide

Configure gVim to launch inside Visual Studio as an internal window using the -P option, useful for developers wanting Vim editing capabilities in Visual Studio

Example: Example (Vim)

```
Command: C:\Program Files\Vim\vim71\gvim.exe
Arguments: -P "Microsoft Visual C++" --servername MDI_VIM
```

Example: Example (Neovim)

```
-- Lua configuration would typically involve setting up external tool
↪  configuration in Visual Studio
-- This is more of an IDE-specific setup than a Neovim-specific
↪  configuration
```

Source: vim.fandom.com

17.214 Lightweight Project Sessions per Tab

Category: configuration
Tags: session-management, workflow, project

Create lightweight project sessions by saving and loading individual tab page configurations

Example: Example (Vim)

```
set sessionoptions-=tabpages
:mksession sub-project.vim
:so sub-project.vim
```

Example: Example (Neovim)

```
-- Remove tabpages from session options
vim.o.sessionoptions = vim.o.sessionoptions:gsub('tabpages', '')
-- Save and load session
vim.cmd('mksession sub-project.vim')
vim.cmd('source sub-project.vim')
```

Source: vim.fandom.com

17.215 Live Word Count in Status Line

Category: configuration
Tags: status-line, word-count, ui

Add a dynamic word count to your Vim/Neovim status line, showing total and current word count

Example: Example (Vim)

```
function WordCount()
    if has_key(wordcount(),'visual_words')
        return wordcount().visual_words.'/'.wordcount().words
    else
        return wordcount().cursor_words.'/'.wordcount().words
    endif
```

```
endfunction

set statusline+=\ w:%{WordCount()},
set laststatus=2
```

Example: Example (Neovim)

```
function _G.word_count()
    local wc = vim.fn.wordcount()
    if wc.visual_words then
        return wc.visual_words .. '/' .. wc.words
    else
        return wc.cursor_words .. '/' .. wc.words
    end
end

vim.o.statusline = vim.o.statusline .. ' w:%{v:lua.word_count()}'
vim.o.laststatus = 2
```

Source: vim.fandom.com

17.216 Manage Filetype Detection Priorities

Category: configuration
Tags: filetype, runtime-path, file-detection

Understand the order of filetype detection across different runtime path locations

Example: Example (Vim)

```
" Filetype detection happens in this order:
" 1. User-specific primary definitions
" 2. System primary definitions
" 3. Vim default ruleset
" 4. System fallback definitions
" 5. User-specific fallback definitions
```

Example: Example (Neovim)

```
-- Note: Filetype detection order remains the same in Neovim
-- Use vim.api.nvim_create_autocmd() with appropriate group and priority
-- Custom rules in ~/.config/nvim/ftdetect/ or filetype.lua take precedence
```

Source: vim.fandom.com

17.217 Manage Language and Encoding Switches

Category: configuration
Tags: language-support, encoding, internationalization

Configure Vim to switch between different language encodings and menu languages

Example: Example (Vim)

```
" Toggle between languages
let $LANG='gr'
:lan mes gr
```

Example: Example (Neovim)

```
-- Toggle between languages
vim.env.LANG = 'gr'
vim.cmd('lan mes gr')
```

Source: vim.fandom.com

17.218 Manage Multiple Project Sessions

Category: configuration
Tags: session-management, project-workflow, vim-config

Automatically save and restore Vim sessions for different projects, allowing easy context switching between work environments

Example: Example (Vim)

```
let g:PathToSessions = $HOME . "/.vim/sessions/"

function! VimEnter()
  if argc() == 0
    let LoadLastSession = confirm("Restore last session?", "&Yes\n&No")
    if LoadLastSession == 1
      exe "source " . g:PathToSessions . "/LastSession.vim"
    endif
  endif
endfunction

function! VimLeave()
  exe "mksession! " . g:PathToSessions . "/LastSession.vim"
endfunction
```

Example: Example (Neovim)

```
vim.g.path_to_sessions = vim.fn.expand('$HOME') .. '/.vim/sessions/'

vim.api.nvim_create_autocmd('VimEnter', {
  callback = function()
    if vim.fn.argc() == 0 then
      local choice = vim.fn.confirm('Restore last session?', '&Yes\n&No')
      if choice == 1 then
        vim.cmd('source ' .. vim.g.path_to_sessions .. '/LastSession.vim')
      end
    end
  end
})

vim.api.nvim_create_autocmd('VimLeave', {
  callback = function()
    vim.cmd('mksession! ' .. vim.g.path_to_sessions .. '/LastSession.vim')
  end
})
```

Source: vim.fandom.com

17.219 Manage Multiple Vim Versions with Symbolic Links

Category: configuration
Tags: version-management, windows, installation

Create a symbolic link to easily switch between different Vim versions without changing system shortcuts or registry entries

Example: Example (Vim)

```
# Use linkd utility to create symbolic link
# linkd current vim70c
```

Example: Example (Neovim)

```
-- In Neovim, this is more commonly handled through package managers
-- Like lazy.nvim or Mason for version management
-- For Windows, you can use powershell to create symlinks
-- New-Item -ItemType SymbolicLink -Path 'C:\Program Files\Vim\current'
↪  -Target 'C:\Program Files\Vim\vim70c'
```

Source: vim.fandom.com

17.220 Managing Vim Source Code with Mercurial

Category: configuration
Tags: version-control, source-code, update

Use Mercurial commands to clone, update, and manage Vim source code repository

Example: Example (Vim)

```
" Clone Vim repository
hg clone https://vim.googlecode.com/hg/ vim

" Update to latest version
hg pull -u
```

Example: Example (Neovim)

```
-- Lua equivalent commands (typically run in shell)
-- git clone https://github.com/vim/vim.git
-- git pull origin master
```

Source: vim.fandom.com

17.221 Manually Add Missing X11 Fonts for Vim

Category: configuration
Tags: fonts, x11, system-config

Manually copy and configure missing X11 fonts for GTK Vim when fontconfig doesn't recognize them

Example: Example (Vim)

```
" Create custom fonts directory
" cp /usr/X11R6/lib/X11/fonts/misc/6x13{,B,O}.pcf.gz ~/.fonts
" Add ~/.fonts to ~/.fonts.conf
```

Example: Example (Neovim)

```
-- Lua equivalent: Manual font configuration
-- Steps remain similar, using system commands
-- vim.fn.mkdir(vim.fn.expand('~/.fonts'), 'p')
-- Manually copy fonts and update font configuration
```

Source: vim.fandom.com

17.222 Manually Set Syntax Highlighting

Category: configuration
Tags: syntax-highlighting, file-types

Quickly set syntax highlighting for files Vim doesn't automatically recognize

Example: Example (Vim)

```
:set syntax=php
:set syntax=perl
:set syntax=html
```

Example: Example (Neovim)

```
vim.opt.syntax = 'php'  -- Or
vim.cmd('set syntax=perl')
vim.cmd('set syntax=html')
```

Source: vim.fandom.com

17.223 Map CapsLock to Escape in X11 Systems

Category: configuration
Tags: key-mapping, productivity

Remap CapsLock key to Escape for easier mode switching in Vim/Neovim, reducing hand movement

Example: Example (Vim)

```
# Xmodmap configuration
remove Lock = Caps_Lock
keysym Caps_Lock = Escape
```

Example: Example (Neovim)

```
-- Neovim doesn't directly handle X11 remapping
-- Use xmodmap in shell:
-- xmodmap -e 'clear Lock' -e 'keycode 0x42 = Escape'
```

Source: vim.fandom.com

17.224 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: Example (Neovim)

```
-- In init.lua
vim.g.markdown_fenced_languages = {
  "html",
  "javascript",
  "typescript",
  "css",
  "scss",
  "lua",
  "vim",
  "python",
  "bash"
}
```

Source: Community contributed

17.225 Minimal Vim Configuration Testing

Category: configuration
Tags: debugging, configuration-test, minimal-setup

Use command-line flags to test Vim behavior with progressively reduced configurations

Example: Example (Vim)

```
" Test flags for minimal Vim startup
" vim -u NONE
" vim -u NORC
" gvim -U NONE  " For GUI-specific issues
```

Example: Example (Neovim)

```
-- Neovim minimal configuration testing
-- Use terminal commands:
-- nvim -u NONE      # Skip all startup
-- nvim -u NORC      # Load plugins, skip RC
-- nvim --noplugin   # Load RC, skip plugins
-- gvim -U NONE      # For GUI-specific tests
```

Source: vim.fandom.com

17.226 Modify C/C++ Comment Behavior

Category: configuration
Tags: comments, language-specific, filetype

Modify comment behavior specifically for C and C++ files to prevent automatic comment insertion on new lines

Example: Example (Vim)

```
" Modify comment behavior for C/C++ files
au FileType c,cpp setlocal comments-=:// comments+=f://
```

Example: Example (Neovim)

```
-- Modify comment behavior for C/C++ files
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'c', 'cpp'},
  callback = function()
    vim.opt_local.comments:remove(':// ')
    vim.opt_local.comments:append('f://')
  end
})
```

Source: vim.fandom.com

17.227 Modify Option Flags Dynamically

Category: configuration
Tags: options, configuration, advanced

Add or remove option flags using += and -= operators

Example: Example (Vim)

```
set listchars+=tab:>-  # Add tab display characters
set listchars-=trail:_  # Remove trail character
```

Example: Example (Neovim)

```
vim.opt.listchars:append({ tab = '>-' })
vim.opt.listchars:remove('trail:')
```

Source: vim.fandom.com

17.228 Multiple Vim Instances for Different File Types

Category: configuration
Tags: file-types, workflow, windows

Create separate Vim instances for different file type categories with unique server names

Example: Example (Vim)

```
gvim.exe --servername TXTVIM --remote-tab-silent "%1"
```

Example: Example (Neovim)

```
-- Neovim typically handles this differently through LSP and filetype
↪  plugins
-- Can configure distinct behaviors per filetype in init.lua
```

Source: vim.fandom.com

17.229 OS-Specific Configuration

Category: configuration
Tags: environment, conditional-config

Use environment variables to create conditional configurations based on user or machine

Example: Example (Vim)

```
if $USER == 'davidr'
  echo "on home pc"
  set ...
else
  echo "on work pc"
  set ...
endif
```

Example: Example (Neovim)

```
if vim.env.USER == 'davidr' then
  print('on home pc')
  -- home pc specific settings
else
  print('on work pc')
  -- work pc specific settings
end
```

Source: vim.fandom.com

17.230 Open Qt Documentation from Vim

Category: configuration
Tags: documentation, external-tools, web-integration

Quickly open Qt class or function documentation from Vim using custom function and browser integration

Example: Example (Vim)

```
function! QtClassDoc()
        let qt_dir = "/usr/share/qt4/doc/html/"
        let class = tolower(expand("<cword>"))
         silent execute "!opera -pd .opera-qt -newpage " . qt_dir . class .
          ↪  ".html &>/dev/null" . " &" | redraw!
endfunction
```

Example: Example (Neovim)

```
function _G.qt_class_doc()
        local qt_dir = "/usr/share/qt4/doc/html/"
        local class = string.lower(vim.fn.expand('<cword>'))
        vim.fn.system(string.format("opera -pd .opera-qt -newpage %s%s.html
         ↪  &>/dev/null &", qt_dir, class))
        vim.cmd.redraw()
end

-- Optional: Add a keymapping
vim.keymap.set('n', '<C-t>', _G.qt_class_doc, { desc = 'Open Qt Class
 ↪  Documentation' })
```

Source: vim.fandom.com

17.231 Optimize Java Code Indentation in Neovim

Category: configuration
Tags: java, indentation, code-style

Configure Vim/Neovim settings to improve Java code indentation and syntax highlighting

Example: Example (Vim)

```
"Java indentation and highlighting settings
set autoindent
set si
set shiftwidth=4
set cinoptions+=j1

"Syntax highlighting
let java_comment_strings=1
let java_highlight_java_lang_ids=1
let java_mark_braces_in_parens_as_errors=1
let java_highlight_all=1
let java_highlight_debug=1
let java_ignore_javadoc=1
let java_highlight_functions="style"
let java_minlines = 150
```

Example: Example (Neovim)

```
-- Java indentation and highlighting settings
vim.opt.autoindent = true
vim.opt.smartindent = true
vim.opt.shiftwidth = 4
vim.opt.cinoptions:append('j1')

-- Syntax highlighting
vim.g.java_comment_strings = 1
vim.g.java_highlight_java_lang_ids = 1
vim.g.java_mark_braces_in_parens_as_errors = 1
vim.g.java_highlight_all = 1
vim.g.java_highlight_debug = 1
vim.g.java_ignore_javadoc = 1
vim.g.java_highlight_functions = 'style'
vim.g.java_minlines = 150
```

Source: vim.fandom.com

17.232 Optimize Java Indentation and Syntax Highlighting

Category: configuration
Tags: java, syntax-highlighting, indentation

Configure Vim/Neovim settings for better Java code formatting and syntax highlighting

Example: Example (Vim)

```
" Java indentation settings
set autoindent
set si
set shiftwidth=4
set cinoptions+=j1

" Java syntax highlighting
let java_comment_strings=1
let java_highlight_java_lang_ids=1
let java_mark_braces_in_parens_as_errors=1
let java_highlight_all=1
let java_highlight_debug=1
let java_ignore_javadoc=1
let java_highlight_functions="style"
let java_minlines = 150
```

Example: Example (Neovim)

```
-- Java indentation settings
vim.opt.autoindent = true
vim.opt.shiftwidth = 4

-- Java syntax highlighting
```

```
vim.g.java_comment_strings = 1
vim.g.java_highlight_java_lang_ids = 1
vim.g.java_mark_braces_in_parens_as_errors = 1
vim.g.java_highlight_all = 1
vim.g.java_highlight_debug = 1
vim.g.java_ignore_javadoc = 1
vim.g.java_highlight_functions = 'style'
vim.g.java_minlines = 150
```

Source: vim.fandom.com

17.233 Organize Language-Specific Vim Settings

Category: configuration
Tags: file-type, organization, settings-management

Move language-specific Vim settings to separate ftplugin files to keep your vimrc clean and modular

Example: Example (Vim)

```
" In .vimrc
filetype plugin on

" In ~/.vim/ftplugin/python.vim
setlocal tabstop=4
setlocal shiftwidth=4
setlocal expandtab
```

Example: Example (Neovim)

```
-- Enable filetype plugins
vim.cmd('filetype plugin on')

-- In ~/.config/nvim/ftplugin/python.lua
vim.bo.tabstop = 4
vim.bo.shiftwidth = 4
vim.bo.expandtab = true
```

Source: vim.fandom.com

17.234 Organize Swap and Backup Files in Dedicated Directories

Category: configuration
Tags: file-management, swapfiles, backup

Configure Vim to store swap and backup files in specific directories, keeping your work-

ing directory clean and organized

Example: Example (Vim)

```
set backupdir=./.backup,.,/tmp
set directory=.,./.backup,/tmp
```

Example: Example (Neovim)

```
vim.opt.backupdir = {'./.backup', '.', '/tmp'}
vim.opt.directory = {'.', './.backup', '/tmp'}
```

Source: vim.fandom.com

17.235 Override Color Scheme Colors Dynamically

Category: configuration
Tags: color-scheme, theming, customization

Customize color scheme colors without modifying the original theme file, using an autocmd to override specific highlight groups

Example: Example (Vim)

```
augroup CustomColorOverride
  autocmd!
  autocmd ColorScheme * highlight Normal ctermbg=NONE guifg=lightgrey
  ↪ guibg=black | highlight MatchParen cterm=bold ctermfg=red ctermbg=NONE
  ↪ gui=bold guifg=red guibg=NONE
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('CustomColorOverride', { clear = true })
vim.api.nvim_create_autocmd('ColorScheme', {
  group = 'CustomColorOverride',
  callback = function()
    vim.api.nvim_set_hl(0, 'Normal', { bg = 'NONE', guifg = 'lightgrey',
    ↪ guibg = 'black' })
    vim.api.nvim_set_hl(0, 'MatchParen', { bold = true, ctermfg = 'red',
    ↪ ctermbg = 'NONE', gui = 'bold', guifg = 'red', guibg = 'NONE' })
  end
})
```

Source: vim.fandom.com

17.236 Override Terminal Color Capabilities

Category: configuration
Tags: terminal, color-configuration

Use command-line flags or terminal settings to specify 256-color terminfo entries

Example: Example (Vim)

```
" Terminal launch with 256-color support
" xterm -tn xterm-256color
" rxvt --termName rxvt-256color
```

Example: Example (Neovim)

```
-- Use terminal configuration methods to enable 256 colors
-- Command-line flags or terminal GUI settings
```

Source: vim.fandom.com

17.237 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: Example (Neovim)

```
-- 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])
})
```

Source: Community contributed

17.238 Persist Color Scheme Across Vim Sessions

Category: configuration
Tags: color-scheme, ui, customization

Automatically set a preferred color scheme when Vim starts by adding it to your vimrc configuration file

Example: Example (Vim)

```
colorscheme morning
```

Example: Example (Neovim)

```
vim.cmd('colorscheme morning')
```

Source: vim.fandom.com

17.239 Persist Vim Window Size and Position Across Sessions

Category: configuration
Tags: gui, window-management, persistence

Automatically save and restore Vim window size, position, and layout between sessions, similar to other desktop applications

Example: Example (Vim)

```
" Configuration options
let g:screen_size_restore_pos = 1
let g:screen_size_by_vim_instance = 1

" Autocmds to save and restore window state
autocmd VimEnter * if g:screen_size_restore_pos == 1 | call ScreenRestore()
 ↪  | endif
autocmd VimLeavePre * if g:screen_size_restore_pos == 1 | call ScreenSave()
 ↪  | endif
```

Example: Example (Neovim)

```
-- Configuration options
vim.g.screen_size_restore_pos = 1
vim.g.screen_size_by_vim_instance = 1

-- Lua equivalent would use vim.api.nvim_create_autocmd
vim.api.nvim_create_autocmd('VimEnter', {
  callback = function()
    if vim.g.screen_size_restore_pos == 1 then
      -- Implement ScreenRestore logic in Lua
```

```
    end
  end
})

vim.api.nvim_create_autocmd('VimLeavePre', {
  callback = function()
    if vim.g.screen_size_restore_pos == 1 then
      -- Implement ScreenSave logic in Lua
    end
  end
})
```

Source: vim.fandom.com

17.240 Preserve Hard Links When Editing Files

Category: configuration
Tags: file-operations, backup, system-config

Automatically break hard links when editing files, creating a backup to prevent unintended modifications to linked files

Example: Example (Vim)

```
set backupcopy=auto,breakhardlink
```

Example: Example (Neovim)

```
vim.opt.backupcopy = 'auto,breakhardlink'
```

Source: vim.fandom.com

17.241 Preserve Indent When Typing # in Insert Mode

Category: configuration
Tags: indentation, insert-mode, configuration

Prevent Vim from removing indentation when typing # at the start of a line, which can disrupt code formatting

Example: Example (Vim)

```
set cindent
set cinkeys-=0#
set indentkeys-=0#

" Optional mapping for smartindent
:inoremap # X<BS>#
```

Example: Example (Neovim)

```
vim.opt.cindent = true
vim.cmd('set cinkeys-=0#')
vim.cmd('set indentkeys-=0#')

-- Optional mapping to preserve indent with smartindent
vim.keymap.set('i', '#', 'X<BS>#', { desc = 'Preserve indent when typing #'
↪  })
```

Source: vim.fandom.com

17.242 Prevent Multiple Plugin Loads

Category: configuration
Tags: plugin-development, best-practices, performance

Use a guard to prevent reloading plugins multiple times and avoid potential errors

Example: Example (Vim)

```
if exists("loaded_myplugin")
  finish
endif
let loaded_myplugin = 1
```

Example: Example (Neovim)

```
if vim.g.loaded_myplugin then
  return
end
vim.g.loaded_myplugin = true
```

Source: vim.fandom.com

17.243 Prevent Multiple Vim Instances

Category: configuration
Tags: instance-management, startup, safety

Check if another Vim instance is already running to prevent potential data conflicts or duplicate windows

Example: Example (Vim)

```
" Check for existing Vim server instances
if serverlist() =~? "\n."
  echo "MyWarning: Another copy of gvim or Vim is probably loaded!"
endif
```

Example: Example (Neovim)

```
-- Check for existing Vim server instances
if #vim.fn.serverlist() > 1 then
  print("Warning: Another Vim instance is already running!")
end
```

Source: vim.fandom.com

17.244 Prevent Vim GUI from Forking Processes

Category: configuration
Tags: gui, process-management

Ensure Vim GUI doesn't fork new processes, which can be important for external command integration

Example: Example (Vim)

```
set guioptions +=f
```

Example: Example (Neovim)

```
vim.opt.guioptions:append('f')
```

Source: vim.fandom.com

17.245 Print Files Using KDE Print Dialog

Category: configuration
Tags: printing, external-integration, system-command

Configure Vim to use KDE's kprinter for printing, which provides a consistent print dialog

Example: Example (Vim)

```
set printexpr=system('kprinter '\.'\
↪  '\.v:fname_in)\.delete(v:fname_in)\.+\.v:shell_error
```

Example: Example (Neovim)

```
vim.o.printexpr = 'system("kprinter " . v:fname_in) . delete(v:fname_in) +
↪  v:shell_error'
```

Source: vim.fandom.com

17.246 Profile Vim Startup Time

Category: configuration
Tags: performance, startup, debugging

Generate a CSV file listing scripts loaded during Vim startup to optimize launch time

Example: Example (Vim)

```
" Use startup_profile plugin
:StartupProfile
```

Example: Example (Neovim)

```
-- Alternative built-in method
vim.cmd('vim --startuptime startup_time.log')
```

Source: vim.fandom.com

17.247 Quick Access to Vimrc File

Category: configuration
Tags: vimrc, editing, configuration

Easily open and edit your Vim configuration file using environment variables

Example: Example (Vim)

```
" Open vimrc directly
:e $MYVIMRC
:e $MYGVIMRC
```

Example: Example (Neovim)

```
-- Open vimrc in Neovim
vim.keymap.set('n', '<leader>ve', function()
  vim.cmd.edit(vim.fn.expand('$MYVIMRC'))
end, { desc = 'Edit Neovim config' })

-- Optional: Automatically source vimrc after saving
vim.api.nvim_create_autocmd('BufWritePost', {
  pattern = vim.fn.expand('$MYVIMRC'),
  callback = function()
    vim.cmd.source(vim.fn.expand('$MYVIMRC'))
  end
})
```

Source: vim.fandom.com

17.248 Quick CapsLock to Escape Command

Category: configuration
Tags: key-mapping, productivity

One-line command to remap CapsLock to Escape using xmodmap

Example: Example (Vim)

```
# Shell command
xmodmap -e 'clear Lock' -e 'keycode 0x42 = Escape'
```

Example: Example (Neovim)

```
-- Shell command remains the same
-- Neovim integration via vim.fn.system()
vim.fn.system('xmodmap -e "clear Lock" -e "keycode 0x42 = Escape"')
```

Source: vim.fandom.com

17.249 Quick Java Compilation in Vim/Neovim

Category: configuration
Tags: compilation, java, quickfix

Configure Vim/Neovim to compile Java files quickly and navigate compilation errors using quickfix list

Example: Example (Vim)

```
set makeprg=javac\ %
set errorformat=%A%f:%l:\ %m,%-Z%p^,%-C%.%#

" Mapping to compile
map <F9> :make <CR>
```

Example: Example (Neovim)

```
-- Java compilation configuration
vim.opt.makeprg = 'javac %'
vim.opt.errorformat = '%A%f:%l: %m,%-Z%p^,%-C%.%#'

-- Mapping to compile
vim.keymap.set('n', '<F9>', ':make<CR>', { desc = 'Compile Java file' })
```

Source: vim.fandom.com

17.250 Quick PHP Function Documentation Lookup

Category: configuration
Tags: documentation, php, online-help, custom-mapping

Configure Vim to quickly look up PHP function documentation using a custom script that fetches online manual pages

Example: Example (Vim)

```
set keywordprg=~/.vim/php_doc

" In ~/.vim/php_doc script
#!/bin/bash
FUNCTION=`echo $1 | sed 's/_/-/g'`
lynx -dump -nolist http://www.php.net/manual/en/print/function.$FUNCTION.php
 ↪  | sed -n /^$1/,/^.*User\ Contributed\ Notes/p | grep -v 'User\
 ↪  Contributed\ Notes'
```

Example: Example (Neovim)

```
-- Configure PHP function documentation lookup
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'php',
  callback = function()
    vim.opt.keywordprg = '~/.vim/php_doc'
  end
})

-- Create a Lua function for more flexible documentation lookup
local function php_doc_lookup(keyword)
  local formatted_keyword = keyword:gsub('_', '-')
  local cmd = string.format(
    'lynx -dump -nolist http://www.php.net/manual/en/print/function.%s.php',
    formatted_keyword
  )
  -- You can customize how to display the output, e.g., in a new buffer
  vim.cmd('new')
  vim.fn.termopen(cmd)
end

-- Optional: Add a mapping
vim.keymap.set('n', '<leader>ph', function()
  local word = vim.fn.expand('<cword>')
  php_doc_lookup(word)
end, { desc = 'Look up PHP function documentation' })
```

Source: vim.fandom.com

17.251 Quick Spelling Correction Setup

Category: configuration
Tags: spell-checking, dictionary

Enable built-in spell checking and dictionary support

Example: Example (Vim)

```
set dict=/usr/dict/words
set spell
```

Example: Example (Neovim)

```
vim.opt.dictionary = '/usr/dict/words'
vim.opt.spell = true
```

Source: vim.fandom.com

17.252 Quick Vim Configuration Editing and Reloading

Category: configuration
Tags: vimrc, configuration, key-mapping

Easily edit and reload your Vim configuration file with custom key mappings

Example: Example (Vim)

```
" source $MYVIMRC reloads the saved $MYVIMRC
:nmap <Leader>s :source $MYVIMRC

" opens $MYVIMRC for editing
:nmap <Leader>v :e $MYVIMRC
```

Example: Example (Neovim)

```
-- Lua equivalent for configuration editing and reloading
vim.keymap.set('n', '<leader>s', ':source $MYVIMRC<CR>', { desc = 'Reload
↪  Vim configuration' })
vim.keymap.set('n', '<leader>v', ':e $MYVIMRC<CR>', { desc = 'Edit Vim
↪  configuration' })
```

Source: vim.fandom.com

17.253 Quick Vim Configuration Management

Category: configuration
Tags: vimrc, productivity, configuration-management

Quickly open and reload your Vim configuration file with leader key mappings

Example: Example (Vim)

```
" source $MYVIMRC reloads the saved $MYVIMRC
:nmap <Leader>s :source $MYVIMRC

" opens $MYVIMRC for editing
:nmap <Leader>v :e $MYVIMRC
```

Example: Example (Neovim)

```
-- Reload Vim configuration
vim.keymap.set('n', '<leader>s', ':source $MYVIMRC<CR>', { desc = 'Reload
↪  Vim config' })

-- Edit Vim configuration
vim.keymap.set('n', '<leader>v', ':e $MYVIMRC<CR>', { desc = 'Edit Vim
↪  config' })
```

Source: vim.fandom.com

17.254 Quickly Disable Syntax Highlighting During Editing

Category: configuration
Tags: syntax, display

Temporary method to turn off syntax highlighting during an editing session

Example: Example (Vim)

```
:set syntax=
```

Example: Example (Neovim)

```
vim.cmd('set syntax=')
```

Source: vim.fandom.com

17.255 Reduce 'Hit ENTER' Prompts in Vim/Neovim

Category: configuration
Tags: ui, user-experience, command-line

Minimize intrusive 'Hit ENTER to continue' messages by configuring shortmess and cmdheight options

Example: Example (Vim)

```
set shortmess=a
set cmdheight=2
```

Example: Example (Neovim)

```
-- Reduce message verbosity and increase command line height
vim.opt.shortmess:append('a')
vim.opt.cmdheight = 2
```

Source: vim.fandom.com

17.256 Remap Caps Lock to Escape or Ctrl

Category: configuration
Tags: key-mapping, productivity

Remapping Caps Lock to Escape or Ctrl can significantly improve typing efficiency and reduce hand strain in Vim

Example: Example (Vim)

```
" No direct Vimscript mapping, this is an OS-level configuration
```

Example: Example (Neovim)

```
-- Use AutoHotkey or system-level key remapping
-- Neovim specific key mapping example:
vim.keymap.set('n', '<CapsLock>', '<Esc>', { noremap = true, silent = true
↪ })
```

Source: vim.fandom.com

17.257 Remap CapsLock to Escape System-Wide

Category: configuration
Tags: key-mapping, productivity, system-config

Remap CapsLock key to Escape system-wide, which is particularly useful for Vim/Neovim users to reduce hand movement when switching modes

Example: Example (Neovim)

```
-- Recommended X11/Linux system-wide mapping
-- Run in terminal:
xmodmap -e 'clear Lock' -e 'keycode 0x42 = Escape'
```

Source: vim.fandom.com

17.258 Remap CapsLock to Escape/Ctrl Systemwide

Category: configuration
Tags: key-mapping, productivity, workflow

Remap CapsLock key to Escape or Ctrl to improve keyboard ergonomics and efficiency in Vim and other applications

Example: Example (Vim)

```
" No direct Vim configuration, system-level mapping
```

Example: Example (Neovim)

```
-- Recommended Neovim-specific alternative using AutoHotkey/system utility
-- Lua mapping example for application-specific remapping
vim.keymap.set('n', '<CapsLock>', '<Esc>', { desc = 'Use CapsLock as Escape'
↪  })
```

Source: vim.fandom.com

17.259 Remove '=' from Filename Characters

Category: configuration
Tags: filename-completion, shell-scripts, file-editing

Modify filename character set to enable better filename completion in shell scripts, especially when dealing with environment variables

Example: Example (Vim)

```
set isfname-==
```

Example: Example (Neovim)

```
vim.opt.isfname:remove('=')
```

Source: vim.fandom.com

17.260 Remove Specific GUI Elements

Category: configuration
Tags: ui, startup

Remove menu bar, toolbar, and scrollbars to maximize text editing space

Example: Example (Vim)

```
set guioptions-=m  "remove menu bar
set guioptions-=T  "remove toolbar
set guioptions-=r  "remove right-hand scroll bar
set guioptions-=L  "remove left-hand scroll bar
```

Example: Example (Neovim)

```
vim.opt.guioptions:remove('m')  -- remove menu bar
vim.opt.guioptions:remove('T')  -- remove toolbar
vim.opt.guioptions:remove('r')  -- remove right-hand scroll bar
vim.opt.guioptions:remove('L')  -- remove left-hand scroll bar
```

Source: vim.fandom.com

17.261 Remove from option value

Category: Configuration
Tags: set, option, remove

Use `:set option-=value` to remove a value from an option.

Example: Example (Vim)

```
:set path-=./include  " remove from search path
:set wildignore-=*.pyc  " stop ignoring Python bytecode
```

Source: Community contributed

17.262 Resolve Remote SSH Vim Key Mapping Issues

Category: configuration
Tags: ssh, remote-editing, compatibility

Disable Vi compatibility mode to fix broken key mappings in remote SSH sessions

Example: Example (Vim)

```
:set nocompatible
```

Example: Example (Neovim)

```
-- Disable Vi compatibility mode
vim.o.compatible = false
```

Source: vim.fandom.com

17.263 Restore Cursor Position Across Editing Sessions

Category: configuration
Tags: cursor-position, persistence, viminfo

Automatically restore the cursor to its previous position when reopening a file across multiple editing sessions

Example: Example (Vim)

```
" Configure viminfo to remember cursor position
set viminfo='10,"100,:20,%,n~/.viminfo

function! ResCur()
  if line("'\"") <= line("$")
    normal! g`"
    return 1
  endif
endfunction

augroup resCur
  autocmd!
  autocmd BufWinEnter * call ResCur()
augroup END
```

Example: Example (Neovim)

```
-- Restore cursor position configuration
vim.opt.viminfo = "'10,\"100,:20,%,n~/.viminfo"

local function restore_cursor()
  local last_position = vim.fn.line("'\"")
  local last_line = vim.fn.line("$")

  if last_position > 0 and last_position <= last_line then
    vim.cmd('normal! g`"')
  end
end

vim.api.nvim_create_augroup("RestoreCursor", { clear = true })
```

```
vim.api.nvim_create_autocmd("BufWinEnter", {
  group = "RestoreCursor",
  callback = restore_cursor
})
```

Source: vim.fandom.com

17.264 Safely Customize Syntax Files

Category: configuration
Tags: syntax, customization, best-practices

Create custom syntax extensions in the 'after' runtime directory to prevent losing changes during upgrades

Example: Example (Vim)

```
" Place custom syntax extensions in after/syntax/
" e.g., after/syntax/c.vim
```

Example: Example (Neovim)

```
-- Use vim.fn.stdpath('config') .. '/after/syntax/'
-- Create language-specific syntax extension files
```

Source: vim.fandom.com

17.265 Safely Manage Plugin Configurations

Category: configuration
Tags: plugin-management, vimscript, best-practices

Best practices for creating Vim/Neovim plugins, including preserving user settings and providing reload mechanisms

Example: Example (Vim)

```
" Save and restore 'cpo' settings
let s:cpo_save = &cpo
set cpo&vim

" Reload plugin mapping
nnoremap <S-F12> :so $VIMRUNTIME/plugins/MyPlugin.vim

" Restore original settings
let &cpo = s:cpo_save
unlet s:cpo_save
```

Example: Example (Neovim)

```
-- Lua equivalent for safe plugin configuration
local original_cpo = vim.o.cpo
vim.o.cpo = vim.o.cpo:gsub(',', '')

-- Option to reload plugin
vim.keymap.set('n', '<S-F12>', function()
  vim.cmd('source ' .. vim.fn.expand('$VIMRUNTIME/plugins/MyPlugin.lua'))
end)

-- Restore original settings
vim.o.cpo = original_cpo
```

Source: vim.fandom.com

17.266 Safely Source and Modify Vimrc

Category: configuration
Tags: vimrc, configuration, startup

Safely redefine commands, functions, and autocmds when sourcing vimrc to prevent duplicate or conflicting configurations

Example: Example (Vim)

```
" Redefine commands and functions safely
command! Mycommand echo "Hello!"
function! Myfunc()
  echo "Hello!"
endfunction

" Clear and recreate autocmd groups
augroup vimrc_autocmds
  au!
  autocmd BufRead * echo "File read!"
augroup END
```

Example: Example (Neovim)

```
-- Lua equivalent for safely managing configurations
-- Use vim.api.nvim_create_user_command for commands
vim.api.nvim_create_user_command('Mycommand', function()
  print("Hello!")
end, {})

-- Functions can be defined normally
function _G.Myfunc()
  print("Hello!")
end

-- Create autocmd group with clear mechanism
```

```
local augroup = vim.api.nvim_create_augroup('vimrc_autocmds', { clear = true
↪  })
vim.api.nvim_create_autocmd('BufRead', {
  group = augroup,
  callback = function()
    print("File read!")
  end
})
```

Source: vim.fandom.com

17.267 Safely Store and Manage Key Mappings

Category: configuration
Tags: key-mapping, vimrc, configuration

Store key mappings in configuration files to persist across Vim sessions, with options for global or filetype-specific mappings

Example: Example (Vim)

```
" Add mappings to .vimrc
map <F2> :echo 'Current time is ' . strftime('%c')<CR>
```

Example: Example (Neovim)

```
-- Add mappings to init.lua
vim.keymap.set('n', '<F2>', function()
  print(string.format('Current time is %s', os.date()))
end, { desc = 'Display current time' })
```

Source: vim.fandom.com

17.268 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: Example (Vim)

```
:sandbox set number            " test setting without permanent change
:sandbox echo expand('%')      " safely test expressions
:sandbox source unsafe.vim     " test configuration safely
```

Source: Community contributed

17.269 Save Macros Persistently

Category: configuration
Tags: macro, persistence, vimrc

Permanently save macros in your Neovim configuration for reuse across sessions

Example: Example (Vim)

```
" Save macro to register in vimrc
let @a='0fa'
```

Example: Example (Neovim)

```
-- Save macro persistently
vim.g.saved_macro_a = '0fa'

-- To restore:
vim.fn.setreg('a', vim.g.saved_macro_a)
```

Source: vim.fandom.com

17.270 Save and Restore Macros Across Sessions

Category: configuration
Tags: macro-persistence, vim-configuration, registers

Automatically save macros between Vim sessions using viminfo or manually via vimrc

Example: Example (Vim)

```
let @a='0fa'  # Manually save macro to register a
```

Example: Example (Neovim)

```
-- Save macro persistently
vim.g.saved_macro_a = '0fa'  -- Store in global variable
-- Retrieve: vim.fn.setreg('a', vim.g.saved_macro_a)
```

Source: vim.fandom.com

17.271 Secure Password File with Encryption and Folding

Category: configuration
Tags: security, encryption, privacy, folding

Create an encrypted password file with automatic folding to hide sensitive information

and prevent accidental exposure

Example: Example (Vim)

```
" Modeline for secure password file
vi: noswapfile bufhidden=wipe tw=0 fdm=expr
↪  foldexpr=getline(v:lnum)=~'^\s\|^$'
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPre', {
  pattern = '*',
  callback = function()
    if vim.fn.system('head -c 9 ' .. vim.fn.expand('<afile>')) ==
    ↪  'VimCrypt~' then
      vim.opt_local.swapfile = false
      vim.opt.viminfo = ''
      vim.opt.foldmethod = 'indent'
      vim.opt.foldlevel = 0
      vim.opt.foldclose = 'all'
      vim.opt.foldopen = ''
    end
  end
})
```

Source: vim.fandom.com

17.272 Secure Vim Configuration for Encrypted Files

Category: configuration
Tags: security, encryption, file-management

Create a secure Vim configuration to minimize trace data when working with sensitive files

Example: Example (Vim)

```
set secure
set viminfo=
set noswapfile
set nobackup
set nowritebackup
set history=0
set noshelltemp
```

Example: Example (Neovim)

```
vim.opt.secure = true
vim.opt.swapfile = false
vim.opt.backup = false
```

```
vim.opt.writebackup = false
vim.opt.history = 0
vim.opt.shelltemp = false
```

Source: vim.fandom.com

17.273 Set Custom VIMRUNTIME Path in Config

Category: configuration
Tags: runtime-path, environment, configuration

Allows setting a custom VIMRUNTIME path within your configuration, useful for managing multiple Vim/Neovim installations

Example: Example (Vim)

```
let $VIMRUNTIME='C:/usr/share/vim/vim63'
```

Example: Example (Neovim)

```
vim.env.VIMRUNTIME = 'C:/usr/share/vim/vim63'
```

Source: vim.fandom.com

17.274 Set Fortran Source Form for .for Files

Category: configuration
Tags: filetype, syntax-highlighting, fortran

Manually set Fortran source form for .for files to fix syntax highlighting issues

Example: Example (Vim)

```
au BufRead,BufNewFile *.for let b:fortran_fixed_source=1
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = '*.for',
  callback = function()
    vim.b.fortran_fixed_source = 1
  end
})
```

Source: vim.fandom.com

17.275 Set Initial Vim Window Size

Category: configuration
Tags: window-management, ui, startup

Configure initial Vim/gVim window size dynamically, with support for GUI and console modes

Example: Example (Vim)

```
if has("gui_running")
  set lines=999 columns=999
else
  if exists("+lines")
    set lines=50
  endif
  if exists("+columns")
    set columns=100
  endif
endif
```

Example: Example (Neovim)

```
if vim.g.neovide or vim.g.gnvim then
  vim.o.lines = 999
  vim.o.columns = 999
else
  vim.o.lines = 50
  vim.o.columns = 100
end
```

Source: vim.fandom.com

17.276 Set Terminal Type for Color Support

Category: configuration
Tags: terminal, color-scheme

Use environment variable to enable color support in Vim on Solaris and similar systems

Example: Example (Vim)

```
set term=sun-color
syntax on
```

Example: Example (Neovim)

```
-- Set terminal type for color support
vim.o.term = "sun-color"
```

```
vim.cmd.syntax("on")
```

Source: vim.fandom.com

17.277 Set Terminal Type for Color Vim

Category: configuration
Tags: terminal, color-support, environment

Configure terminal type to enable color display in Vim, specifically for HP-UX and Putty environments

Example: Example (Vim)

```
export TERM=dtterm
syntax on
```

Example: Example (Neovim)

```
-- Set terminal type in shell profile
vim.env.TERM = 'dtterm'

-- Enable syntax highlighting
vim.cmd('syntax on')
```

Source: vim.fandom.com

17.278 Set Vim Language and Encoding for Internationalization

Category: configuration
Tags: internationalization, localization, language-support

Configure Vim to use a specific language for messages and menus, useful for non-English users

Example: Example (Vim)

```
" Set language for messages
let $LANG='el' / let $LANG='gr'
:lan mes el / :lan mes gr

" Set menu language
:menut English Greek
let menut=Greek
```

Example: Example (Neovim)

```
-- Set language for messages
vim.env.LANG = 'el' -- or 'gr'
vim.cmd('lan mes el') -- set message language

-- Set menu language (if supported)
vim.cmd('menut English Greek')
```

Source: vim.fandom.com

17.279 Set Vim as Default Editor for Unknown File Types

Category: configuration
Tags: windows, file-operations, registry

Modify Windows registry to make Vim the default editor for files with unregistered extensions

Example: Example (Vim)

```
"REGEDIT4
[HKEY_CLASSES_ROOT\Unknown\shell\Open\Command]
@="d:\\program files\\vim\\vim60\\gvim.exe \"%1\""
```

Example: Example (Neovim)

```
-- For Neovim on Windows, use external registry edit
-- Recommend using a .reg file or PowerShell script
-- Note: Requires caution and system administrator privileges
```

Source: vim.fandom.com

17.280 Set Vim as Default PostgreSQL Editor

Category: configuration
Tags: environment, external-tool

Configure psql to use Vim as the default editor with optional SQL syntax highlighting

Example: Example (Vim)

```
export PSQL_EDITOR=vim
# For older psql versions with syntax highlighting
export PSQL_EDITOR='vim +"set syntax=sql"'
```

Example: Example (Neovim)

```
-- This is an environment variable setting, not directly a Lua/Neovim
↪  configuration
-- Set in shell or .bashrc/.zshrc
```

Source: vim.fandom.com

17.281 Shorten Long Messages in Vim

Category: configuration
Tags: messages, display, scripting

Truncate long messages in the middle to fit the line, adding '...' when necessary

Example: Example (Vim)

```
set shortmess+=T

function! ShortEcho(msg)
  let saved=&shortmess
  set shortmess+=T
  exe "norm :echomsg a:msg\n"
  let &shortmess=saved
endfunction
```

Example: Example (Neovim)

```
-- Add T flag to shortmess option
vim.opt.shortmess:append('T')

-- Function to echo shortened messages
function _G.short_echo(msg)
  local saved_shortmess = vim.opt.shortmess:get()
  vim.opt.shortmess:append('T')
  vim.cmd('norm :echomsg ' .. msg .. '\n')
  vim.opt.shortmess = saved_shortmess
end
```

Source: vim.fandom.com

17.282 Show File Encoding in Status Line

Category: configuration
Tags: status-line, encoding, file-info

Customize the status line to display file encoding and BOM (byte order mark) status, providing additional context about the current file

Example: Example (Vim)

```
if has("statusline")
 set statusline=%<%f\ %h%m%r%=%{"["."(&fenc==""?&enc:&fenc).((exists("+bomb")
 ↪  && &bomb)?",B":"")."]"}\ %k\ %-14.(%l,%c%V%)\ %P
endif
```

Example: Example (Neovim)

```
vim.opt.statusline = '%<%f %h%m%r%=%{"[" .. (vim.bo.fileencoding == "" and
↪  vim.o.encoding or vim.bo.fileencoding) .. (vim.o.bomb and ",B" or "") ..
↪  "]"} %k %-14.(%l,%c%V%) %P'
```

Source: vim.fandom.com

17.283 Show Longest Common Completion Text

Category: configuration
Tags: completion, settings

Configure completion to insert the longest common text of match candidates

Example: Example (Vim)

```
set completeopt+=longest
```

Example: Example (Neovim)

```
vim.opt.completeopt:append('longest')
```

Source: vim.fandom.com

17.284 Show Matching Braces Instantly

Category: configuration
Tags: ui, editing, visual-feedback

Briefly highlight matching braces when typing or cursor moves over them

Example: Example (Vim)

```
set showmatch
set matchtime=3  " Duration of highlight in tenths of a second"
```

Example: Example (Neovim)

```
vim.opt.showmatch = true
vim.opt.matchtime = 3
```

Source: vim.fandom.com

17.285 Smart Language-Specific Line Commenting

Category: configuration
Tags: commenting, filetype, customization

Dynamically set comment characters based on file type, allowing easy commenting across different programming languages

Example: Example (Vim)

```
function CommentLines()
  exe ":s@^@".g:Comment."@g"
  exe ":s@$@".g:EndComment."@g"
endfunction

" In filetype.vim
au BufRead,BufNewFile *.sh let Comment="#" | let EndComment=""
au BufRead,BufNewFile *.c let Comment="/*" | let EndComment="*/"

vmap co :call CommentLines()<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({"BufRead", "BufNewFile"}, {
  pattern = "*.sh",
  callback = function()
    vim.g.Comment = "#"
    vim.g.EndComment = ""
  end
})

vim.api.nvim_create_autocmd({"BufRead", "BufNewFile"}, {
  pattern = "*.c",
  callback = function()
    vim.g.Comment = "/*"
    vim.g.EndComment = "*/"
  end
})

vim.keymap.set('v', 'co', function()
  local start_line = vim.fn.line("'<")
  local end_line = vim.fn.line("'>")

  for line = start_line, end_line do
    local current_line = vim.fn.getline(line)
```

```
    local commented_line = vim.g.Comment .. current_line .. vim.g.EndComment
    vim.fn.setline(line, commented_line)
  end
end, { desc = "Comment lines based on filetype" })
```

Source: vim.fandom.com

17.286 Specify Custom SCP Port

Category: configuration
Tags: ssh, remote-editing, networking

Change the default SCP port dynamically within Vim session

Example: Example (Vim)

```
let g:netrw_scp_cmd="scp -q -P <desired_new_port>"
```

Example: Example (Neovim)

```
-- Set custom SCP port
vim.g.netrw_scp_cmd = 'scp -q -P 2222'
```

Source: vim.fandom.com

17.287 Speed Up Vim Startup in Remote Sessions

Category: configuration
Tags: performance, remote-session, startup

Reduce Vim startup time in remote/telnet sessions by disabling X server connection and plugin loading

Example: Example (Vim)

```
vim -X --noplugin
```

Example: Example (Neovim)

```
-- Use command line arguments to speed up Vim startup
-- vim -X --noplugin
-- Equivalent Lua configuration would be to modify startup settings in
↪  init.lua
vim.g.loaded_netrw = 1       -- Disable network plugin
vim.g.loaded_netrwPlugin = 1 -- Disable network plugin
-- Additional performance optimization can be done in Neovim init config
```

Source: vim.fandom.com

17.288 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: Example (Vim)

```
:noautocmd vimgrep /pattern/ **/*.txt  " faster vimgrep
:noautocmd bufdo %s/old/new/ge         " faster buffer operations
```

Source: Community contributed

17.289 Swap CapsLock and Escape Keys

Category: configuration
Tags: key-mapping, system-config

Completely swap the CapsLock and Escape keys using Xmodmap configuration

Example: Example (Neovim)

```
-- Create .speedswapper file with:
-- remove Lock = Caps_Lock
-- keysym Escape = Caps_Lock
-- keysym Caps_Lock = Escape
-- add Lock = Caps_Lock

-- Apply with:
-- xmodmap ~/.speedswapper
```

Source: vim.fandom.com

17.290 Systematic Vim Debugging via Bisection

Category: configuration
Tags: debugging, troubleshooting, configuration-management

A systematic method to isolate configuration or plugin issues by progressively eliminating potential problem sources

Example: Example (Vim)

```
" Debug steps:
" 1. Start with minimal configuration
" vim -u NONE
" vim -u NORC
" vim --noplugin

" Gradually restore configuration sections
```

Example: Example (Neovim)

```
-- Debugging commands in Neovim terminal
-- vim -u NONE    # Skip all startup
-- vim -u NORC    # Load plugins, skip RC
-- vim --noplugin # Load RC, skip plugins

-- Systematic bisection process:
-- 1. Move config files to /tmp
-- 2. Restore incrementally
-- 3. Test after each restoration
```

Source: vim.fandom.com

17.291 Terminal-Specific Cursor Shape and Color

Category: configuration
Tags: terminal, cursor-shape, escape-sequences

Set dynamic cursor shape and color for different terminals, particularly for xterm and tmux

Example: Example (Vim)

```
if &term =~ "xterm\|rxvt"
  let &t_SI = "\<Esc>]12;orange\x7"
  let &t_EI = "\<Esc>]12;red\x7"
endif
```

Example: Example (Neovim)

```
-- Terminal-specific cursor configuration
if vim.env.TERM:match("xterm") or vim.env.TERM:match("rxvt") then
  vim.opt.t_SI = "\x1b]12;orange\x7"
  vim.opt.t_EI = "\x1b]12;red\x7"
end
```

Source: vim.fandom.com

17.292 Toggle Boolean Options Easily

Category: configuration
Tags: options, toggle, settings

Quickly toggle boolean Vim options with simple commands, useful for settings like line numbers, mouse support, etc.

Example: Example (Vim)

```
" Toggle line numbers
:set number!
:set invnumber

" Check current option value
:set number?
```

Example: Example (Neovim)

```
-- Toggle line numbers
vim.opt.number = not vim.opt.number:get()

-- Check current option value
print(vim.opt.number:get())
```

Source: vim.fandom.com

17.293 Toggle Caps Lock in Vim with Custom Functions

Category: configuration
Tags: key-mapping, input-mode, productivity

Create custom commands to toggle between uppercase and mixed case input modes in Vim, with visual indicators

Example: Example (Vim)

```
command -nargs=0 Caps :call s:Caps()
command -nargs=0 Mixed :call s:Mixed()

function s:Caps()
  im a A
  im b B
  im c C
  im z Z
  echo "CAPS ON"
  hi LineNr term=underline ctermfg=3 guifg=Red3 guibg=#cccccc
endfunction

function s:Mixed()
```

```
  im a a
  im b b
  im c c
  im z z
  echo "Caps Off"
  hi LineNr term=underline ctermfg=3 guifg=Red3 guibg=#99dddd
endfunction

map <F5> :Caps<CR>
map <F6> :Mixed<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Caps', function()
  for char = string.byte('a'), string.byte('z') do
    vim.api.nvim_set_keymap('i', string.char(char),
    ↪  string.char(char):upper(), { noremap = true })
  end
  print('CAPS ON')
  vim.cmd('highlight LineNr guifg=Red3 guibg=#cccccc')
end, {})

vim.api.nvim_create_user_command('Mixed', function()
  for char = string.byte('a'), string.byte('z') do
    vim.api.nvim_set_keymap('i', string.char(char):upper(),
    ↪  string.char(char), { noremap = true })
  end
  print('Caps Off')
  vim.cmd('highlight LineNr guifg=Red3 guibg=#99dddd')
end, {})

vim.keymap.set('n', '<F5>', ':Caps<CR>', { noremap = true })
vim.keymap.set('n', '<F6>', ':Mixed<CR>', { noremap = true })
```

Source: vim.fandom.com

17.294 Toggle Editing Helpers with Function Key

Category: configuration
Tags: toggle, insert-mode, key-mapping

Toggle automatic bracket, quote, and parenthesis insertion with a function key

Example: Example (Vim)

```
fun! s:Toggle_EditHelpers()
  if !exists('b:edithelpers_on') || b:edithelpers_on == 0
    let b:edithelpers_on=1
    inoremap '' ''<Esc>i
    inoremap ''' '''
    inoremap "" ""<Esc>i
```

```
    inoremap () ()<Esc>i
    inoremap {} {}<Esc>i
  else
    let b:edithelpers_on=0
    iunmap ''
    iunmap '''
    iunmap ""
    iunmap ()
    iunmap {}
  endif
endfun

nnoremap <silent><F9> :call <SID>Toggle_EditHelpers()<CR>
```

Example: Example (Neovim)

```
local function toggle_edit_helpers()
  if not vim.b.edithelpers_on then
    vim.b.edithelpers_on = true
    vim.keymap.set('i', "''", "''\27i")
    vim.keymap.set('i', "'''", "''''")
    vim.keymap.set('i', '""', '""\27i')
    vim.keymap.set('i', '()', '()\27i')
    vim.keymap.set('i', '{}', '{}\27i')
  else
    vim.b.edithelpers_on = false
    vim.keymap.del('i', "''")
    vim.keymap.del('i', "'''")
    vim.keymap.del('i', '""')
    vim.keymap.del('i', '()')
    vim.keymap.del('i', '{}')
  end
end

vim.keymap.set('n', '<F9>', toggle_edit_helpers, { silent = true })
```

Source: vim.fandom.com

17.295 Toggle GUI Elements in Neovim

Category: configuration
Tags: ui, interface, customization

Easily toggle or remove GUI elements like menu bar, toolbar, and scrollbars to maximize screen space

Example: Example (Vim)

```
" Remove menu bar
set guioptions-=m
```

```
" Remove toolbar
set guioptions-=T

" Remove right scrollbar
set guioptions-=r

" Toggle menu bar
nnoremap <C-F1> :if &go=~#'m'<Bar>set go-=m<Bar>else<Bar>set
↪  go+=m<Bar>endif<CR>
```

Example: Example (Neovim)

```
-- Remove menu bar
vim.o.guioptions = vim.o.guioptions:gsub('m', '')

-- Remove toolbar
vim.o.guioptions = vim.o.guioptions:gsub('T', '')

-- Remove right scrollbar
vim.o.guioptions = vim.o.guioptions:gsub('r', '')

-- Toggle menu bar
vim.keymap.set('n', '<C-F1>', function()
  if vim.o.guioptions:find('m') then
    vim.o.guioptions = vim.o.guioptions:gsub('m', '')
  else
    vim.o.guioptions = vim.o.guioptions .. 'm'
  end
end, { desc = 'Toggle menu bar' })
```

Source: vim.fandom.com

17.296 Toggle Line Numbers Quickly

Category: configuration
Tags: line-numbers, ui, key-mapping

Easily toggle line numbers on and off with a single key mapping

Example: Example (Vim)

```
map gn :set nu<CR>
map gN :set nonu<CR>

" Alternative toggle version:
map gn :set invnu<CR>
```

Example: Example (Neovim)

```
-- Toggle line numbers
vim.keymap.set('n', 'gn', function()
  vim.wo.number = not vim.wo.number
end, { desc = 'Toggle line numbers' })
```

Source: vim.fandom.com

17.297 Toggle Options Easily

Category: configuration
Tags: options, toggle, configuration

Quick way to toggle boolean options like spell checking on and off

Example: Example (Vim)

```
set spell!
```

Example: Example (Neovim)

```
vim.opt.spell = not vim.opt.spell:get()
```

Source: vim.fandom.com

17.298 Toggle Paste Mode to Preserve Indentation

Category: configuration
Tags: paste-mode, indentation, clipboard

Prevent unwanted auto-indentation when pasting code from external sources by using paste toggle

Example: Example (Vim)

```
set pastetoggle=<F2>

" Optional visual feedback
nnoremap <F2> :set invpaste paste?<CR>
set showmode
```

Example: Example (Neovim)

```
-- Set pastetoggle key
vim.o.pastetoggle = '<F2>'
```

```
-- Optional mapping with visual feedback
vim.keymap.set('n', '<F2>', function()
  vim.o.paste = not vim.o.paste
  print('Paste mode: ' .. (vim.o.paste and 'ON' or 'OFF'))
end, { desc = 'Toggle paste mode' })
```

Source: vim.fandom.com

17.299 Toggle Vim Option Flags Dynamically

Category: configuration
Tags: options, toggle, key-mapping

Create a flexible function to toggle specific option flags, allowing easy runtime configuration of Vim settings

Example: Example (Vim)

```
function ToggleFlag(option,flag)
  exec ('let lopt = &' . a:option)
  if lopt =~ (".*" . a:flag . ".*")
    exec ('set ' . a:option . '-=' . a:flag)
  else
    exec ('set ' . a:option . '+=' . a:flag)
  endif
endfunction

" Example mappings
map <silent> <F8> :call ToggleFlag("guioptions","m")<CR>
map <silent> <F9> :call ToggleFlag("guioptions","T")<CR>
```

Example: Example (Neovim)

```
function _G.toggle_flag(option, flag)
  local current_opts = vim.o[option]
  if current_opts:find(flag) then
    vim.o[option] = current_opts:gsub(flag, '')
  else
    vim.o[option] = current_opts .. flag
  end
end

-- Example mappings
vim.keymap.set('n', '<F8>', function() _G.toggle_flag('guioptions', 'm')
↪  end)
vim.keymap.set('n', '<F9>', function() _G.toggle_flag('guioptions', 'T')
↪  end)
```

Source: vim.fandom.com

17.300 Toggle Vim Options Easily

Category: configuration
Tags: options, toggle, settings

Provides multiple ways to manipulate boolean Vim options, including toggling, checking, and resetting to default

Example: Example (Vim)

```
:set number
:set nonumber
:set invnumber
:set number!
:set number&
:set number?
```

Example: Example (Neovim)

```
-- Toggle line numbers
vim.opt.number = not vim.opt.number:get()

-- Check current option value
print(vim.opt.number:get())

-- Reset to default
vim.opt.number:set(vim.o.number)
```

Source: vim.fandom.com

17.301 Toggle Word Completion and Paste Mode

Category: configuration
Tags: key-mapping, plugin, mode-switching

Provides a flexible way to toggle word completion and paste mode using F12 key, with smart switching between normal and insert modes

Example: Example (Vim)

```
fun! SetComplete()
  call DoWordComplete()
  set nopaste
  nunmap <F12>
  iunmap <F12>
  nmap <F12> :call UnsetComplete()<CR>
  imap <F12> <Esc>:call UnsetComplete()<CR>a
  echo
endfun
```

```
fun! UnsetComplete()
  call EndWordComplete()
  set paste
  nunmap <F12>
  iunmap <F12>
  nmap <F12> :call SetComplete()<CR>
  imap <F12> <Esc>:call SetComplete()<CR>a
  echo
endfun

nmap <F12> :call UnsetComplete()<CR>
imap <F12> <Esc>:call UnsetComplete()<CR>a
```

Example: Example (Neovim)

```
local function set_complete()
  vim.fn.DoWordComplete()
  vim.o.paste = false
  vim.keymap.del('n', '<F12>')
  vim.keymap.del('i', '<F12>')
  vim.keymap.set('n', '<F12>', unset_complete)
  vim.keymap.set('i', '<F12>', '<Esc>' .. unset_complete .. 'a')
end

local function unset_complete()
  vim.fn.EndWordComplete()
  vim.o.paste = true
  vim.keymap.del('n', '<F12>')
  vim.keymap.del('i', '<F12>')
  vim.keymap.set('n', '<F12>', set_complete)
  vim.keymap.set('i', '<F12>', '<Esc>' .. set_complete .. 'a')
end

vim.keymap.set('n', '<F12>', unset_complete)
vim.keymap.set('i', '<F12>', '<Esc>' .. unset_complete .. 'a')
```

Source: vim.fandom.com

17.302 Toggle Wrapping with Cursor Movement Handling

Category: configuration
Tags: key-mapping, text-wrapping, toggle

Implement a function to toggle text wrapping and dynamically adjust cursor movement keys based on wrap state

Example: Example (Vim)

```
function! ToggleWrap()
  if &wrap
    echo "Wrap OFF"
```

```
    setlocal nowrap
    set virtualedit=all
    silent! nunmap <buffer> <Up>
    silent! nunmap <buffer> <Down>
  else
    echo "Wrap ON"
    setlocal wrap linebreak nolist
    set virtualedit=
    noremap <buffer> <silent> <Up> gk
    noremap <buffer> <silent> <Down> gj
  endif
endfunction

noremap <leader>w :call ToggleWrap()<CR>
```

Example: Example (Neovim)

```
function _G.toggle_wrap()
  if vim.wo.wrap then
    print('Wrap OFF')
    vim.wo.wrap = false
    vim.o.virtualedit = 'all'
  else
    print('Wrap ON')
    vim.wo.wrap = true
    vim.wo.linebreak = true
    vim.o.virtualedit = ''
  end
end

vim.keymap.set('n', '<leader>w', _G.toggle_wrap, { desc = 'Toggle text
↪  wrapping' })
```

Source: vim.fandom.com

17.303 Toggle and Manage Vim Boolean Options

Category: configuration
Tags: options, toggle, settings

Multiple ways to manage boolean options like line numbers, with toggle and query capabilities

Example: Example (Vim)

```
" Toggle line numbers
:set number
:set nonumber
:set invnumber
:set number!
```

```
" Query current option value
:set number?
```

Example: Example (Neovim)

```
-- Toggle line numbers
vim.opt.number = true  -- Turn on
vim.opt.number = false  -- Turn off

-- Query current option value
print(vim.opt.number:get())
```

Source: vim.fandom.com

17.304 Troubleshoot Color Scheme Loading Issues

Category: configuration
Tags: color-scheme, troubleshooting, ui

If color scheme isn't loading properly, try removing or commenting out highlight commands in gvimrc

Example: Example (Vim)

```
" Comment out highlight lines in gvimrc
```

Example: Example (Neovim)

```
-- Remove or comment out any custom highlight configurations
```

Source: vim.fandom.com

17.305 Troubleshoot Terminal Color Issues

Category: configuration
Tags: terminal, troubleshooting, encoding

Resolve color and syntax highlighting problems by checking terminal encoding and TERM environment variable

Example: Example (Vim)

```
syntax on

" Recommended terminal settings:
" - Set encoding to UTF-8
" - Select Linux terminal emulation
```

```
" - Export TERM=linux in bashrc
```

Example: Example (Neovim)

```
-- Enable syntax highlighting
vim.cmd('syntax on')

-- Consider adding to init.lua:
-- vim.env.TERM = 'linux'
```

Source: vim.fandom.com

17.306 Ubuntu Development Package Dependencies

Category: configuration
Tags: development, dependencies, installation

Install necessary development packages to build Vim with full GUI and language support

Example: Example (Vim)

```
sudo apt-get install libncurses5-dev libgnome2-dev libgtk2.0-dev
```

Example: Example (Neovim)

```
-- Shell command for package installation
vim.fn.system('sudo apt-get install libncurses5-dev libgnome2-dev
↪  libgtk2.0-dev')
```

Source: vim.fandom.com

17.307 Understand Vim's Indentation Options

Category: configuration
Tags: indentation, settings, code-formatting

Key options that control how indentation works in Vim/Neovim

Example: Example (Vim)

```
" Key indentation settings
" tabstop: width of tab character
" softtabstop: behavior of tab/backspace keys
" shiftwidth: indent width for >> << and auto-indent
" expandtab: use spaces instead of tabs
```

Example: Example (Neovim)

```
-- Detailed explanation of indentation settings
-- Use vim.opt to configure similar settings in Neovim
```

Source: vim.fandom.com

17.308 Unique Swap Files Across Projects

Category: configuration
Tags: file-management, swapfiles, project-settings

Use double path separators to ensure unique swap file names when editing files with the same name in different directories

Example: Example (Vim)

```
set directory=~/.backup//
```

Example: Example (Neovim)

```
vim.opt.directory = {'~/.backup//'}
```

Source: vim.fandom.com

17.309 Update Vim Source with Mercurial

Category: configuration
Tags: version-control, development, source-management

Simple method to pull and update the latest Vim source code changes using Mercurial

Example: Example (Vim)

```
# Pull and update Vim source
hg pull -u
```

Example: Example (Neovim)

```
-- Lua equivalent for pulling updates
vim.fn.system('cd ~/.build/hg/vim && hg pull -u')
```

Source: vim.fandom.com

17.310 Use SuperTab for Easy Omni Completion

Category: configuration
Tags: completion, plugin, productivity

Configure SuperTab to use context-aware or omnifunc completion with a simple keypress

Example: Example (Vim)

```
" Use Tab for context-aware completion
let g:SuperTabDefaultCompletionType = "context"
```

Example: Example (Neovim)

```
-- Configure SuperTab for context-aware completion
vim.g.SuperTabDefaultCompletionType = "context"
```

Source: vim.fandom.com

17.311 Use Unix Line Endings for Cross-Platform Vim Scripts

Category: configuration
Tags: cross-platform, file-formatting, compatibility

Ensure Vim scripts work across different operating systems by using Unix-style line endings (\n) instead of Windows-style (\r\n)

Example: Example (Vim)

```
" vim:ff=unix
```

Example: Example (Neovim)

```
-- Add to top or bottom of script to enforce Unix line endings
vim.opt.fileformat = 'unix'
```

Source: vim.fandom.com

17.312 Use rsync to Sync Vim Runtime Files

Category: configuration
Tags: sync, runtime, file-management

Efficiently synchronize Vim runtime files using rsync, keeping your Vim environment up-to-date

Example: Example (Vim)

```
rsync -avzcP --delete --exclude="/dos/" ftp.nluug.nl::Vim/runtime/
↪  ./runtime/
```

Example: Example (Neovim)

```
-- Use shell command in Lua
vim.fn.system('rsync -avzcP --delete --exclude="/dos/"
↪  ftp.nluug.nl::Vim/runtime/ ./runtime/')
```

Source: vim.fandom.com

17.313 Use scriptencoding for Vimrc Character Sets

Category: configuration
Tags: encoding, script-configuration

Specify script encoding to ensure correct interpretation of character sets in configuration files

Example: Example (Vim)

```
scriptencoding latin1
```

Example: Example (Neovim)

```
-- In Lua, encoding is typically handled by the runtime environment
-- Use UTF-8 by default in Neovim
```

Source: vim.fandom.com

17.314 Use scriptencoding for Vimrc Encoding

Category: configuration
Tags: encoding, vimrc, script-configuration

A simpler way to specify script encoding without affecting global Vim settings

Example: Example (Vim)

```
scriptencoding latin1
```

Example: Example (Neovim)

```
-- Lua doesn't have a direct scriptencoding equivalent
-- But you can handle this through vim.o settings
vim.o.encoding = 'latin1'
```

Source: vim.fandom.com

17.315 User-Specific File Associations in Windows

Category: configuration
Tags: windows, customization, user-settings

Create user-specific file associations in Windows registry without affecting other users, allowing personalized file type handling

Example: Example (Vim)

```
" Conceptual Vim script for .c file association
" Requires manual registry editing
```

Example: Example (Neovim)

```
-- Conceptual Lua representation of user-specific file association
-- Typically requires external registry manipulation
-- Demonstrates principle of user-specific configuration
```

Source: vim.fandom.com

17.316 Verbose Option Checking Without Noise

Category: configuration
Tags: debugging, startup, options

Dynamically enable verbose mode only during runtime to avoid startup 'file not found' messages

Example: Example (Vim)

```
if &verbose == 0
  augroup late-verbose
    autocmd VimEnter * set verbose=1
    autocmd VimLeave * set verbose=0
  augroup END
endif
```

Example: Example (Neovim)

```
if vim.o.verbose == 0 then
  vim.api.nvim_create_augroup('late-verbose', { clear = true })
  vim.api.nvim_create_autocmd('VimEnter', {
    group = 'late-verbose',
    callback = function()
      vim.o.verbose = 1
    end
  })
  vim.api.nvim_create_autocmd('VimLeave', {
    group = 'late-verbose',
    callback = function()
      vim.o.verbose = 0
    end
  })
end
```

Source: vim.fandom.com

17.317 Verifying Python Support in Vim/Neovim

Category: configuration
Tags: python, debugging

Check Python scripting capabilities and installed versions

Example: Example (Vim)

```
:py print 2**0.5
:py import sys; print sys.version
:py3 print(2**0.5)
:py3 import sys; print(sys.version)
```

Example: Example (Neovim)

```
-- Lua equivalents for checking Python support
print(vim.fn.has('python'))
print(vim.fn.has('python3'))
-- Use vim.cmd to run Python checks
vim.cmd('py print(2**0.5)')
vim.cmd('py3 print(2**0.5)')
```

Source: vim.fandom.com

17.318 Version Control for Vim Configuration

Category: configuration
Tags: version-control, dotfiles, backup

Use distributed version control (like Git) to manage and backup Vim/Neovim configuration files across multiple machines

Example: Example (Vim)

```
" Recommended approach: Use Git to version control .vimrc and .vim directory
```

Example: Example (Neovim)

```
-- Create a Git repository for your Neovim config
-- Store in ~/.config/nvim and use symbolic links if needed
-- Allows easy sync and backup across different machines
```

Source: vim.fandom.com

17.319 Version-Independent Vim Installation on Windows

Category: configuration
Tags: windows, installation, file-organization

Create a consistent Vim directory structure that separates runtime files from personal configurations, making version upgrades easier

Example: Example (Neovim)

```
-- Recommended directory structure for Vim on Windows
-- vim/
--   vim/runtime/
--   vim/vimfiles/
--   vim/gvim.exe
--   vim/_vimrc

-- Automatically set $VIM and $VIMRUNTIME
-- Keeps personal scripts separate from official distribution
-- Makes upgrading Vim versions smoother
```

Source: vim.fandom.com

17.320 Version-Safe Script Loading

Category: configuration
Tags: version-check, compatibility, script-loading

Safely load configuration and scripts by checking Vim version before applying settings

Example: Example (Vim)

```
if version >= 600
 set foldcolumn=2
endif
```

Example: Example (Neovim)

```
if vim.fn.has('nvim-0.6') == 1 then
  vim.opt.foldcolumn = '2'
end
```

Source: vim.fandom.com

17.321 View Java Docs Directly in Vim

Category: configuration
Tags: documentation, java, external-tools

Use a Vim doclet to generate help files for Java documentation, allowing quick access to Java API docs directly within Vim

Example: Example (Vim)

```
" Configure keywordprg to access Java docs
set keywordprg=
" Use 'K' to bring up docs for Java keywords
```

Example: Example (Neovim)

```
-- In Neovim, you can set keywordprg similarly
vim.o.keywordprg = ''
-- Recommended to use a plugin like vimdoclet for Java docs
```

Source: vim.fandom.com

17.322 View runtime paths

Category: Configuration
Tags: runtime, path, debug

Use `:echo &runtimepath` to see all runtime paths Neovim is using.

Example: Example (Vim)

```
:echo &runtimepath  " show runtime paths
```

Source: Community contributed

17.323 Vim-Style Editing in Visual Studio

Category: configuration
Tags: vim-emulation, ide-integration

Use VsVim or ViEmu to get Vim-like editing commands in Visual Studio

Example: Example (Vim)

```
-- Recommendation to install VsVim extension
```

Example: Example (Neovim)

```
-- Recommend installing VsVim extension for Visual Studio
-- No direct Neovim equivalent, as this is IDE-specific
```

Source: vim.fandom.com

17.324 Windows Vim Download Options

Category: configuration
Tags: installation, windows, download

Multiple ways to download Vim for Windows, including stable and nightly builds

Example: Example (Vim)

```
" Recommended sources:
" - Nightly builds: https://github.com/vim/vim-win32-installer
" - Stable builds: http://sourceforge.net/projects/cream/files/Vim/
```

Example: Example (Neovim)

```
-- Recommended Vim download sources
-- Nightly builds: https://github.com/vim/vim-win32-installer
-- Stable builds: http://sourceforge.net/projects/cream/files/Vim/
```

Source: vim.fandom.com

17.325 Windows Vim Installation After Compilation

Category: configuration
Tags: windows, installation, build

Comprehensive steps to install compiled Vim on Windows system

Example: Example (Vim)

```
# Steps to install after compilation
1. Create directory like C:\Program Files\vim\vim74
2. Copy compiled .exe files
3. Copy runtime files
4. Run install.exe as administrator
```

Example: Example (Neovim)

```
-- Lua equivalent is documentation of installation steps
-- Actual installation is a system/shell process
```

Source: vim.fandom.com

17.326 Windows-like Behavior in Vim

Category: configuration
Tags: key-mapping, behavior

Use built-in Vim commands to mimic Windows-style text selection and clipboard behavior

Example: Example (Vim)

```
set keymodel=behave
set selectmode=key
" Or use: :behave mswin
```

Example: Example (Neovim)

```
vim.o.keymodel = 'behave'
vim.o.selectmode = 'key'
-- Alternatively:
vim.cmd('behave mswin')
```

Source: vim.fandom.com

CHAPTER 18

Cut and paste

18.1 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: Example (Vim)

```
dw  " delete to next word
de  " delete to end of word
db  " delete to start of word
```

18.2 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: Example (Vim)

```
p  " paste after
P  " paste before
```

18.3 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: Example (Vim)

```
[p  " paste after with auto-indent
[P  " paste before with auto-indent
]p  " paste after with auto-indent
]P  " same as [P
```

18.4 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: Example (Vim)

```
yy   " yank current line
3yy  " yank 3 lines
```

18.5 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: Example (Vim)

```
yw  " yank to next word
ye  " yank to end of word
```

CHAPTER 19

Diagnostics

19.1 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: Example (Neovim)

```
-- 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' })
```

19.2 Find mapping source

Category: Diagnostics
Tags: mapping, verbose, source

Use `:verbose map <key>` to see where a specific mapping was defined.

Example: Example (Vim)

```
:verbose map <leader>f  " see where <leader>f was mapped
```

19.3 Find option source

Category: Diagnostics
Tags: option, verbose, source

Use `:verbose set option?` to see where a specific option was last set.

Example: Example (Vim)

```
:verbose set number?  " see where 'number' option was set
```

19.4 Health diagnostics

Category: Diagnostics
Tags: health, check, diagnostics

Use `:checkhealth` to run health diagnostics for your Neovim setup.

Example: Example (Vim)

```
:checkhealth  " run health diagnostics
```

19.5 View messages

Category: Diagnostics
Tags: messages, log, history

Use `:messages` to view past messages and notifications.

Example: Example (Vim)

```
:messages  " view past messages
```

CHAPTER 20
Display

20.1 Automatic Status Line Update with Timer

Category: display
Tags: status-line, timer, auto-update

Use Vim 8+ timers to automatically refresh the status line at regular intervals

Example: Example (Vim)

```
let timer = timer_start(4000, 'UpdateStatusBar', {'repeat':-1})
function! UpdateStatusBar(timer)
  execute 'let &ro = &ro'
endfunction
```

Example: Example (Neovim)

```
local timer = vim.fn.timer_start(4000, function()
  vim.o.readonly = vim.o.readonly
end, {repeat = -1})
```

Source: vim.fandom.com

20.2 Center Cursor Vertically in Window

Category: display
Tags: scrolling, window-management, navigation

Dynamically keep the cursor centered vertically by setting scrolloff to half the window height, with a toggle function

Example: Example (Vim)

```
" Toggle centered cursor
nnoremap <Leader>zz :let &scrolloff=999-&scrolloff<CR>

" Automatic centering autocmd
augroup VCenterCursor
  au!
```

```
  au BufEnter,WinEnter,WinNew,VimResized *,*.*
        \ let &scrolloff=winheight(win_getid())/2
augroup END
```

Example: Example (Neovim)

```
-- Toggle centered cursor
vim.keymap.set('n', '<Leader>zz', function()
  vim.o.scrolloff = 999 - vim.o.scrolloff
end, { desc = 'Toggle cursor centering' })

-- Automatic centering
vim.api.nvim_create_augroup('VCenterCursor', { clear = true })
vim.api.nvim_create_autocmd({'BufEnter', 'WinEnter', 'WinNew',
↪  'VimResized'}, {
  group = 'VCenterCursor',
  pattern = {'*', '*.*'},
  callback = function()
    vim.o.scrolloff = math.floor(vim.api.nvim_win_get_height(0) / 2)
  end
})
```

Source: vim.fandom.com

20.3 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: Example (Vim)

```
" Vimscript:
: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>
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.conceallevel = 2  -- hide concealed text completely
vim.opt.conceallevel = 0  -- show all text normally
vim.cmd('syntax match htmlTag \'<[^>]*>\' conceal')  -- hide HTML tags

-- Toggle conceal on/off
vim.keymap.set('n', '<leader>c', function()
```

```
  vim.opt.conceallevel = vim.opt.conceallevel:get() == 2 and 0 or 2
end, { desc = 'Toggle conceal' })
```

Source: Community contributed

20.4 Conditionally Set Command Height

Category: display
Tags: ui, startup-configuration

Automatically increase command height to prevent 'Hit-Enter' prompts

Example: Example (Vim)

```
if &cmdheight == 1
  set cmdheight=2
endif
```

Example: Example (Neovim)

```
if vim.o.cmdheight == 1 then
  vim.o.cmdheight = 2
end
```

Source: vim.fandom.com

20.5 Configure Scrolloff for Context

Category: display
Tags: scrolling, view-management, configuration

Set a minimum number of lines visible above and below the cursor to maintain context during navigation

Example: Example (Vim)

```
" Ensure at least 5 lines of context around cursor
set scrolloff=5

" Keep cursor always centered
set scrolloff=999
```

Example: Example (Neovim)

```
-- Set minimum context lines around cursor
vim.opt.scrolloff = 5  -- 5 lines of context
```

```
-- Alternatively, keep cursor always centered
vim.opt.scrolloff = 999
```

Source: vim.fandom.com

20.6 Custom Syntax Highlighting Colors

Category: display
Tags: syntax-highlighting, color-customization

Modify specific syntax highlighting colors for comments, search results, and error messages

Example: Example (Vim)

```
hi Comment ctermfg=DarkGrey guifg=DarkGrey
hi Search guibg=LightBlue
highlight ErrorMsg guibg=White guifg=Red
```

Example: Example (Neovim)

```
vim.cmd('hi Comment ctermfg=DarkGrey guifg=DarkGrey')
vim.cmd('hi Search guibg=LightBlue')
vim.cmd('highlight ErrorMsg guibg=White guifg=Red')
```

Source: vim.fandom.com

20.7 Custom Syntax Highlighting for Printing

Category: display
Tags: printing, syntax-highlighting, customization

Create a separate color scheme for printing that differs from screen display, allowing unique formatting for printed documents

Example: Example (Vim)

```
command! -nargs=* Hardcopy call DoMyPrint('<args>')

function DoMyPrint(args)
  let colorsave=g:colors_name
  color print
  exec 'hardcopy '.a:args
  exec 'color '.colorsave
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Hardcopy', function(opts)
  local current_colorscheme = vim.g.colors_name
  vim.cmd('colorscheme print')
  vim.cmd('hardcopy ' .. (opts.args or ''))
  vim.cmd('colorscheme ' .. current_colorscheme)
end, { nargs = '*' })
```

Source: vim.fandom.com

20.8 Customize GVim Colors to Match Console Vim

Category: display
Tags: color-scheme, gui-colors, theme

Customize GVim colors to match console Vim's color palette, providing a consistent visual experience across different Vim interfaces

Example: Example (Vim)

```
set background=dark
hi SpecialKey guifg=Blue
hi Normal guifg=Gray guibg=Black
hi Comment guifg=Cyan
hi Constant guifg=Magenta
hi Statement guifg=Yellow
```

Example: Example (Neovim)

```
vim.o.background = 'dark'

-- Custom highlight groups
vim.api.nvim_set_hl(0, 'SpecialKey', { fg = 'Blue' })
vim.api.nvim_set_hl(0, 'Normal', { fg = 'Gray', bg = 'Black' })
vim.api.nvim_set_hl(0, 'Comment', { fg = 'Cyan' })
vim.api.nvim_set_hl(0, 'Constant', { fg = 'Magenta' })
vim.api.nvim_set_hl(0, 'Statement', { fg = 'Yellow' })
```

Source: vim.fandom.com

20.9 Customize Line Number Appearance

Category: display
Tags: ui, configuration, theming

Customize the appearance of line numbers in the gutter

Example: Example (Vim)

```
set numberwidth=3
highlight LineNr term=bold cterm=NONE ctermfg=DarkGrey ctermbg=NONE gui=NONE
↪  guifg=DarkGrey guibg=NONE
```

Example: Example (Neovim)

```
vim.opt.numberwidth = 3
vim.api.nvim_set_hl(0, 'LineNr', {
  fg = 'DarkGrey',
  bold = true
})
```

Source: vim.fandom.com

20.10 Customize PuTTY Colors for Vim Desert Scheme

Category: display
Tags: color-scheme, terminal, customization

Detailed guide for configuring PuTTY color palette to match Vim's desert color scheme, improving terminal syntax highlighting and readability

Example: Example (Vim)

```
:colorscheme desert
```

Example: Example (Neovim)

```
vim.cmd('colorscheme desert')
```

Source: vim.fandom.com

20.11 Display Character ASCII/Unicode Value

Category: display
Tags: character-info, unicode, status-line

Show the ASCII or Unicode value of the current character in the status line, which helps with encoding and character analysis

Example: Example (Vim)

```
set statusline=%<%f%h%m%r%=%b\ 0x%B\ \ %l,%c%V\ %P
set laststatus=2
```

Example: Example (Neovim)

```
vim.opt.statusline = '%<%f%h%m%r%=%b 0x%B  %l,%c%V %P'
vim.opt.laststatus = 2
```

Source: vim.fandom.com

20.12 Display Date and Time in Status Line

Category: display
Tags: status-line, time-display, configuration

Customize the status line to show current date and time, updating when you interact with the editor

Example: Example (Vim)

```
set ruler
set rulerformat=%55(%{strftime('%a\ %b\ %e\ %I:%M\ %p')}\ %5l,%-6(%c%V%)\
↪  %P%)
```

Example: Example (Neovim)

```
vim.opt.ruler = true
vim.opt.rulerformat = '%55(%{strftime("%a %b %e %I:%M %p")} %5l,%-6(%c%V%)
↪  %P%)'
```

Source: vim.fandom.com

20.13 Dynamic Color Scheme Switcher

Category: display
Tags: color-schemes, ui, customization

Easily cycle through color schemes using keyboard shortcuts, with options to select specific color schemes or use all installed schemes

Example: Example (Vim)

```
" Color scheme switching script
nnoremap <F8> :call NextColor(1)<CR>
nnoremap <S-F8> :call NextColor(-1)<CR>
nnoremap <A-F8> :call NextColor(0)<CR>

" Set color schemes dynamically
:SetColors all     " Use all installed color schemes
:SetColors my      " Use predefined color schemes
:SetColors blue slate ron  " Use specific color schemes
```

Example: Example (Neovim)

```
-- Lua equivalent for color scheme switching
vim.keymap.set('n', '<F8>', function()
  vim.cmd('call NextColor(1)')
end, { desc = 'Next color scheme' })

vim.keymap.set('n', '<S-F8>', function()
  vim.cmd('call NextColor(-1)')
end, { desc = 'Previous color scheme' })

vim.keymap.set('n', '<A-F8>', function()
  vim.cmd('call NextColor(0)')
end, { desc = 'Random color scheme' })

-- Use :SetColors command as in Vimscript
```

Source: vim.fandom.com

20.14 Dynamic Color Testing Across Terminal and GUI

Category: display
Tags: colors, testing, terminal

Generate color test scripts for both console and GUI Vim, allowing comprehensive color visualization

Example: Example (Vim)

```
function! VimColorTest(outfile, fgend, bgend)
  let result = []
  for fg in range(a:fgend)
    for bg in range(a:bgend)
      let kw = printf('%-7s', printf('c_%d_%d', fg, bg))
      let h = printf('hi %s ctermfg=%d ctermbg=%d', kw, fg, bg)
      let s = printf('syn keyword %s %s', kw, kw)
      call add(result, printf('%-32s | %s', h, s))
    endfor
  endfor
  call writefile(result, a:outfile)
  execute 'edit '.a:outfile
  source %
endfunction
```

Example: Example (Neovim)

```
function _G.color_test(outfile, fgend, bgend)
  local result = {}
  for fg = 0, fgend - 1 do
    for bg = 0, bgend - 1 do
      local kw = string.format('c_%d_%d', fg, bg)
      local h = string.format('hi %s ctermfg=%d ctermbg=%d', kw, fg, bg)
```

```
      local s = string.format('syn keyword %s %s', kw, kw)
      table.insert(result, string.format('%-32s | %s', h, s))
    end
  end

  vim.fn.writefile(result, outfile)
  vim.cmd('edit ' .. outfile)
  vim.cmd('source %')
end

-- Usage: :lua color_test('colortest.txt', 16, 16)
```

Source: vim.fandom.com

20.15 Dynamic Cursor Line Highlighting After Jumps

Category: display
Tags: cursor-highlight, autocmd, navigation

Automatically highlight the cursor line after significant cursor jumps, making it easier to track cursor position in large files

Example: Example (Vim)

```
function s:Cursor_Moved()
  let cur_pos = winline()
  if g:last_pos == 0
    set cul
    let g:last_pos = cur_pos
    return
  endif
  let diff = g:last_pos - cur_pos
  if diff > 1 || diff < -1
    set cul
  else
    set nocul
  endif
  let g:last_pos = cur_pos
endfunction
autocmd CursorMoved,CursorMovedI * call s:Cursor_Moved()
let g:last_pos = 0
```

Example: Example (Neovim)

```
local function cursor_moved()
  local cur_pos = vim.fn.winline()
  if vim.g.last_pos == 0 then
    vim.opt.cursorline = true
    vim.g.last_pos = cur_pos
    return
  end
```

```
  local diff = vim.g.last_pos - cur_pos
  if math.abs(diff) > 1 then
    vim.opt.cursorline = true
  else
    vim.opt.cursorline = false
  end

  vim.g.last_pos = cur_pos
end

vim.api.nvim_create_autocmd({'CursorMoved', 'CursorMovedI'}, {
  callback = cursor_moved
})

vim.g.last_pos = 0
```

Source: vim.fandom.com

20.16 Dynamic Font Size Adjustment in GUI

Category: display
Tags: gui, font-size, key-mapping

Quickly change font size in GVim using function and key mappings for better readability

Example: Example (Vim)

```
function! ScaleFontUp()
  let gf_size_whole = matchstr(&guifont, '\(\:h\)\@<=\d\+')
  let gf_size_frac = matchstr(&guifont, '\(\:h\d\+\.\)\@<=\d\=')
  let font_size = gf_size_whole * 10 + gf_size_frac
  let font_size = font_size + 5
  let gf_size_whole = font_size / 10
  let gf_size_frac = font_size - gf_size_whole * 10
  let new_font_size = ':h'.gf_size_whole.'.'.gf_size_frac.':'
  let &guifont = substitute(&guifont, '\:h.\{-}\:', new_font_size, '')
endfunction

function! ScaleFontDown()
  let gf_size_whole = matchstr(&guifont, '\(\:h\)\@<=\d\+')
  let gf_size_frac = matchstr(&guifont, '\(\:h\d\+\.\)\@<=\d\=')
  let font_size = gf_size_whole * 10 + gf_size_frac
  let font_size = font_size - 5
  let gf_size_whole = font_size / 10
  let gf_size_frac = font_size - gf_size_whole * 10
  let new_font_size = ':h'.gf_size_whole.'.'.gf_size_frac.':'
  let &guifont = substitute(&guifont, '\:h.\{-}\:', new_font_size, '')
endfunction

nmap <C-S-F11> :call ScaleFontDown()<CR>
nmap <C-S-F12> :call ScaleFontUp()<CR>
```

Example: Example (Neovim)

```
local function scale_font_up()
  local guifont = vim.o.guifont
  local current_size = tonumber(guifont:match(':h([%d.]+)'))
  local new_size = current_size and (current_size + 0.5) or 10
  vim.o.guifont = guifont:gsub(':h[%d.]+', ':h' .. string.format('%.1f',
  ↪  new_size))
end

local function scale_font_down()
  local guifont = vim.o.guifont
  local current_size = tonumber(guifont:match(':h([%d.]+)'))
  local new_size = current_size and (current_size - 0.5) or 10
  vim.o.guifont = guifont:gsub(':h[%d.]+', ':h' .. string.format('%.1f',
  ↪  new_size))
end

vim.keymap.set('n', '<C-S-F11>', scale_font_down, { desc = 'Decrease font
↪  size' })
vim.keymap.set('n', '<C-S-F12>', scale_font_up, { desc = 'Increase font
↪  size' })
```

Source: vim.fandom.com

20.17 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: Example (Vim)

```
" Vimscript:
:set foldcolumn=4     " show fold indicators in 4-char column
:set fillchars=fold:.,vert:|  " customize fill characters
:set foldtext=MyFoldText()    " custom fold text function
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.foldcolumn = '4'  -- show fold indicators in 4-char column
vim.opt.fillchars = { fold = '.', vert = '|' }  -- customize fill characters
vim.opt.foldtext = 'v:lua.MyFoldText()'  -- custom fold text function
```

Source: Community contributed

20.18 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: Example (Vim)

```
" Vimscript:
: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%%
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.laststatus = 2  -- always show status line
vim.opt.showtabline = 2  -- always show tab line
vim.opt.statusline = '%f %m%r%h%w [%Y] [%{&ff}] %=%l,%c %p%%'
```

Source: Community contributed

20.19 Force Redraw to Prevent Message Disappearance

Category: display
Tags: messages, redraw

Use :redraw to ensure messages are displayed and not overwritten by postponed redraws

Example: Example (Vim)

```
:new | redraw | echo "there is a new window"
```

Example: Example (Neovim)

```
vim.cmd('new')
vim.cmd('redraw')
vim.api.nvim_echo({{'there is a new window', 'Normal'}}, false, {})
```

Source: vim.fandom.com

20.20 Get Character Information Quickly

Category: display
Tags: character-info, unicode, command-mode

Use built-in commands to quickly retrieve ASCII/Unicode values of the current character in decimal, hex, and octal formats

Example: Example (Vim)

```
" In command mode, press ga
" Or use :ascii command
```

Example: Example (Neovim)

```
-- In command mode, press ga
-- Or use :ascii command
-- Neovim inherits these Vim commands directly
```

Source: vim.fandom.com

20.21 Get Character Value in Command Mode

Category: display
Tags: character-info, command-mode

Quickly view ASCII/Unicode value of the current character using built-in commands

Example: Example (Vim)

```
" Use ga in normal mode
" Or :ascii/:as in command mode
```

Example: Example (Neovim)

```
-- These commands work the same in Neovim
-- Press 'ga' in normal mode
-- Or use :ascii or :as in command mode
```

Source: vim.fandom.com

20.22 Highlight Code Between Matching Curly Braces

Category: display
Tags: syntax-highlighting, code-navigation

Automatically highlight the code section between matching curly braces to help identify

code blocks and prevent errors

Example: Example (Vim)

```
highlight ShowMatches guibg=darkgrey guifg=white
au! Cursorhold * exe 'match ShowMatches /\v%(%#\{%(%(\....))/'
set ut=30
```

Example: Example (Neovim)

```
-- Lua equivalent for highlighting matching braces
vim.api.nvim_create_autocmd('CursorHold', {
  callback = function()
    vim.fn.matchadd('ShowMatches', '\v%(%#\{%(%(\....))/)
  end
})

vim.api.nvim_set_hl(0, 'ShowMatches', { bg = 'darkgrey', fg = 'white' })
vim.o.updatetime = 30
```

Source: vim.fandom.com

20.23 Highlight Current Line and Column

Category: display
Tags: ui, cursor-highlight, navigation

Dynamically highlight the current line and column to improve cursor visibility and spatial awareness in large files

Example: Example (Vim)

```
" Highlight current line and column
:hi CursorLine   cterm=NONE ctermbg=darkred ctermfg=white guibg=darkred
↪  guifg=white
:hi CursorColumn cterm=NONE ctermbg=darkred ctermfg=white guibg=darkred
↪  guifg=white
:nnoremap <Leader>c :set cursorline! cursorcolumn!<CR>
```

Example: Example (Neovim)

```
-- Highlight current line and column
vim.api.nvim_set_hl(0, 'CursorLine', { bg = 'darkred', fg = 'white' })
vim.api.nvim_set_hl(0, 'CursorColumn', { bg = 'darkred', fg = 'white' })
vim.keymap.set('n', '<Leader>c', function()
  vim.wo.cursorline = not vim.wo.cursorline
  vim.wo.cursorcolumn = not vim.wo.cursorcolumn
end, { desc = 'Toggle cursor line and column highlight' })
```

Source: vim.fandom.com

20.24 Highlight Cursor Line in Insert Mode

Category: display
Tags: mode-feedback, visual-cue, autocmd

Provides a visual indicator of insert mode by changing the cursor line background color

Example: Example (Vim)

```
augroup InsertModeHighlight
    autocmd InsertEnter * highlight CursorLine ctermbg=17
    autocmd InsertLeave * highlight CursorLine ctermbg=none
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('InsertModeHighlight', { clear = true })

vim.api.nvim_create_autocmd('InsertEnter', {
    group = 'InsertModeHighlight',
    callback = function()
        vim.cmd('highlight CursorLine guibg=#000055')
    end
})

vim.api.nvim_create_autocmd('InsertLeave', {
    group = 'InsertModeHighlight',
    callback = function()
        vim.cmd('highlight CursorLine guibg=NONE')
    end
})
```

Source: vim.fandom.com

20.25 Highlight Debug Blocks in C/C++ Code

Category: display
Tags: syntax-highlighting, debugging, code-visualization

Create custom syntax highlighting for debug blocks to easily distinguish and visually skip debug-related code sections

Example: Example (Vim)

```
syn region MySkip contained start="^\s*#\s*\(if\>\|ifdef\>\|ifndef\>\)"
↪  skip="\\$" end="^\s*#\s*endif\>" contains=MySkip

let g:CommentDefines = ""

hi link MyCommentOut2 MyCommentOut
```

```
hi link MySkip MyCommentOut
hi link MyCommentOut Comment

map <silent> ,a :call AddCommentDefine()<CR>
map <silent> ,x :call ClearCommentDefine()<CR>

function! AddCommentDefine()
  let g:CommentDefines = "\\(" . expand("<cword>") . "\\)"
  syn clear MyCommentOut
  syn clear MyCommentOut2
  exe 'syn region MyCommentOut start="^\s*#\s*ifdef\s\+' . g:CommentDefines
  ↪  . '\>" end=".\|$" contains=MyCommentOut2'
  exe 'syn region MyCommentOut2 contained start="' . g:CommentDefines . '"
  ↪  end="^\s*#\s*\(endif\>\|else\>\|elif\>\)" contains=MySkip'
endfunction

function! ClearCommentDefine()
  let g:ClearCommentDefine = ""
  syn clear MyCommentOut
  syn clear MyCommentOut2
endfunction
```

Example: Example (Neovim)

```
vim.cmd[[
  augroup DebugHighlight
    autocmd!
    autocmd Syntax * syn region MySkip contained
    ↪  start="^\s*#\s*\(if\>\|ifdef\>\|ifndef\>\)" skip="\\$"
    ↪  end="^\s*#\s*endif\>" contains=MySkip
  augroup END

  hi link MyCommentOut2 MyCommentOut
  hi link MySkip MyCommentOut
  hi link MyCommentOut Comment
]]

-- Function to add comment define
function _G.add_comment_define()
  local word = vim.fn.expand('<cword>')
  vim.cmd('syn clear MyCommentOut')
  vim.cmd('syn clear MyCommentOut2')
  vim.cmd(string.format('syn region MyCommentOut
  ↪  start="^\s*#\s*ifdef\s\+%s\>" end=".\|$" contains=MyCommentOut2',
  ↪  word))
  vim.cmd(string.format('syn region MyCommentOut2 contained start="%s"
  ↪  end="^\s*#\s*\(endif\>\|else\>\|elif\>\)" contains=MySkip', word))
end

-- Function to clear comment define
function _G.clear_comment_define()
  vim.cmd('syn clear MyCommentOut')
  vim.cmd('syn clear MyCommentOut2')
end
```

```
-- Key mappings
vim.keymap.set('n', ',a', _G.add_comment_define, { desc = 'Add comment
↪  define' })
vim.keymap.set('n', ',x', _G.clear_comment_define, { desc = 'Clear comment
↪  define' })
```

Source: vim.fandom.com

20.26 Highlight Lines Exceeding Length Limit

Category: display
Tags: highlighting, text-width, code-style

Automatically highlight lines that exceed a specified character width, helping maintain code readability and consistent formatting

Example: Example (Vim)

```
:let w:m1=matchadd('Search', '\%<81v.\%>77v', -1)
:let w:m2=matchadd('ErrorMsg', '\%>80v.\+', -1)

" Automatically apply on buffer enter
:au BufWinEnter * let w:m1=matchadd('Search', '\%<81v.\%>77v', -1)
:au BufWinEnter * let w:m2=matchadd('ErrorMsg', '\%>80v.\+', -1)
```

Example: Example (Neovim)

```
-- Highlight lines exceeding 80 characters
vim.api.nvim_create_autocmd('BufWinEnter', {
  callback = function()
    vim.fn.matchadd('Search', '\%<81v.\%>77v', -1)
    vim.fn.matchadd('ErrorMsg', '\%>80v.\+', -1)
  end
})
```

Source: vim.fandom.com

20.27 Highlight Long Lines

Category: display
Tags: syntax-highlighting, code-style, visual-guide

Visually highlight lines that exceed a specified length, helping maintain code readability and adherence to line length standards

Example: Example (Vim)

```
highlight NearColLimit term=italic,bold cterm=italic ctermbg=yellow
↪  ctermfg=darkblue gui=bold,italic guibg=yellow guifg=darkblue
highlight OverColLimit term=inverse,bold cterm=bold ctermbg=red
↪  ctermfg=white gui=bold guibg=red guifg=white
syntax match NearColLimit /\%<81v.\%>77v/
syntax match OverColLimit /\%>80v.\+/
```

Example: Example (Neovim)

```
vim.cmd('highlight NearColLimit term=italic,bold cterm=italic ctermbg=yellow
↪  ctermfg=darkblue gui=bold,italic guibg=yellow guifg=darkblue')
vim.cmd('highlight OverColLimit term=inverse,bold cterm=bold ctermbg=red
↪  ctermfg=white gui=bold guibg=red guifg=white')
vim.cmd('syntax match NearColLimit /\%<81v.\%>77v/')
vim.cmd('syntax match OverColLimit /\%>80v.\+/')
```

Source: vim.fandom.com

20.28 Highlight Matching Braces

Category: display
Tags: ui, editing, visual-cues

Briefly highlight matching braces when typing to improve code readability and navigation

Example: Example (Vim)

```
set showmatch
set matchtime=3  " Highlight for 300ms"
```

Example: Example (Neovim)

```
vim.opt.showmatch = true
vim.opt.matchtime = 3
```

Source: vim.fandom.com

20.29 Highlight Multiple Words in Different Colors

Category: display
Tags: search, highlighting, visual-enhancement

Easily highlight multiple words with different colors in a single buffer, allowing for quick visual pattern recognition and search navigation

Example: Example (Vim)

```
" Toggle highlight mapping
map <leader>m :call ToggleHighlightMapping()<CR>

" Highlight word under cursor
" Use numeric keypad 1-9 to highlight with different colors
```

Example: Example (Neovim)

```
-- Example implementation in Lua
-- Note: Full conversion would require custom function
vim.keymap.set('n', '<leader>m', function()
  -- Toggle highlight mapping logic
end, { desc = 'Toggle multiple word highlighting' })
```

Source: vim.fandom.com

20.30 Highlight Multiple Words with Different Colors

Category: display
Tags: highlighting, search, visual-enhancement

Easily highlight multiple words in different colors across a document, with advanced search and management capabilities for word highlights

Example: Example (Vim)

```
" Enable multiple word highlighting
" Toggle highlight mode
map <leader>m :call ToggleHighlight()<CR>

" Highlight current word
nnoremap <silent> <k1> :call HighlightWord(1)<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for multiple word highlighting
vim.keymap.set('n', '<leader>m', function()
  -- Toggle highlight mode
end, { desc = 'Toggle multiple word highlighting' })

vim.keymap.set('n', '<k1>', function()
  -- Highlight current word in first highlight group
end, { desc = 'Highlight current word' })
```

Source: vim.fandom.com

20.31 Highlight Problematic Whitespace

Category: display
Tags: whitespace, syntax-highlighting, code-quality

Dynamically highlight various whitespace issues like trailing spaces, mixed indentation, and misplaced tabs to improve code cleanliness

Example: Example (Vim)

```
function! ShowWhitespace(flags)
  let bad = ''
  let pat = []
  for c in split(a:flags, '\zs')
    if c == 'e'
      call add(pat, '\s\+$')
    elseif c == 'i'
      call add(pat, '^\t*\zs \+')
    elseif c == 's'
      call add(pat, ' \+\ze\t')
    elseif c == 't'
      call add(pat, '[^\t]\zs\t\+')
    else
      let bad .= c
    endif
  endfor
  if len(pat) > 0
    let s = join(pat, '\|')
    exec 'syntax match ExtraWhitespace "'.s.'" containedin=ALL'
  else
    syntax clear ExtraWhitespace
  endif
endfunction

nnoremap <Leader>ws :call ToggleShowWhitespace()<CR>
highlight ExtraWhitespace ctermbg=darkgreen guibg=darkgreen
```

Example: Example (Neovim)

```
local function show_whitespace(flags)
  local pat = {}
  local bad = ''

  for c in flags:gmatch('.') do
    if c == 'e' then
      table.insert(pat, '\s\+$')
    elseif c == 'i' then
      table.insert(pat, '^\t*\zs \+')
    elseif c == 's' then
      table.insert(pat, ' \+\ze\t')
    elseif c == 't' then
      table.insert(pat, '[^\t]\zs\t\+')
    else
```

```
      bad = bad .. c
    end
  end

  if #pat > 0 then
    local s = table.concat(pat, '\|')
    vim.cmd('syntax match ExtraWhitespace "' .. s .. '" containedin=ALL')
  else
    vim.cmd('syntax clear ExtraWhitespace')
  end

  if #bad > 0 then
    print('ShowWhitespace ignored: ' .. bad)
  end
end

vim.keymap.set('n', '<Leader>ws', function()
  vim.cmd('highlight ExtraWhitespace ctermbg=darkgreen guibg=darkgreen')
  show_whitespace('est')
end)
```

Source: vim.fandom.com

20.32 Highlight Unwanted Whitespace

Category: display
Tags: whitespace, highlighting, code-cleanup

Automatically highlight trailing spaces, spaces before tabs, and other problematic whitespace to maintain clean code

Example: Example (Vim)

```
" Highlight trailing whitespace and spaces before tabs
autocmd ColorScheme * highlight ExtraWhitespace ctermbg=red guibg=red
match ExtraWhitespace /\s\+$\| \+\ze\t/

" Prevent highlighting during insert mode
autocmd InsertEnter * match ExtraWhitespace /\s\+\%#\@<!$/
autocmd InsertLeave * match ExtraWhitespace /\s\+$/
```

Example: Example (Neovim)

```
-- Highlight unwanted whitespace in Neovim
vim.api.nvim_create_augroup('WhitespaceHighlight', { clear = true })

vim.api.nvim_create_autocmd('ColorScheme', {
  group = 'WhitespaceHighlight',
  callback = function()
    vim.api.nvim_set_hl(0, 'ExtraWhitespace', { bg = 'red' })
  end
```

```
})

-- Match and highlight trailing whitespace
vim.api.nvim_create_autocmd({'InsertEnter', 'InsertLeave'}, {
  group = 'WhitespaceHighlight',
  pattern = '*',
  callback = function(ev)
    local hl_pattern = ev.event == 'InsertEnter'
      and '\s\+\%#\@<!$'
      or '\s\+$'
    vim.fn.matchadd('ExtraWhitespace', hl_pattern)
  end
})
```

Source: vim.fandom.com

20.33 Highlight Valid IP Addresses

Category: display
Tags: syntax-highlighting, regex, pattern-matching

Automatically highlight valid IP addresses in a file using a sophisticated regex pattern

Example: Example (Vim)

```
syn match ipaddr /\(\(25\_[0-5]\|2\_[0-4]\_[0-9]\|\_[01]\?\_[0-9]\_[0-9]\?\
↪  )\.\)\{3\}\(25\_[0-5]\|2\_[0-4]\_[0-9]\|\_[01]\?\_[0-9]\_[0-9]\?\)/
hi link ipaddr Identifier
```

Example: Example (Neovim)

```
vim.cmd([[syn match ipaddr /\(\(25\[0-5]\|2\[0-4]\[0-9]\|\[01]?\[0-9]\[0-9]
↪  ?\)\.\)\{3\}\(25\[0-5]\|2\[0-4]\[0-9]\|\[01]?\[0-9]\[0-9]?\)/]])
vim.cmd('hi link ipaddr Identifier')
```

Source: vim.fandom.com

20.34 Highlight and Manage Unwanted Whitespace

Category: display
Tags: whitespace, highlighting, formatting

Easily identify and manage trailing whitespace, tabs, and inconsistent indentation across files

Example: Example (Vim)

```
" Highlight trailing whitespace
autocmd ColorScheme * highlight ExtraWhitespace ctermbg=red guibg=red
match ExtraWhitespace /\s\+$/

" Prevent whitespace highlighting during insert mode
autocmd InsertEnter * match ExtraWhitespace /\s\+\%#\@<!$/
autocmd InsertLeave * match ExtraWhitespace /\s\+$/
```

Example: Example (Neovim)

```
local function setup_whitespace_highlight()
  vim.api.nvim_create_augroup('WhitespaceHighlight', { clear = true })

  vim.api.nvim_create_autocmd('ColorScheme', {
    group = 'WhitespaceHighlight',
    callback = function()
      vim.api.nvim_set_hl(0, 'ExtraWhitespace', { bg = 'red' })
    end
  })

  vim.api.nvim_create_autocmd('InsertEnter', {
    group = 'WhitespaceHighlight',
    pattern = '*',
    command = 'match ExtraWhitespace /\s\+\%#\@<!$/'
  })

  vim.api.nvim_create_autocmd('InsertLeave', {
    group = 'WhitespaceHighlight',
    pattern = '*',
    command = 'match ExtraWhitespace /\s\+$/'
  })
end

setup_whitespace_highlight()
```

Source: vim.fandom.com

20.35 Ignore Whitespace in Diff Mode

Category: display
Tags: diff, whitespace, comparison

Ignore whitespace changes when performing diff operations, reducing noise in comparisons

Example: Example (Vim)

```
set diffopt+=iwhite
```

Example: Example (Neovim)

```
vim.opt.diffopt:append('iwhite')
```

Source: vim.fandom.com

20.36 Improve Long Line Readability

Category: display
Tags: wrapping, line-display, readability

Configure Vim/Neovim to handle long lines more gracefully with improved wrapping and display settings

Example: Example (Vim)

```
" Settings for better long line handling
set wrap
set linebreak
set display+=lastline
set scrolloff=99999
```

Example: Example (Neovim)

```
-- Settings for better long line handling
vim.opt.wrap = true
vim.opt.linebreak = true
vim.opt.display:append('lastline')
vim.opt.scrolloff = 99999
```

Source: vim.fandom.com

20.37 Improve Wrapped Line Handling

Category: display
Tags: text-display, line-wrapping, editor-config

Configure settings to improve text wrapping and display behavior for better readability

Example: Example (Vim)

```
setlocal linebreak
setlocal nolist
setlocal display+=lastline
```

Example: Example (Neovim)

```
vim.opt.linebreak = true
vim.opt.list = false
vim.opt.display:append('lastline')
```

Source: vim.fandom.com

20.38 Print Syntax Highlighted Buffer in Black and White

Category: display
Tags: printing, syntax-highlighting, color-scheme

Create a mapping to print syntax-highlighted code in black and white, preserving original syntax highlighting while printing

Example: Example (Vim)

```
map <C-p> :color print_bw<CR>:hardcopy<CR>:color sean<CR>:syn on<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-p>', function()
  vim.cmd('color print_bw')
  vim.cmd('hardcopy')
  vim.cmd('color sean')  -- Replace with your color scheme
  vim.cmd('syn on')
end, { desc = 'Print buffer in black and white' })
```

Source: vim.fandom.com

20.39 Print with Line Numbers

Category: display
Tags: printing, configuration

Enable line numbers when printing without modifying the actual file

Example: Example (Vim)

```
:set printoptions=number:y
```

Example: Example (Neovim)

```
vim.opt.printoptions = 'number:y'
```

Source: vim.fandom.com

20.40 Quick Background Color Toggle

Category: display
Tags: color-scheme, ui-customization, theme

Easily switch between light and dark backgrounds with a single keymap

Example: Example (Vim)

```
map <F11> :let &background = ( &background == "dark"? "light" : "dark" )<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F11>', function()
  vim.o.background = vim.o.background == 'dark' and 'light' or 'dark'
end, { desc = 'Toggle background color' })
```

Source: vim.fandom.com

20.41 Quick Syntax Color Customization

Category: display
Tags: syntax-highlighting, color-scheme, customization

Easily modify specific syntax highlight colors directly in your vimrc without editing full color scheme files

Example: Example (Vim)

```
highlight comment ctermfg=lightblue
highlight constant ctermfg=red
```

Example: Example (Neovim)

```
vim.cmd('highlight comment ctermfg=lightblue')
vim.cmd('highlight constant ctermfg=red')
```

Source: vim.fandom.com

20.42 Quickly Customize Syntax Highlighting Colors

Category: display
Tags: syntax-highlighting, color-scheme, customization

Easily modify specific syntax highlighting colors directly in your configuration without editing full color scheme files

Example: Example (Vim)

```
highlight comment ctermfg=lightblue
highlight constant ctermfg=red
```

Example: Example (Neovim)

```
vim.cmd('highlight comment ctermfg=lightblue')
vim.cmd('highlight constant ctermfg=red')
```

Source: vim.fandom.com

20.43 Quickly Highlight Current Line

Category: display
Tags: cursor-highlight, ui-enhancement, visual-guide

Add a simple mapping to highlight the current line, making it easier to track cursor position

Example: Example (Vim)

```
highlight LineTooLong cterm=bold ctermbg=red guibg=LightYellow
match LineTooLong /\%>80v.\+/
```

Example: Example (Neovim)

```
vim.cmd('highlight LineTooLong cterm=bold ctermbg=red guibg=LightYellow')
vim.cmd('match LineTooLong /\%>80v.\+/')
```

Source: vim.fandom.com

20.44 Relative Line Numbers for Easy Navigation

Category: display
Tags: navigation, ui, editing

Show relative line numbers to make movement and editing commands more intuitive

Example: Example (Vim)

```
set relativenumber
set number
```

Example: Example (Neovim)

```
vim.opt.relativenumber = true
vim.opt.number = true
```

Source: vim.fandom.com

20.45 Screen Drawing and Color Output in Vim Scripts

Category: display
Tags: scripting, ui, colors

Techniques for creating colorful screen output and animations in Vim scripts using echo commands and highlighting

Example: Example (Vim)

```
set nomore
let &ch=&lines-1

" Colored text output
echohl None
echon "xxxx"
echohl Visual
echon "yyyy"

while drawing
  redraw!
  " echo some lines
endwhile
let &ch=1
```

Example: Example (Neovim)

```
-- Disable 'more' prompts
vim.o.nomore = true

-- Set command height for drawing
vim.o.cmdheight = vim.o.lines - 1

-- Colored text output example
vim.cmd('echohl None')
vim.cmd('echon "xxxx"')
vim.cmd('echohl Visual')
vim.cmd('echon "yyyy"')

-- Screen drawing technique
for _ in ipairs(drawing_frames) do
  vim.cmd('redraw!')
  -- Draw frame
end
```

```
-- Reset command height
vim.o.cmdheight = 1
```

Source: vim.fandom.com

20.46 Show ASCII/Unicode Value of Character

Category: display
Tags: character-info, unicode, status-line

Display the ASCII or Unicode value of the character under the cursor in decimal, hex, and octal formats

Example: Example (Vim)

```
" Add to statusline to show character values
:set statusline=%<%f%h%m%r%=%b\ 0x%B\ \ %l,%c%V\ %P
:set laststatus=2
```

Example: Example (Neovim)

```
-- Configure status line to show character values
vim.opt.statusline = '%<%f%h%m%r%=%b 0x%B  %l,%c%V %P'
vim.opt.laststatus = 2
```

Source: vim.fandom.com

20.47 Show Whitespace with List Chars

Category: display
Tags: whitespace, visualization

Customize display of whitespace characters to make invisible characters visible

Example: Example (Vim)

```
" Toggle list view and customize list characters
:set list listchars=tab:»·,trail:·,extends:>,precedes:<
```

Example: Example (Neovim)

```
-- Configure list characters in Neovim
vim.opt.list = true
vim.opt.listchars:append({
  tab = '»·',
  trail = '·',
  extends = '>',
```

```
  precedes = '<'
})
```

Source: vim.fandom.com

20.48 Smart Line Number and Column Display

Category: display
Tags: ui, line-numbers, configuration

Customize line number display width and behavior

Example: Example (Vim)

```
set number
set numberwidth=4
```

Example: Example (Neovim)

```
vim.opt.number = true
vim.opt.numberwidth = 4
```

Source: vim.fandom.com

20.49 Soft Line Wrapping

Category: display
Tags: text-display, ui, navigation

Enable soft line wrapping that displays long lines across multiple screen lines without changing the actual text

Example: Example (Vim)

```
set wrap
set linebreak
```

Example: Example (Neovim)

```
vim.opt.wrap = true
vim.opt.linebreak = true

-- Use gj and gk to move by screen lines instead of real lines
vim.keymap.set('n', 'j', 'gj', { noremap = true })
vim.keymap.set('n', 'k', 'gk', { noremap = true })
```

Source: vim.fandom.com

20.50 Toggle Between Light and Dark Color Schemes

Category: display
Tags: color-scheme, key-mapping, theme-switching

Create a function to easily switch between two color scheme styles with a single keypress

Example: Example (Vim)

```
function! s:SwitchPSCStyle()
  if exists('g:psc_style')
    if g:psc_style == 'cool'
      let g:psc_style = 'warm'
    elseif g:psc_style == 'warm'
      let g:psc_style = 'cool'
    endif
  else
    let g:psc_style = 'warm'
  endif
  colorscheme ps_color
endfunction
map <silent> <F6> :call <SID>SwitchPSCStyle()<CR>
```

Example: Example (Neovim)

```
local function switch_color_style()
  local style = vim.g.psc_style

  if style == nil then
    vim.g.psc_style = 'warm'
  elseif style == 'cool' then
    vim.g.psc_style = 'warm'
  elseif style == 'warm' then
    vim.g.psc_style = 'cool'
  end

  vim.cmd('colorscheme ps_color')
end

vim.keymap.set('n', '<F6>', switch_color_style, { silent = true })
```

Source: vim.fandom.com

20.51 Toggle Cursor Line and Column Highlighting

Category: display
Tags: ui, cursor-highlight, visual-enhancement

Easily toggle highlighting of the current line and column to improve cursor visibility

Example: Example (Vim)

```
" Highlight current line and column
:hi CursorLine   cterm=NONE ctermbg=darkred ctermfg=white guibg=darkred
↪  guifg=white
:hi CursorColumn cterm=NONE ctermbg=darkred ctermfg=white guibg=darkred
↪  guifg=white
:nnoremap <Leader>c :set cursorline! cursorcolumn!<CR>
```

Example: Example (Neovim)

```
-- Highlight current line and column
vim.cmd('hi CursorLine cterm=NONE ctermbg=darkred ctermfg=white
↪  guibg=darkred guifg=white')
vim.cmd('hi CursorColumn cterm=NONE ctermbg=darkred ctermfg=white
↪  guibg=darkred guifg=white')
vim.keymap.set('n', '<Leader>c', function()
  vim.o.cursorline = not vim.o.cursorline
  vim.o.cursorcolumn = not vim.o.cursorcolumn
end, { desc = 'Toggle cursor line and column highlight' })
```

Source: vim.fandom.com

20.52 Toggle Diff Highlighting in Merge Conflicts

Category: display
Tags: syntax-highlighting, diff-mode, ui

Add mappings to quickly disable diff syntax highlighting in one window, which can help reduce visual noise during three-way merges

Example: Example (Vim)

```
" Example mapping to toggle diff highlighting
nnoremap <leader>dh :set nodiff<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for toggling diff highlighting
vim.keymap.set('n', '<leader>dh', function()
  vim.wo.diff = not vim.wo.diff
end, { desc = 'Toggle diff highlighting' })
```

Source: vim.fandom.com

20.53 Toggle Tab Visualization Function

Category: display
Tags: toggle, visualization, whitespace

Create a toggle function to switch between showing and hiding tab characters with custom highlighting

Example: Example (Vim)

```
function! SeeTab()
  if !exists("g:SeeTabEnabled")
    let g:SeeTabEnabled = 1
    set list
    set listchars=tab:\|
    highlight SpecialKey guifg=black guibg=magenta
  else
    set nolist
    unlet g:SeeTabEnabled
  endif
endfunction
command! SeeTab call SeeTab()
```

Example: Example (Neovim)

```
local function see_tab()
  if not vim.g.see_tab_enabled then
    vim.g.see_tab_enabled = true
    vim.opt.list = true
    vim.opt.listchars:append({ tab = '| ' })
    vim.cmd('highlight SpecialKey guifg=black guibg=magenta')
  else
    vim.opt.list = false
    vim.g.see_tab_enabled = nil
  end
end

vim.api.nvim_create_user_command('SeeTab', see_tab, {})
```

Source: vim.fandom.com

20.54 Toggle invisible characters

Category: Display
Tags: invisible, characters, toggle

Use `:set list!` to toggle display of invisible characters (tabs, spaces, etc.).

Example: Example (Vim)

```
" Vimscript:
:set list!  " toggle invisible characters
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.list = not vim.opt.list:get()  -- toggle invisible characters
```

Source: Community contributed

20.55 Truncate Long Echoed Messages

Category: display
Tags: messages, truncation, ui

Automatically shorten long messages in the command line to fit within the screen width

Example: Example (Vim)

```
set shortmess+=T

" Function to safely echo truncated messages
function! ShortEcho(msg)
  let saved=&shortmess
  set shortmess+=T
  exe "norm :echomsg " . a:msg . "\n"
  let &shortmess=saved
endfunction
```

Example: Example (Neovim)

```
-- Enable message truncation
vim.opt.shortmess:append('T')

-- Function to safely echo truncated messages
function _G.short_echo(msg)
  local saved_shortmess = vim.o.shortmess
  vim.o.shortmess = vim.o.shortmess .. 'T'
  vim.cmd.norm({ args = {':echomsg ' .. msg .. '\n'} })
  vim.o.shortmess = saved_shortmess
end
```

Source: vim.fandom.com

20.56 View and Test Available Colors in Vim/Neovim

Category: display
Tags: colors, colorscheme, testing

Script to generate a comprehensive view of available colors, helping users explore and select color schemes interactively

Example: Example (Vim)

```
" Script to display all named colors in rgb.txt
new
setlocal buftype=nofile bufhidden=hide noswapfile
0read $VIMRUNTIME/rgb.txt
" Filter and process color names
silent execute 'v/'.find_color.'/d'
silent g/grey/d
```

Example: Example (Neovim)

```
-- Lua equivalent for color exploration
local function show_colors()
  local colors = vim.fn.readfile(vim.fn.expand('$VIMRUNTIME/rgb.txt'))
  local filtered_colors = vim.tbl_filter(function(color)
    return color:match('^%s*%d+%s*%d+%s*%d+%s*%w+$') and not
    ↪  color:match('grey')
  end, colors)

  local buf = vim.api.nvim_create_buf(true, true)
  vim.api.nvim_buf_set_lines(buf, 0, -1, false, filtered_colors)
  vim.api.nvim_set_current_buf(buf)
end

-- Call with :lua ShowColors()
```

Source: vim.fandom.com

20.57 Visualize Tab Characters in Vim

Category: display
Tags: whitespace, formatting, visual-guide

Show tab characters as vertical lines or symbols to help understand indentation and whitespace

Example: Example (Vim)

```
" Show tabs as vertical lines
set list
set listchars=tab:\|\

" Alternative for newer Vim versions
set listchars=tab:>-
```

Example: Example (Neovim)

```
-- Show tabs as vertical lines
vim.opt.list = true
```

```
vim.opt.listchars:append({ tab = '| ' })

-- Alternative style
vim.opt.listchars:append({ tab = '>-' })
```

Source: vim.fandom.com

20.58 Visually Highlight Unwanted Whitespace

Category: display
Tags: whitespace, visual-cues, formatting

Configure Vim to display whitespace characters, making trailing or problematic spaces visible

Example: Example (Vim)

```
set list listchars=tab:»·,trail:·
" Alternative style
set list lcs=tab:·⎕,trail:¶
```

Example: Example (Neovim)

```
vim.opt.list = true
vim.opt.listchars:append({
  tab = '»·',
  trail = '·'
})

-- Alternative style
vim.opt.listchars:append({
  tab = '·⎕',
  trail = '¶'
})
```

Source: vim.fandom.com

CHAPTER 21

Edit

21.1 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

21.2 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: Example (Vim)

```
dw    " delete word
c3j   " change 3 lines down
y$    " yank to end of line
>i{   " indent inside braces
=ap   " format around paragraph
```

21.3 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: Example (Vim)

```
d<Esc>          " abandon delete operation
c<Backspace>    " cancel change operation
```

21.4 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: Example (Vim)

```
dvw   " force characterwise delete word
dV    " force linewise delete (whole line)
dCtrl+v}  " force blockwise delete to closing brace
```

21.5 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: Example (Vim)

```
d    " delete operator (enters pending mode)
dw   " delete word (operator + motion)
ci(  " change inside parentheses (operator + text object)
```

21.6 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: Example (Vim)

```
ciw   " change inside word
da(   " delete around parentheses
yi"   " yank inside quotes
>ap   " indent around paragraph
=i{   " format inside braces
```

21.7 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: Example (Vim)

```
Ctrl+l  " redraw screen
```

21.8 Repeat last change

Category: Edit
Tags: repeat, change, command

Use `.` (dot) to repeat the last change command.

Example: Example (Vim)

```
.  " repeat last change
```

21.9 Show file information

Category: Edit
Tags: file, info, position, status

Use `Ctrl+g` to display current file name, cursor position, and buffer information.

Example: Example (Vim)

```
Ctrl+g  " show file info and cursor position
```

21.10 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: Example (Vim)

```
s   " substitute character under cursor
S   " substitute entire line
5s  " substitute 5 characters
```

21.11 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: Example (Vim)

```
:earlier 10m  " revert to 10 minutes ago
:later 5m     " go forward 5 minutes
```

CHAPTER 22

Editing

22.1 Add Unicode Combining Characters to Text

Category: editing
Tags: unicode, text-formatting, advanced-editing

Add underlines, overlines, and strikethroughs using Unicode combining characters across selected text

Example: Example (Vim)

```
" Define commands for adding combining characters
command! -range -nargs=0 Overline        call s:CombineSelection(<line1>,
↪  <line2>, '0305')
command! -range -nargs=0 Underline       call s:CombineSelection(<line1>,
↪  <line2>, '0332')
command! -range -nargs=0 DoubleUnderline call s:CombineSelection(<line1>,
↪  <line2>, '0333')
command! -range -nargs=0 Strikethrough   call s:CombineSelection(<line1>,
↪  <line2>, '0336')

function! s:CombineSelection(line1, line2, cp)
  execute 'let char = "\u'.a:cp.'"'
  execute a:line1.','.a:line2.'s/\%V[^[:cntrl:]]/&'.char.'/ge'
endfunction

" Optional mapping for quick access
vnoremap OO :Overline<CR>
```

Example: Example (Neovim)

```
-- Unicode combining character utility functions
local function combine_selection(line1, line2, cp)
  vim.cmd(string.format(
    '%d,%ds/\%%V[^[:cntrl:]]/&\u%s/ge',
    line1, line2, cp
  ))
end

-- Create commands for different text decorations
vim.api.nvim_create_user_command('Overline', function()
  combine_selection(vim.fn.line("'<"), vim.fn.line("'>"), '0305')
```

```
end, { range = true })

vim.api.nvim_create_user_command('Underline', function()
  combine_selection(vim.fn.line("'<"), vim.fn.line("'>"), '0332')
end, { range = true })

-- Optional keymapping
vim.keymap.set('v', 'OO', ':Overline<CR>', { desc = 'Apply overline to
↪  selected text' })
```

Source: vim.fandom.com

22.2 Advanced Word Replacement Techniques

Category: editing
Tags: registers, replace, key-mapping

Multiple methods to replace words or text using different Vim registers and techniques, with easy repeatability

Example: Example (Vim)

```
" Remap paste in visual mode to preserve original register
xnoremap <silent> p p:let @"=@0<CR>

" Stamping mapping to replace current word
nnoremap S diw"0P
```

Example: Example (Neovim)

```
-- Preserve register during visual paste
vim.keymap.set('x', 'p', function()
  vim.fn.feedkeys('p', 'n')
  vim.fn.setreg('"', vim.fn.getreg('0'))
end, { desc = 'Paste without overwriting register' })

-- Stamp mapping to replace word with last yanked text
vim.keymap.set('n', 'S', '"_diw"0P', { desc = 'Replace word with last yank'
↪  })
```

Source: vim.fandom.com

22.3 Alternative Word Deletion Methods

Category: editing
Tags: text-editing, word-deletion

Use different word deletion commands for more precise text removal

Example: Example (Vim)

```
" Delete a WORD (includes special characters)
dW

" Delete up to (not including) a character
dt<char>
```

Example: Example (Neovim)

```
-- These are standard Vim motions that work identically in Neovim
-- No specific Lua translation required
```

Source: vim.fandom.com

22.4 Auto Spelling Correction with Abbreviations

Category: editing
Tags: spelling, autocorrect, abbreviations

Automatically correct common misspellings and typos using Vim's abbreviation feature

Example: Example (Vim)

```
iabbrev mispell misspell
iabbrev funciton function
iabbrev functiton function
iabbrev fucntion function
iabbrev funtion function
iabbrev erturn return
iabbrev retunr return
iabbrev reutrn return
iabbrev reutn return
```

Example: Example (Neovim)

```
-- Create abbreviations in Lua
vim.cmd([[iabbrev mispell misspell
iabbrev funciton function
iabbrev functiton function
iabbrev fucntion function
iabbrev funtion function
iabbrev erturn return
iabbrev retunr return
iabbrev reutrn return
iabbrev reutn return]])
```

Source: vim.fandom.com

22.5 Auto-Close HTML Tags with Omni Completion

Category: editing
Tags: html, completion, auto-insert

Automatically complete closing HTML tags using Vim's omni completion in insert mode

Example: Example (Vim)

```
" Abbreviation to auto-close HTML tags
:iabbrev <// </<C-X><C-O>

" Remap Ctrl-Space for tag completion
:imap <C-Space> <C-X><C-O>
```

Example: Example (Neovim)

```
-- Lua equivalent for auto-closing HTML tags
vim.api.nvim_create_autocmd({'BufRead', 'BufNewFile'}, {
  pattern = {'*.html', '*.js', '*.xml'},
  callback = function()
    vim.keymap.set('i', '>', function()
      return '>' .. '</>' .. vim.fn.complete(vim.fn.col('.'),
      ↪  vim.fn['compl#omnifunc']())
    end, { buffer = true, expr = true })
  end
})
```

Source: vim.fandom.com

22.6 Autocomplete Words in Insert Mode

Category: editing
Tags: insert-mode, completion, productivity

Use Ctrl-N and Ctrl-P to autocomplete words from the current file

Example: Example (Vim)

```
" In insert mode
<C-N>  " Complete word forward
<C-P>  " Complete word backward
```

Example: Example (Neovim)

```
-- Built-in completion
-- Configure with:
vim.opt.complete:append('k')
```

Source: vim.fandom.com

22.7 Block Commenting in Lisp

Category: editing
Tags: commenting, block-edit

Quick way to comment out an entire code block between parentheses

Example: Example (Vim)

```
map .; v%:s/^/;;;/<CR>:noh<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '.;', function()
  vim.cmd('normal! v%:s/^/;;;/\<CR>:nohl\<CR>')
end, { desc = 'Comment out code block' })
```

Source: vim.fandom.com

22.8 Block Line Sorting with Global Command

Category: editing
Tags: sorting, text-manipulation, global-command

Sort each block of lines in a file, separating blocks by blank lines

Example: Example (Vim)

```
":g/^\s*$/;//-1sort"
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.cmd
vim.cmd('g/^\s*$/;//-1sort')
```

Source: vim.fandom.com

22.9 Calculate expressions

Category: Editing
Tags: calculate, math, expression, replace

Use `<C-r>=` in insert mode to calculate mathematical expressions and insert the result.

Example: Example (Vim)

```
" In insert mode:
<C-r>=2+2<CR>       " inserts '4'
<C-r>=16*1024<CR>   " inserts '16384'
```

Source: Community contributed

22.10 Case Modification Techniques

Category: editing
Tags: text-case, regex-substitution

Use \U, \L, \u, and \l for comprehensive case transformation in regex substitutions

Example: Example (Vim)

```
" Convert to uppercase with termination
:s/\(test\)/\U\1\e file/

" Uppercase first character
:s/\(test\)/\u\1/
```

Example: Example (Neovim)

```
-- Similar transformations can be done using vim.cmd()
-- Note: Lua will require escaping backslashes
vim.cmd(':s/\\(test\\)/\\U\\1\\e file/')
vim.cmd(':s/\\(test\\)/\\u\\1/')
```

Source: vim.fandom.com

22.11 Comment Code with Flexible Delimiters

Category: editing
Tags: commenting, search-replace, code-editing

Use alternative delimiters to easily comment out code ranges with less escaping

Example: Example (Vim)

```
:5,8 s#^#//#
```

Example: Example (Neovim)

```
-- Comment lines 5-8 with // in Neovim
vim.cmd('5,8 s#^#//#')
```

Source: vim.fandom.com

22.12 Compact Multiple Blank Lines

Category: editing
Tags: text-cleanup, whitespace, file-manipulation

Remove consecutive blank lines, replacing multiple empty lines with a single blank line

Example: Example (Vim)

```
" Compress multiple blank lines to single line
:v/./,/./-j

" Alternative method
:%s/^$\n^$//g
```

Example: Example (Neovim)

```
-- Compress blank lines
vim.cmd('v/./,/./-j')

-- Alternative substitution method
vim.cmd('%s/^$\\n^$//g')
```

Source: vim.fandom.com

22.13 Condense Multiple Blank Lines

Category: editing
Tags: text-cleanup, search-replace

Reduce multiple consecutive blank lines to a single blank line while removing trailing whitespace

Example: Example (Vim)

```
" Remove trailing whitespace and condense blank lines
:%s/\s\+$//e
:%s/\n\{3,}/\r\r/e
```

Example: Example (Neovim)

```
-- Remove trailing whitespace and condense blank lines
vim.cmd('%s/\s\+$//e')
vim.cmd('%s/\n\{3,}/\r\r/e')
```

Source: vim.fandom.com

22.14 Consistent Yank to End of Line

Category: editing
Tags: yank, text-objects, consistency

Makes 'Y' behave consistently with 'D' and 'C' by yanking to end of line

Example: Example (Vim)

```
nnoremap Y y$
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'Y', 'y$', { desc = 'Yank to end of line' })
```

Source: vim.fandom.com

22.15 Convert Between CamelCase and snake_case

Category: editing
Tags: text-transformation, case-conversion, regex

Provides flexible regex-based substitution commands to convert variable naming styles between CamelCase and snake_case, useful for code refactoring and consistent naming

Example: Example (Vim)

```
" Convert snake_case to CamelCase
:s#_\([a-z]\)#\u\1#g

" Convert CamelCase to snake_case
:s#\C\(\<\u[a-z0-9]\+\|[a-z0-9]\+\)\(\u\)#\l\1_\l\2#g
```

Example: Example (Neovim)

```
-- Lua function to convert snake_case to CamelCase
function convert_to_camel_case(str)
  return str:gsub('_(%a)', function(letter)
    return letter:upper()
  end)
end

-- Lua function to convert CamelCase to snake_case
function convert_to_snake_case(str)
  return str:gsub('(%u)', function(letter)
    return '_' .. letter:lower()
  end):lower()
end
```

Source: vim.fandom.com

22.16 Convert Enum to String Table Automatically

Category: editing
Tags: c-programming, code-generation, transformation

Quickly generate a string representation for enum values, useful for debugging and logging

Example: Example (Vim)

```
function! Enum2Array()
    exe "normal! :'<,'>g/^\s*$/d\n"
    exe "normal! :'<,'>s/\(\s*\)\([[:alnum:]_]*\).*/\1[\2] = \"\2\",/\n"
    normal `>
    exe "normal a\n};\n"
    normal `<
    exe "normal iconst char *[] =\n{\n"
    exe ":'<,'>normal ==" " try some indentation
    normal `< " set the cursor at the top
endfunction
```

Example: Example (Neovim)

```
function _G.enum_to_array()
    local start_line = vim.fn.line("'<")
    local end_line = vim.fn.line("'>")

    -- Remove empty lines
    vim.cmd(start_line .. ',' .. end_line .. 'g/^\s*$/d')

    -- Transform enum entries to string array
    vim.cmd(start_line .. ',' .. end_line .. 's/\s*\(\w\+\)/[\1] = "\1",/')

    -- Add array declaration and closing brace
    vim.cmd('normal! O const char *[] = {\<Esc>')
    vim.cmd('normal! o};\<Esc>')

    -- Optionally reindent
    vim.cmd(start_line .. ',' .. end_line .. 'normal! ==')
end

-- Usage: Select enum lines and call the function
vim.keymap.set('x', '<leader>ce', _G.enum_to_array, { desc = 'Convert Enum
↪  to String Array' })
```

Source: vim.fandom.com

22.17 Convert Variable Naming Styles

Category: editing
Tags: text-transformation, variable-naming, refactoring

Easily convert between camelCase, snake_case, and UPPER_CASE variable naming conventions

Example: Example (Vim)

```
" Convert snake_case to CamelCase
:s#_\(\l\)#\u\1#g

" Convert CamelCase to snake_case
:s#\C\(\<\u[a-z0-9]\+\|[a-z0-9]\+\)\(\u\)#\l\1_\l\2#g
```

Example: Example (Neovim)

```
-- Convert snake_case to CamelCase
vim.cmd('s#_\(\l\)#\u\1#g')

-- Convert CamelCase to snake_case
vim.cmd('s#\C\(\<\u[a-z0-9]\+\|[a-z0-9]\+\)\(\u\)#\l\1_\l\2#g')
```

Source: vim.fandom.com

22.18 Copy and Replace Lines Across Document

Category: editing
Tags: line-manipulation, copy, paste

Quickly copy a line and paste it to multiple locations with minimal keystrokes

Example: Example (Vim)

```
# Copy line: Y
# Paste after/before: p/P
# Alternative replace method:
# 0y$ to copy line content
# S<C-R>0<Esc> to replace line
```

Example: Example (Neovim)

```
-- Copy line: yy or Y
-- Paste after/before: p/P
-- Complex replace:
-- 0y$ to copy line content
-- S to replace line content
```

Source: vim.fandom.com

22.19 Copy and Replace Words Across File

Category: editing
Tags: registers, text-manipulation, copy-paste

Efficiently copy a word and replace multiple occurrences using registers and repeat commands

Example: Example (Vim)

```
" To copy and replace words:
yiw   " Yank current word
ciw<C-R>0<Esc>  " Change word with yanked text
.     " Repeat the replacement
```

Example: Example (Neovim)

```
-- Lua equivalent for copying and replacing words
-- yiw to yank word
-- ciw<C-R>0<Esc> to replace
-- . to repeat
```

Source: vim.fandom.com

22.20 Copy and Replace Words Across Multiple Locations

Category: editing
Tags: copy, replace, registers

Efficiently copy a word and replace words in multiple locations using yank and repeat commands

Example: Example (Vim)

```
# Workflow:
# 1. yiw - yank current word
# 2. Move cursor
# 3. ciw<C-R>0<Esc> - change word with yanked text
# 4. Use '.' to repeat
```

Example: Example (Neovim)

```
-- Similar workflow can be achieved
-- Use vim.fn.feedkeys() for complex macros if needed
-- Mostly relies on standard Vim/Neovim keybindings
```

Source: vim.fandom.com

22.21 Copy and Replace Words Efficiently

Category: editing
Tags: registers, text-manipulation, copy-paste

Quickly copy a word and replace multiple occurrences using yanking and dot command repetition

Example: Example (Vim)

```
" Workflow:
" yiw - yank current word
" ciw<C-R>0<Esc> - change word with yanked text
" . - repeat the replacement
```

Example: Example (Neovim)

```
-- Similar workflow in Neovim
-- Use vim.fn.feedkeys() for complex macros if needed
-- Essentially the same key sequence works
```

Source: vim.fandom.com

22.22 Copy and Replace Words Quickly

Category: editing
Tags: registers, text-manipulation, copy-paste

Efficiently copy a word and replace multiple words with the copied text using yanking and register manipulation

Example: Example (Vim)

```
# Copy current word: yiw
# Replace at new location: ciw<C-R>0<Esc>
# Repeat with '.'
```

Example: Example (Neovim)

```
-- Lua equivalent for word copying and replacing
-- First copy word: vim.cmd('yiw')
-- Replace word: vim.cmd('ciw' .. vim.fn.getreg('0') .. '\<Esc>')
-- Repeat with '.'
```

Source: vim.fandom.com

22.23 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: Example (Vim)

```
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
```

Source: Community contributed

22.24 Delete Lines Matching a Pattern

Category: editing
Tags: search-replace, text-manipulation, global-command

Use :g command to delete lines matching or not matching a specific pattern globally in the file

Example: Example (Vim)

```
" Delete all lines containing 'profile'
:g/profile/d

" Delete empty or whitespace-only lines
:g/^\s*$/d

" Delete all lines not containing comments in Vim script
:g!/^\s*"/d
```

Example: Example (Neovim)

```
-- Delete lines containing 'profile'
vim.cmd('g/profile/d')

-- Delete empty or whitespace-only lines
vim.cmd('g/^\s*$/d')

-- Delete lines not containing comments in Vim script
vim.cmd('g!/^\s*"/d')
```

Source: vim.fandom.com

22.25 Delete XML/HTML Tags Efficiently

Category: editing
Tags: xml, html, text-objects, editing

Quickly delete a pair of XML or HTML tags using text objects, which works for both small and large tag blocks

Example: Example (Vim)

```
vat<Esc>`<df>`>F<df>
```

Example: Example (Neovim)

```
-- Delete around tag (at text object)
-- Requires manual mapping in Lua
vim.keymap.set('n', '<leader>dt', function()
  vim.cmd('normal! vat<Esc>`<df>`>F<df>')
end, { desc = 'Delete XML/HTML tag pair' })
```

Source: vim.fandom.com

22.26 Delete to Custom Delimiter Efficiently

Category: editing
Tags: text-editing, motion, deletion

Delete text up to a specific character, which is more flexible than standard word deletion

Example: Example (Vim)

```
" Delete to space
df

" Delete to period (end of sentence)
df.
```

Example: Example (Neovim)

```
-- Lua equivalents for custom delimiter deletion
-- Use native Vim motions, so no specific Lua translation needed
-- Can use 'df ' or 'df.' directly in Neovim
```

Source: vim.fandom.com

22.27 Dictionary-Based Word Completion

Category: editing
Tags: completion, insert-mode, text-editing

Enable dictionary-based word completion to quickly insert words from a predefined dictionary, useful for writing and coding

Example: Example (Vim)

```
" Set dictionary location
set dictionary+=/usr/share/dict/words

" Enable dictionary completion
set complete+=k

" Optional: Map F12 for dictionary completion
:inoremap <F12> <C-X><C-K>
```

Example: Example (Neovim)

```
-- Set dictionary location
vim.opt.dictionary:append('/usr/share/dict/words')

-- Enable dictionary completion
vim.opt.complete:append('k')

-- Optional: Map F12 for dictionary completion
vim.keymap.set('i', '<F12>', '<C-X><C-K>', { desc = 'Dictionary word
↪  completion' })
```

Source: vim.fandom.com

22.28 Duplicate Previous Line Word by Word

Category: editing
Tags: key-mapping, text-manipulation, productivity

Quickly duplicate words from the previous line, useful for coding and documentation

Example: Example (Vim)

```
" Duplicate previous line word by word
imap <F1> @<Esc>kyWjPA<BS>
nmap <F1> @<Esc>kyWjPA<BS>
imap <F2> <Esc>o<Esc>kyWjPA<BS><Space>
nmap <F2> <Esc>o<Esc>kyWjPA<BS><Space>
```

Example: Example (Neovim)

```
-- Lua equivalent for duplicating words from previous line
vim.keymap.set({'n', 'i'}, '<F1>', function()
  local line = vim.fn.getline(vim.fn.line('.') - 1)
  local word = vim.fn.matchstr(line, '\k\+')
  vim.api.nvim_put({word}, 'c', true, true)
end, { desc = 'Duplicate word from previous line' })

-- Alternative approach using built-in Neovim mapping style
vim.keymap.set({'n', 'i'}, '<F2>', function()
  vim.cmd('normal! o')
  local line = vim.fn.getline(vim.fn.line('.') - 2)
  local word = vim.fn.matchstr(line, '\k\+')
  vim.api.nvim_put({word}, 'c', true, true)
end, { desc = 'Duplicate word from previous line and create new line' })
```

Source: vim.fandom.com

22.29 Easy Character and Word Swapping

Category: editing
Tags: text-manipulation, quick-edit, normal-mode

Quick methods to swap characters and words in normal mode without moving the cursor

Example: Example (Vim)

```
" Swap current character with next
xp

" Swap current word with next
:nnoremap <silent> gw
 ↪  "_yiw:s/\(\%#\w\+\)\(\W\+\)\(\w\+\)/\3\2\1/<CR><c-o><c-l>:nohlsearch<CR>
```

Example: Example (Neovim)

```
-- Swap current character with next
vim.keymap.set('n', 'gc', 'xph', { desc = 'Swap current character' })

-- Swap current word with next
vim.keymap.set('n', 'gw', function()
  local current_word = vim.fn.expand('<cword>')
  local next_word = vim.fn.expand('<cWORD>')
  local line = vim.api.nvim_get_current_line()
  local new_line = line:gsub(vim.fn.escape(current_word, '%'), next_word, 1)
  new_line = new_line:gsub(vim.fn.escape(next_word, '%'), current_word, 1)
  vim.api.nvim_set_current_line(new_line)
end, { desc = 'Swap current word with next' })
```

Source: vim.fandom.com

22.30 Easy Comment/Uncomment for Multiple Languages

Category: editing
Tags: commenting, text-manipulation, multi-language

Dynamically set comment leaders for different file types, with easy comment/uncomment mappings

Example: Example (Vim)

```
au FileType haskell,vhdl,ada let b:comment_leader = '-- '
au FileType vim let b:comment_leader = '" '
au FileType c,cpp,java let b:comment_leader = '// '
au FileType sh,make let b:comment_leader = '# '
au FileType tex let b:comment_leader = '% '

noremap <silent> ,c :<C-B>sil <C-E>s/^/<C-R>=escape(b:comment_leader,'\/')
 ↪  <CR>/<CR>:noh<CR>
noremap <silent> ,u :<C-B>sil <C-E>s/^\V<C-R>=escape(b:comment_leader,'\/')
 ↪  <CR>//e<CR>:noh<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = {'haskell', 'vhdl', 'ada'},
  callback = function() vim.b.comment_leader = '-- ' end
})

vim.api.nvim_create_autocmd('FileType', {
  pattern = {'vim'},
  callback = function() vim.b.comment_leader = '" ' end
})

-- Similar autocmds for other filetypes

-- Comment mapping
vim.keymap.set('n', ',c', function()
  local comment_leader = vim.b.comment_leader or '// '
  vim.cmd('silent s/^/' .. vim.fn.escape(comment_leader, '\/') .. '/')
  vim.cmd('nohlsearch')
end)

-- Uncomment mapping
vim.keymap.set('n', ',u', function()
  local comment_leader = vim.b.comment_leader or '// '
  vim.cmd('silent s/^\\V' .. vim.fn.escape(comment_leader, '\/') .. '//')
  vim.cmd('nohlsearch')
end)
```

Source: vim.fandom.com

22.31 Easy Word Replacement Across Files

Category: editing
Tags: registers, text-objects, search-replace

Replace words efficiently using registers and text objects, with repeatable operations across multiple locations

Example: Example (Vim)

```
" Stamp mapping to replace words with last yanked text
nnoremap S diw"0P

" Remap paste in visual mode to preserve register
xnoremap <silent> p p:let @"=@0<CR>
```

Example: Example (Neovim)

```
-- Stamp mapping to replace words with last yanked text
vim.keymap.set('n', 'S', 'diw"0P', { desc = 'Replace word with last yanked
↪  text' })

-- Remap paste in visual mode to preserve register
vim.keymap.set('x', 'p', 'p<Cmd>let @"=@0<CR>', { desc = 'Paste and preserve
↪  register' })
```

Source: vim.fandom.com

22.32 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: Example (Vim)

```
: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
```

Source: Community contributed

22.33 Efficient Text Completion and Navigation

Category: editing
Tags: autocomplete, navigation, productivity

Use Vim's built-in word completion and home row navigation to speed up editing

Example: Example (Vim)

```
# Word completion
Ctrl-P  # Complete previous word
Ctrl-N  # Complete next word
```

Example: Example (Neovim)

```
-- Word completion mappings
vim.keymap.set('i', '<C-p>', '<C-x><C-p>', { desc = 'Complete previous word'
↪  })
vim.keymap.set('i', '<C-n>', '<C-x><C-n>', { desc = 'Complete next word' })
```

Source: vim.fandom.com

22.34 Efficient Undo and Redo

Category: editing
Tags: undo, redo, history

Quickly undo and redo changes with simple commands

Example: Example (Vim)

```
u       # Undo last change
Ctrl-r  # Redo last undone change
```

Example: Example (Neovim)

```
-- Set up undo/redo keymaps
vim.keymap.set('n', '<leader>u', 'u', { desc = 'Undo last change' })
vim.keymap.set('n', '<leader>r', '<C-r>', { desc = 'Redo last change' })
```

Source: vim.fandom.com

22.35 Enter Characters by Unicode Value

Category: editing
Tags: unicode, character-input, encoding

Insert characters by their decimal, octal, hex, or Unicode value in insert mode

Example: Example (Vim)

```
" Insert characters by value in insert mode
" Decimal: Ctrl-V followed by 3-digit number
" Hex Unicode: Ctrl-V u1234
" Full Unicode: Ctrl-V U00001234
```

Example: Example (Neovim)

```
-- For full Unicode support, ensure encoding is set to UTF-8
vim.opt.encoding = 'utf-8'

-- Character input methods remain similar in insert mode
-- Requires knowing the specific Unicode code point
```

Source: vim.fandom.com

22.36 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: Example (Vim)

```
:join          " join current line with next
:j             " short form of join
:5,8join       " join lines 5-8
:join!         " join without inserting spaces
```

Source: Community contributed

22.37 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: Example (Vim)

```
: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
```

Source: Community contributed

22.38 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: Example (Vim)

```
: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
```

Source: Community contributed

22.39 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: Example (Vim)

```
:%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
```

Source: Community contributed

22.40 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: Example (Vim)

```
: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
```

Source: Community contributed

22.41 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: Example (Vim)

```
" 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
```

Source: Community contributed

22.42 Expand Abbreviations Without Extra Characters

Category: editing
Tags: insert-mode, abbreviations, text-expansion

Use Ctrl-] to expand an abbreviation without inserting an extra space

Example: Example (Vim)

```
:iabbrev hh hello
" hh<Ctrl-]> will expand to just 'hello'
```

Example: Example (Neovim)

```
-- Use vim.cmd to set abbreviation
vim.cmd('iabbrev hh hello')
-- Use Ctrl-] in insert mode to expand without space
```

Source: vim.fandom.com

22.43 Extend Word Completion with Precision

Category: editing
Tags: autocomplete, insert-mode, advanced-editing

Incrementally complete words by repeatedly pressing Ctrl-X Ctrl-P to refine matches

Example: Example (Vim)

```
" In insert mode, use Ctrl-X Ctrl-P to extend word matches
```

Example: Example (Neovim)

```
-- Lua doesn't change the core Vim completion behavior
-- Use Ctrl-X Ctrl-P in insert mode to extend matches
-- Customize 'complete' option for broader matching
vim.opt.complete:append('i') -- include current and included files
```

Source: vim.fandom.com

22.44 Fast Print Statement Generation

Category: editing
Tags: debugging, code-generation, productivity

Quickly insert print/debug statements for C/C++ languages

Example: Example (Vim)

```
" Shortcut to insert print statement
imap <F3> printf("DEBUG: ");<Left><Left>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<F3>', 'printf("DEBUG: ");<Left><Left>', { desc =
↪  'Insert print statement' })
```

Source: vim.fandom.com

22.45 Filter Text Using External Shell Commands

Category: editing
Tags: text-processing, shell-filter, visual-mode

Use Vim's filter functionality to process selected text through shell commands, enabling powerful text transformations

Example: Example (Vim)

```
" Example filtering workflow
" 1. Select lines in visual mode
" 2. Press ! to filter
" 3. Enter command like 'cut -f2-3 -d,'
```

Example: Example (Neovim)

```
-- Lua note: Native Vim filtering works the same in Neovim
-- Use visual mode selection + ! to filter text
-- Example commands remain consistent with Vim
```

Source: vim.fandom.com

22.46 Fix Accidental 4 Instead of $ in PHP

Category: editing
Tags: php, key-mapping, error-correction

Quickly replace accidentally typed '4' with '$' in PHP variable names

Example: Example (Vim)

```
nnoremap <Leader>4 m`F4r$``
inoremap <Leader>4 <Esc>m`F4r$``a
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>4', 'm`F4r$``', { desc = 'Replace 4 with $ in
↪  PHP variable' })
vim.keymap.set('i', '<Leader>4', '<Esc>m`F4r$``a', { desc = 'Replace 4 with
↪  $ in PHP variable' })
```

Source: vim.fandom.com

22.47 Fix Numeric Prefix Mistake in PHP Variables

Category: editing
Tags: php, error-correction, key-mapping

Quickly replace an accidental numeric prefix with $ in PHP variables

Example: Example (Vim)

```
nnoremap <Leader>4 m`F4r$``
inoremap <Leader>4 <Esc>m`F4r$``a
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>4', function()
  vim.cmd('normal! m`F4r$``')
end, { desc = 'Replace 4 with $ in PHP variable' })
vim.keymap.set('i', '<Leader>4', '<Esc>m`F4r$``a', { desc = 'Replace 4 with
↪  $ in PHP variable' })
```

Source: vim.fandom.com

22.48 Flexible Case Conversion in Text

Category: editing
Tags: text-manipulation, case-conversion, text-editing

Provides multiple ways to change text case, including toggling, uppercase, and lowercase conversions using movement commands or visual selection

Example: Example (Vim)

```
" Toggle case of characters
g~3w  " Toggle case of next 3 words
g~iw  " Toggle case of current word
gUU   " Convert current line to uppercase
guu   " Convert current line to lowercase
```

Example: Example (Neovim)

```
-- Toggle case of characters
-- Lua uses same Vim commands, but can be mapped with vim.keymap
vim.keymap.set('n', 'g~3w', 'g~3w', { desc = 'Toggle case of next 3 words'
↪  })
vim.keymap.set('n', 'g~iw', 'g~iw', { desc = 'Toggle case of current word'
↪  })
vim.keymap.set('n', 'gUU', 'gUU', { desc = 'Convert line to uppercase' })
vim.keymap.set('n', 'guu', 'guu', { desc = 'Convert line to lowercase' })
```

Source: vim.fandom.com

22.49 Flexible Case Conversion with Visual Selection

Category: editing
Tags: text-transformation, visual-mode, custom-commands

Create custom commands to convert case within visual selections or entire lines

Example: Example (Vim)

```
function! s:Camelize(range) abort
  if a:range == 0
    s#\(\%\(\<\l\+\)\%\(_\)\@=\)\|_\(\l\)#\u\1\2#g
  else
    s#\%V\(\%\(\<\l\+\)\%\(_\)\@=\)\|_\(\l\)\%V#\u\1\2#g
  endif
endfunction

command! -range CamelCase silent! call <SID>Camelize(<range>)
```

Example: Example (Neovim)

```
local function camelize(range)
  if range == 0 then
    vim.cmd('s#\\(\\%\\(\\<\\l\\+\\)\\%\\(_\\)\\@=\\)\\|_\\(\\l\\)#\\u\1\2#
    ↪ g')
  else
    vim.cmd('s#\\%V\\(\\%\\(\\<\\l\\+\\)\\%\\(_\\)\\@=\\)\\|_\\(\\l\\)\\%V#
    ↪ \\u\1\2#g')
  end
end

vim.api.nvim_create_user_command('CamelCase', function(opts)
  camelize(opts.range)
end, { range = true })
```

Source: vim.fandom.com

22.50 Generate Numbered Lists Quickly

Category: editing
Tags: list-generation, range, formatting

Easily insert a list of incrementing numbers with Vim's range() function, which can be formatted and customized

Example: Example (Vim)

```
:put =range(11,15)

" Formatted example
:put =map(range(1,150), 'printf(''%04d'', v:val)')
```

Example: Example (Neovim)

```
-- Insert range of numbers
vim.cmd.put(vim.fn.range(11, 15))

-- Formatted numbers with leading zeros
```

```
vim.cmd.put(vim.fn.map(vim.fn.range(1, 150), function(_, val)
  return string.format('%04d', val)
end))
```

Source: vim.fandom.com

22.51 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: Example (Vim)

```
: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'
```

Source: Community contributed

22.52 HTML Entity Conversion Utility

Category: editing
Tags: html, text-processing, search-replace

Easily escape or unescape HTML entities like <, >, & in text, with support for whole file or specific ranges

Example: Example (Vim)

```
function! HtmlEntities(line1, line2, action)
  let search = @/
  let range = 'silent ' . a:line1 . ',' . a:line2
  if a:action == 0  " must convert &amp; last
    execute range . 'sno/&lt;/</eg'
    execute range . 'sno/&gt;/>/eg'
    execute range . 'sno/&amp;/&/eg'
  else              " must convert & first
    execute range . 'sno/&/&amp;/eg'
    execute range . 'sno/</&lt;/eg'
    execute range . 'sno/>/&gt;/eg'
  endif
  nohl
  let @/ = search
endfunction
```

```
command! -range -nargs=1 Entities call HtmlEntities(<line1>, <line2>,
↪  <args>)
noremap <silent> <Leader>h :Entities 0<CR>
noremap <silent> <Leader>H :Entities 1<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Entities', function(opts)
  local start_line = opts.line1
  local end_line = opts.line2
  local action = tonumber(opts.args)

  local function html_entities(action)
    local view = vim.fn.winsaveview()
    local search = vim.fn.getreg('/')

    if action == 0 then  -- Unescape
      vim.cmd(start_line .. ',' .. end_line .. 's/&lt;/</eg')
      vim.cmd(start_line .. ',' .. end_line .. 's/&gt;/>/eg')
      vim.cmd(start_line .. ',' .. end_line .. 's/&amp;/&/eg')
    else  -- Escape
      vim.cmd(start_line .. ',' .. end_line .. 's/&/\&amp;/eg')
      vim.cmd(start_line .. ',' .. end_line .. 's/</\&lt;/eg')
      vim.cmd(start_line .. ',' .. end_line .. 's/>/\&gt;/eg')
    end

    vim.fn.winrestview(view)
    vim.fn.setreg('/', search)
    vim.cmd.nohlsearch()
  end

  html_entities(action)
end', {nargs = 1, range = true})

-- Keymappings
vim.keymap.set('n', '<Leader>h', ':Entities 0<CR>', {silent = true})
vim.keymap.set('n', '<Leader>H', ':Entities 1<CR>', {silent = true})
```

Source: vim.fandom.com

22.53 Increment Numbers in Visual Block

Category: editing
Tags: visual-mode, number-manipulation, increment

Quickly generate incrementing sequences in visual block mode, useful for creating arrays or sequences

Example: Example (Vim)

```
" Select first column in block mode
" Press g Ctrl-A to increment numbers
my_array[0] = 0;
(select block, g Ctrl-A)
```

Example: Example (Neovim)

```
-- Vim 8+ functionality, equivalent Lua approach
-- Use vim.cmd to simulate visual block and increment
-- Note: Exact visual block manipulation might require additional scripting
 ↪  in Lua
```

Source: vim.fandom.com

22.54 Increment Numbers in Visual Selection

Category: editing
Tags: visual-mode, number-manipulation, increment

Easily increment numbers across multiple lines using block visual mode and g Ctrl-A

Example: Example (Vim)

```
# In Vim 8+ visual block mode
# Select first column of numbers
g Ctrl-A
```

Example: Example (Neovim)

```
-- Requires Vim 8+ functionality
-- Use Ctrl-V to enter block visual mode
-- Select first column of numbers
-- Use 'g Ctrl-A' to increment sequentially
```

Source: vim.fandom.com

22.55 Increment and Decrement Numbers Quickly

Category: editing
Tags: editing, text-manipulation

Use Ctrl-A and Ctrl-X to quickly increment or decrement numbers under the cursor

Example: Example (Vim)

```
Ctrl-A  # Increment number
Ctrl-X  # Decrement number
```

Example: Example (Neovim)

```
-- In normal mode
vim.keymap.set('n', '<C-a>', '<C-a>', { desc = 'Increment number' })
vim.keymap.set('n', '<C-x>', '<C-x>', { desc = 'Decrement number' })
```

Source: vim.fandom.com

22.56 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: Example (Vim)

```
" 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
```

Source: Community contributed

22.57 Increment/Decrement Numbers Flexibly

Category: editing
Tags: number-manipulation, text-editing, key-mapping

Easily increment or decrement numbers in various formats (decimal, hex, alpha) using Ctrl-A and Ctrl-X with enhanced search capabilities

Example: Example (Vim)

```
function! AddSubtract(char, back)
  let pattern = &nrformats =~ 'alpha' ? '[[:alpha:][:digit:]]' :
  ↪  '[[:digit:]]'
  call search(pattern, 'cw' . a:back)
  execute 'normal! ' . v:count1 . a:char
  silent! call repeat#set(":\<C-u>call AddSubtract('" .a:char. "', '"
  ↪  .a:back. "')\<CR>")
endfunction
```

```
nnoremap <silent>         <C-a> :<C-u>call AddSubtract("\<C-a>", '')<CR>
nnoremap <silent> <Leader><C-a> :<C-u>call AddSubtract("\<C-a>", 'b')<CR>
nnoremap <silent>         <C-x> :<C-u>call AddSubtract("\<C-x>", '')<CR>
nnoremap <silent> <Leader><C-x> :<C-u>call AddSubtract("\<C-x>", 'b')<CR>
```

Example: Example (Neovim)

```
local function add_subtract(char, back)
  local pattern = vim.o.nrformats:find('alpha') and '[%a%d]' or '[%d]'
  vim.fn.search(pattern, 'cw' .. back)
  vim.cmd('normal! ' .. vim.v.count1 .. char)

  -- Optional: Use repeat.vim if available
  pcall(function()
    require('repeat').set(string.format(':lua add_subtract("%s", "%s")<CR>',
    ↪  char, back))
  end)
end

vim.keymap.set('n', '<C-a>', function() add_subtract('\<C-a>', '') end)
vim.keymap.set('n', '<Leader><C-a>', function() add_subtract('\<C-a>', 'b')
↪  end)
vim.keymap.set('n', '<C-x>', function() add_subtract('\<C-x>', '') end)
vim.keymap.set('n', '<Leader><C-x>', function() add_subtract('\<C-x>', 'b')
↪  end)
```

Source: vim.fandom.com

22.58 Insert Current Filename

Category: editing
Tags: insert-mode, filename, completion

Insert current filename in insert or command mode using register expansion

Example: Example (Vim)

```
" In insert mode: Ctrl-R %
" In command mode: Ctrl-R %
" In normal mode: "%p to put filename after cursor
```

Example: Example (Neovim)

```
-- In insert mode, use vim.api.nvim_feedkeys to simulate Ctrl-R %
vim.api.nvim_feedkeys(vim.fn.expand('%'), 'i', true)
```

Source: vim.fandom.com

22.59 Insert Filename in Different Modes

Category: editing
Tags: insert-mode, command-line, file-operations

Various methods to insert the current filename in different Vim modes

Example: Example (Vim)

```
" In insert mode: Ctrl-R %
" In command mode: Ctrl-R %
" In normal mode: "%p (after cursor) or "%P (before cursor)
```

Example: Example (Neovim)

```
-- These are mode-specific Vim commands, so equivalent Lua mapping approach:
vim.keymap.set('i', '<C-x>f', function()
  vim.api.nvim_put({vim.fn.expand('%')}, 'c', true, true)
end, { desc = 'Insert current filename' })
```

Source: vim.fandom.com

22.60 Insert Filename in Neovim

Category: editing
Tags: insert-mode, filename, expansion

Quickly insert the current filename or path in insert mode using expand() function

Example: Example (Vim)

```
" Insert filename without extension
:inoremap <Leader>fn <C-R>=expand("%:t:r")<CR>

" Insert full filename with extension
:inoremap <Leader>fn <C-R>=expand("%:t")<CR>

" Insert full absolute path
:inoremap <Leader>fn <C-R>=expand("%:p:h")<CR>
```

Example: Example (Neovim)

```
-- Insert filename without extension
vim.keymap.set('i', '<Leader>fn', function()
  vim.api.nvim_put({vim.fn.expand('%:t:r')}, '', true, true)
end, { desc = 'Insert filename without extension' })

-- Insert full filename with extension
vim.keymap.set('i', '<Leader>fn', function()
```

```
  vim.api.nvim_put({vim.fn.expand('%:t')}, '', true, true)
end, { desc = 'Insert full filename' })

-- Insert full absolute path
vim.keymap.set('i', '<Leader>fn', function()
  vim.api.nvim_put({vim.fn.expand('%:p:h')}, '', true, true)
end, { desc = 'Insert file path' })
```

Source: vim.fandom.com

22.61 Insert Line Numbers in File

Category: editing
Tags: line-numbering, text-manipulation, search-replace

Programmatically insert sequential line numbers into a file or selected section using Vim's search and replace functionality

Example: Example (Vim)

```
:%s/^/\=printf('%-4d', line('.'))
:'<,'>s/^/\=printf('%d.\t', line('.') - line("'<") + 1)
```

Example: Example (Neovim)

```
-- Insert line numbers for entire file
vim.cmd(':%s/^/\=printf("%-4d", line("."))')

-- Insert line numbers for selected range
vim.cmd("'<,'>s/^/\=printf('%d.\t', line('.') - line("'<") + 1)")
```

Source: vim.fandom.com

22.62 Insert Multiple Lines Efficiently

Category: editing
Tags: line-insertion, key-mapping, productivity

Quickly insert multiple blank lines above or below the current line, with automatic cursor positioning and mode entry

Example: Example (Vim)

```
function! OpenLines(nrlines, dir)
  let nrlines = a:nrlines < 3 ? 3 : a:nrlines
  let start = line('.') + a:dir
  call append(start, repeat([''], nrlines))
  if a:dir < 0
```

```
    normal! 2k
  else
    normal! 2j
  endif
endfunction

nnoremap <Leader>o :<C-u>call OpenLines(v:count, 0)<CR>S
nnoremap <Leader>O :<C-u>call OpenLines(v:count, -1)<CR>S
```

Example: Example (Neovim)

```
function OpenLines(nrlines, dir)
  nrlines = nrlines < 3 and 3 or nrlines
  local start = vim.fn.line('.') + dir

  local empty_lines = vim.fn.repeat({''},nrlines)
  vim.fn.append(start, empty_lines)

  if dir < 0 then
    vim.cmd('normal! 2k')
  else
    vim.cmd('normal! 2j')
  end
end

vim.keymap.set('n', '<Leader>o', function()
  local count = vim.v.count > 0 and vim.v.count or 3
  OpenLines(count, 0)
  vim.cmd('startinsert')
end)

vim.keymap.set('n', '<Leader>O', function()
  local count = vim.v.count > 0 and vim.v.count or 3
  OpenLines(count, -1)
  vim.cmd('startinsert')
end)
```

Source: vim.fandom.com

22.63 Insert Numbered List with Range Function

Category: editing
Tags: list-generation, range, text-manipulation

Easily insert a list of ascending numbers using Vim's range() function, with optional formatting

Example: Example (Vim)

```
:put =range(11,15)
```

```
" Formatted list with leading zeros
:put =map(range(1,150), 'printf(''%04d'', v:val)')
```

Example: Example (Neovim)

```
-- Insert range of numbers
vim.api.nvim_command('put =range(11,15)')

-- Formatted list with leading zeros
vim.api.nvim_command('put =map(range(1,150), function(v) return
↪  string.format('%04d', v) end)')
```

Source: vim.fandom.com

22.64 Insert Special Characters Quickly

Category: editing
Tags: special-characters, unicode, menu-mapping

Create custom menu mappings to easily insert special Unicode characters using nr2char() function

Example: Example (Vim)

```
" Insert special characters via menu
20imenu Editieren.Sonderzeichen.copyright\ \ © <C-R>=nr2char(169)<CR>
20nmenu Editieren.Sonderzeichen.copyright\ \ © a<C-R>=nr2char(169)<CR><Esc>
```

Example: Example (Neovim)

```
-- Lua equivalent for special character insertion
vim.api.nvim_set_keymap('i', '<M-c>', '<C-R>=nr2char(169)<CR>', { noremap =
↪  true, desc = 'Insert copyright symbol' })
vim.api.nvim_set_keymap('n', '<M-c>', 'a<C-R>=nr2char(169)<CR><Esc>', {
↪  noremap = true, desc = 'Insert copyright symbol' })
```

Source: vim.fandom.com

22.65 Insert Special Characters with Digraphs

Category: editing
Tags: special-characters, unicode, input-methods

Easily insert special characters like umlauts, accents, and symbols using digraphs in insert mode

Example: Example (Vim)

```
" Insert special characters
:set digraph
" In insert mode
<C-K>a:
<C-K>e>
```

Example: Example (Neovim)

```
-- Enable digraphs
vim.opt.digraph = true

-- In insert mode, use Ctrl-K followed by two characters
-- e.g., a: for ä, e> for ê
```

Source: vim.fandom.com

22.66 Insert Unicode Characters Easily

Category: editing
Tags: unicode, character-input, insert-mode

Multiple methods to insert Unicode characters in Vim/Neovim: using digraphs, direct Unicode code input, and keyboard input

Example: Example (Vim)

```
" Insert Unicode character methods:
" 1. Digraph: Ctrl-K followed by two characters
" 2. Direct Unicode input: Ctrl-V u XXXX or Ctrl-V U XXXXXXXX
" 3. Keyboard characters work directly in UTF-8 mode
```

Example: Example (Neovim)

```
-- Unicode character input methods remain the same
-- Use :lua vim.api.nvim_input('<C-k>') for digraphs
-- Use :lua vim.api.nvim_input('<C-v>u2023') for direct Unicode input
```

Source: vim.fandom.com

22.67 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: Example (Vim)

```
I  " insert at line start
A  " append at line end
```

Source: Community contributed

22.68 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: Example (Vim)

```
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
```

Source: Community contributed

22.69 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: Example (Vim)

```
" Vimscript:
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>
```

Example: Example (Neovim)

```
-- Lua:
-- Use: o<Esc> or O<Esc> (built-in commands)
-- Or map for convenience:
vim.keymap.set('n', '<leader>o', 'o<Esc>', { desc = 'Insert line below' })
vim.keymap.set('n', '<leader>O', 'O<Esc>', { desc = 'Insert line above' })
```

Source: Community contributed

22.70 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: Example (Vim)

```
" Vimscript:
" 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
```

Example: Example (Neovim)

```
-- Lua:
-- Insert single character and return to normal mode
vim.keymap.set('n', '<leader>i', 'i_<Esc>r', { desc = 'Insert single
↪  character' })
-- Or use: s{char}<Esc> to substitute character under cursor
```

Source: Community contributed

22.71 Inspect Unicode Character Details

Category: editing
Tags: unicode, character-info, inspection

Use built-in commands to get detailed information about Unicode characters

Example: Example (Vim)

```
" Normal mode commands:
ra  " Show character details
g8  " Show byte representation
```

Example: Example (Neovim)

```
-- Character inspection mapping example
vim.keymap.set('n', '<Leader>u', function()
  local char = vim.fn.getcharstr()
  local codepoint = vim.fn.char2nr(char)
  print(string.format('Character: %s, Codepoint: U+%04X', char, codepoint))
end, { desc = 'Inspect Unicode character' })
```

Source: vim.fandom.com

22.72 Join Lines Matching a Pattern Globally

Category: editing
Tags: text-manipulation, global-command, line-joining

Quickly join lines that match a specific pattern using the global command with join

Example: Example (Vim)

```
g/conf/j
```

Example: Example (Neovim)

```
-- Lua equivalent (using ex command)
vim.cmd('g/conf/j')
```

Source: vim.fandom.com

22.73 Join Lines with Customizable Spacing

Category: editing
Tags: line-manipulation, text-editing

Control how lines are joined using 'J' command and joinspaces option

Example: Example (Vim)

```
set nojoinspaces
" Use 'J' to join lines, 'gJ' to join without extra spaces
```

Example: Example (Neovim)

```
vim.opt.joinspaces = false
-- Use 'J' in normal mode to join lines with custom spacing
```

Source: vim.fandom.com

22.74 Line Replacement Across Multiple Locations

Category: editing
Tags: line-manipulation, registers, text-replacement

Copy a line and replace lines at different locations using registers and dot command

Example: Example (Vim)

```
" Workflow:
" Y - yank current line
" p/P - paste after/before current line
" . - repeat paste
```

Example: Example (Neovim)

```
-- Neovim supports the same line replacement techniques
-- Use standard vim.api.nvim_buf_set_lines() for programmatic replacements
```

Source: vim.fandom.com

22.75 Merge Terminal-Split Long Lines

Category: editing
Tags: text-manipulation, terminal-output

Automatically merge long lines split by terminal windows, useful for cleaning up copied output with a consistent 80-character line length

Example: Example (Vim)

```
g/^.\{79}\S$/normal Jx
```

Example: Example (Neovim)

```
-- Merge long lines split by terminal
vim.cmd.global('/^.\{79}\S$/normal! Jx')
```

Source: vim.fandom.com

22.76 Move Comment to End of Line Efficiently

Category: editing
Tags: line-manipulation, code-formatting, key-mapping

Quickly move a comment from a separate line to the end of the previous line of code, reducing vertical space and improving code compactness

Example: Example (Vim)

```
nmap <C-S-j> mz:m-2<CR>J`z
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-S-j>', function()
  -- Move line up and join with previous line
  vim.cmd('normal! mz:m-2\<CR>J`z')
end, { desc = 'Move comment to end of previous line' })
```

Source: vim.fandom.com

22.77 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: Example (Vim)

```
: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
```

Source: Community contributed

22.78 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: Example (Vim)

```
ma      " mark current line as 'a'
:.m'a   " move current line to mark 'a'
:5m'b   " move line 5 to mark 'b'
```

Source: Community contributed

22.79 Number Lines Sequentially in Visual Selection

Category: editing
Tags: text-manipulation, line-numbering, visual-mode

Easily number lines in a visual selection with custom start number and optional suffix

Example: Example (Vim)

```
function! Nlist(...) range
  if 2 == a:0
    let start = a:1
    let append = a:2
  elseif 1 == a:0
    let start = a:1
    let append = " "
  else
    let start = 1
    let append = " "
  endif
  exe a:firstline . "," . a:lastline .
  ↪  's/^/\=line(".")-a:firstline+start.append/'
endfunction
```

Example: Example (Neovim)

```
function _G.number_lines(start, append)
  start = start or 1
  append = append or " "
  local first_line = vim.fn.line("'<")
  local last_line = vim.fn.line("'>")
  for i = first_line, last_line do
    local line_text = vim.fn.getline(i)
    local new_line = string.format("%d%s%s", i - first_line + start, append,
    ↪  line_text)
    vim.fn.setline(i, new_line)
  end
end

-- Usage: In visual mode, call :lua _G.number_lines()
```

Source: vim.fandom.com

22.80 Number Lines Using External Utility

Category: editing
Tags: external-tools, line-numbering

Use Unix 'nl' utility to insert line numbers with flexible formatting options

Example: Example (Vim)

```
:%!nl -ba
:'<,'>!nl -s '. ' -w 2
```

Example: Example (Neovim)

```
-- Number entire file
vim.cmd('%!nl -ba')

-- Number selected range with custom formatting
vim.cmd("'<,'>!nl -s '. ' -w 2")
```

Source: vim.fandom.com

22.81 Number Lines with External Filter

Category: editing
Tags: text-manipulation, external-tools, numbering

Quickly number a range of lines using external Unix tools like 'nl'

Example: Example (Vim)

```
map g# :'n,. !nl<CR>
```

Example: Example (Neovim)

```
-- Number lines from mark 'n' to current position
vim.keymap.set('n', 'g#', ":'n,.!nl<CR>", { desc = 'Number lines using nl'
↪  })
```

Source: vim.fandom.com

22.82 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: Example (Vim)

```
o  " open line below
O  " open line above
```

Source: Community contributed

22.83 Outline Manipulation Techniques

Category: editing
Tags: outline, text-editing, indentation

Use backspace in insert mode to promote and tab to demote headlines, providing quick outline editing

Example: Example (Vim)

```
" In insert mode:
" Backspace to promote
" Tab to demote
```

Example: Example (Neovim)

```
-- Recommended to set up custom keymappings for outline manipulation
-- Example:
vim.keymap.set('i', '<S-Tab>', function()
  -- Implement promote logic
end)

vim.keymap.set('i', '<Tab>', function()
  -- Implement demote logic
end)
```

Source: vim.fandom.com

22.84 Paste with Automatic Indentation

Category: editing
Tags: paste, indentation, text-formatting

Automatically adjust indentation when pasting text to match the surrounding context

Example: Example (Vim)

```
" Map p to paste with correct indentation
:nnoremap p ]p

" Alternative: Paste and re-indent
:nnoremap p p=`]
```

Example: Example (Neovim)

```
-- Map p to paste with correct indentation
vim.keymap.set('n', 'p', ']p', { desc = 'Paste with correct indentation' })

-- Alternative: Paste and re-indent
vim.keymap.set('n', 'p', function()
  vim.cmd('normal! p')
  vim.cmd('normal! =`]')
end, { desc = 'Paste and re-indent' })
```

Source: vim.fandom.com

22.85 Powerful Text Case Manipulation

Category: editing
Tags: text-case, editing, transformation

Quickly change text case in normal mode using simple commands

Example: Example (Vim)

```
guu     " lowercase line
gUU     " uppercase line
~       " invert case of current character
```

Example: Example (Neovim)

```
-- Lua equivalents for text case manipulation
-- Note: These are vim commands that work similarly in Neovim
-- vim.cmd('normal! guu')  -- lowercase line
-- vim.cmd('normal! gUU')  -- uppercase line
-- vim.cmd('normal! ~')    -- invert case of current character
```

Source: vim.fandom.com)

22.86 Preserve Cursor Position When Pasting

Category: editing
Tags: clipboard, cursor-preservation

Use a mark to maintain cursor position when pasting from clipboard

Example: Example (Vim)

```
map <F7> mz:-1r !xclip -o -sel clip<CR>`z
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F7>', 'mz:-1r !xclip -o -sel clip<CR>`z', { desc =
↪  'Paste and preserve cursor position' })
```

Source: vim.fandom.com

22.87 Put (paste) operations

Category: Editing
Tags: put, paste, clipboard

Use `p` to paste after cursor, `P` to paste before cursor.

Example: Example (Vim)

```
p  " paste after cursor
P  " paste before cursor
```

Source: Community contributed

22.88 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: Example (Vim)

```
: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
```

Source: Community contributed

22.89 Quick Email Quote Formatting Techniques

Category: editing
Tags: text-editing, email, formatting

Multiple ways to reformat email quotations using Vim's text formatting operators

Example: Example (Vim)

```
" Formatting techniques:
" Visual mode: V, then gq
" Movement-based: gq} or gq4j
" Text object: gqip or gqap
```

Example: Example (Neovim)

```
-- Note: These are built-in Vim/Neovim operations
-- Use in normal mode with equivalent commands
-- Relies on formatoptions and textwidth settings
```

Source: vim.fandom.com

22.90 Quick Email Thread Cleanup

Category: editing
Tags: email, text-manipulation, autocmd

Easily remove nested reply threads in email messages by deleting lines with multiple '>' characters

Example: Example (Vim)

```
autocmd FileType mail map <F8> :%g/^> >/d<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'mail',
  callback = function()
    vim.keymap.set('n', '<F8>', function()
      vim.cmd('g/^> >/d')
    end, { buffer = true })
  end
})
```

Source: vim.fandom.com

22.91 Quick Footnote Insertion Abbreviations

Category: editing
Tags: abbreviations, text-editing, insertion

Provides abbreviations for quickly inserting and navigating footnotes in a document

Example: Example (Vim)

```
ab (1 [1]<Esc>:/^--\s/-1/<CR>o<insert><CR>Footnotes:<CR>----------<CR>[1]
ab (2 [2]<Esc>:/^Footnotes\:/+2/<CR>o<insert>[2]
ab (3 [3]<Esc>:/^Footnotes\:/+3/<CR>o<insert>[3]
```

Example: Example (Neovim)

```
-- Lua equivalents would typically be implemented using autocmds or custom
↪  functions
-- This is more complex and would require a custom Lua function
-- Example skeleton:
vim.cmd([[cnoreabbrev (1
↪  [1]<Esc>:/^--\s/-1/<CR>o<insert><CR>Footnotes:<CR>----------<CR>[1]]])
```

Source: vim.fandom.com

22.92 Quick Heading Underline with Dashes

Category: editing
Tags: text-formatting, markdown, key-mapping

Quickly add dashed underlines to headings using a simple mapping or function, useful for markdown and documentation

Example: Example (Vim)

```
" Underline the current line with dashes in normal mode
nnoremap <F5> yyp<c-v>$r-

" Underline the current line with dashes in insert mode
inoremap <F5> <Esc>yyp<c-v>$r-A
```

Example: Example (Neovim)

```
-- Underline current line with dashes in normal mode
vim.keymap.set('n', '<F5>', function()
  -- Duplicate current line
  vim.cmd('normal! yyp')
  -- Select entire line and replace with dashes
  vim.cmd('normal! V$r-')
end, { desc = 'Underline with dashes' })

-- Alternatively, with more explicit Lua commands
vim.keymap.set('n', '<F5>', function()
  local line = vim.api.nvim_get_current_line()
  local line_length = #line
  local underline = string.rep('-', line_length)

  vim.api.nvim_buf_set_lines(0, vim.fn.line('.'), vim.fn.line('.'), false,
  ↪  {line, underline})
end, { desc = 'Underline with dashes' })
```

Source: vim.fandom.com

22.93 Quick Minify File in Vim

Category: editing
Tags: text-manipulation, formatting

Quickly remove line breaks and compress a file into a single line using visual mode and Shift-J

Example: Example (Vim)

```
# In visual mode, select entire file
# Then press SHIFT-J
```

Example: Example (Neovim)

```
-- In Neovim, same approach works
-- 1. Select entire file with ggVG
-- 2. Press J to join lines
```

Source: vim.fandom.com

22.94 Quick PHP Debug Code Insertion

Category: editing
Tags: php, abbreviations, debugging

Fast abbreviations for inserting PHP debug statements

Example: Example (Vim)

```
iab phpb exit("<hr>Debug ");
iab phpv echo "<hr><pre>";var_dump($a);exit("debug ");
iab phpallv print_r(get_defined_vars());
```

Example: Example (Neovim)

```
vim.cmd('iab phpb exit("<hr>Debug ");')
vim.cmd('iab phpv echo "<hr><pre>";var_dump($a);exit("debug ");')
vim.cmd('iab phpallv print_r(get_defined_vars());')
```

Source: vim.fandom.com

22.95 Quick Paragraph Line Joining

Category: editing
Tags: text-manipulation, line-editing

Easily join lines within paragraphs, removing excess newlines in a single command

Example: Example (Vim)

```
:v/^\s*$/norm vipJ
```

Example: Example (Neovim)

```
vim.cmd('v/^\s*$/norm vipJ')
```

Source: vim.fandom.com

22.96 Quick Paragraph Sorting in Normal Mode

Category: editing
Tags: text-objects, paragraph-manipulation

Quickly sort a paragraph using a normal mode command

Example: Example (Vim)

```
Vip:
```

Example: Example (Neovim)

```
-- In normal mode, sort current paragraph
vim.cmd('normal Vip:')
```

Source: vim.fandom.com

22.97 Quick Separating Line Abbreviations

Category: editing
Tags: abbreviations, text-formatting, documentation

Create quick abbreviations for inserting different styles of separating lines in documentation or code

Example: Example (Vim)

```
" Line separator abbreviations
abb dlin
 ↪  ==========================================================================
abb cdlin
 ↪  /*========================================================================*/
abb lin
 ↪  --------------------------------------------------------------------------
abb clin
 ↪  /*----------------------------------------------------------------------*/
```

Example: Example (Neovim)

```
-- Lua equivalent for line separator abbreviations
vim.cmd('iabbrev dlin ==========================================================⌋
 ↪  ==================')
vim.cmd('iabbrev cdlin /*======================================================⌋
 ↪  =================*/')
vim.cmd('iabbrev lin ----------------------------------------------------------⌋
 ↪  -----------------')
vim.cmd('iabbrev clin /*------------------------------------------------------⌋
 ↪  ----------------*/')
```

Source: vim.fandom.com

22.98 Quick Text Filtering with External Commands

Category: editing
Tags: text-processing, external-commands, filtering

Use '!<motion><command>' to filter text through external commands, like formatting paragraphs or performing calculations.

Example: Example (Vim)

```
# Filter paragraph through fmt
!}fmt

# Calculate expression
!}bc
```

Example: Example (Neovim)

```
-- Lua equivalent for external command filtering
-- Note: Requires external commands
vim.keymap.set('n', '<leader>fp', '!}fmt<CR>', { desc = 'Format paragraph'
↪  })
vim.keymap.set('n', '<leader>bc', '!}bc<CR>', { desc = 'Calculate
↪  expression' })
```

Source: vim.fandom.com

22.99 Quick Word Completion in Insert Mode

Category: editing
Tags: completion, insert-mode, productivity

Quickly complete words by typing the first few characters and using Ctrl-N or Ctrl-P to cycle through matching completions

Example: Example (Vim)

```
" In insert mode
Ctrl-N  " Complete next matching word
Ctrl-P  " Complete previous matching word
```

Example: Example (Neovim)

```
-- Neovim uses the same default behavior
-- No specific Lua configuration needed
```

```
-- Just use Ctrl-N and Ctrl-P in insert mode
```

Source: vim.fandom.com

22.100 Quick Word Deletion in Insert Mode

Category: editing
Tags: insert-mode, text-editing, key-mapping

Map Ctrl-Backspace to delete previous word in insert mode

Example: Example (Vim)

```
imap <C-BS> <C-W>
```

Example: Example (Neovim)

```
vim.keymap.set('i', '<C-BS>', '<C-W>', { desc = 'Delete previous word' })
```

Source: vim.fandom.com

22.101 Quick Word Replacement with Stamping

Category: editing
Tags: text-objects, replacement, macros

Easily replace words with previously yanked text using a single key mapping

Example: Example (Vim)

```
nnoremap S diw"0P
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'S', 'diw"0P', { desc = 'Replace word with last yanked
↪  text' })
```

Source: vim.fandom.com

22.102 Quick Word and Line Editing Commands

Category: editing
Tags: text-objects, editing-shortcuts, change-command

Efficient ways to change text using Vim's change (c) commands for words, lines, and text objects

Example: Example (Vim)

```
" Change word variants:
cw    " Change word from cursor
caw   " Change whole word
ciw   " Change inner word

" Change line variants:
c$    " Change to end of line
C     " Change to end of line
cc    " Change entire line
cis   " Change inner sentence
```

Example: Example (Neovim)

```
-- These are built-in Vim/Neovim commands, so no direct Lua translation
↪  needed
-- But can be used directly in Neovim
-- Examples:
-- cw, caw, ciw for word changes
-- c$, C, cc for line changes
```

Source: vim.fandom.com

22.103 Quickly Add/Delete Empty Lines

Category: editing
Tags: line-manipulation, key-mapping, productivity

Efficiently insert or remove blank lines above or below the current line without moving the cursor

Example: Example (Vim)

```
" Ctrl-j/k deletes blank line below/above, and Alt-j/k inserts
nnoremap <silent><C-j> m`:silent +g/\m^\s*$/d<CR>``:noh<CR>
nnoremap <silent><C-k> m`:silent -g/\m^\s*$/d<CR>``:noh<CR>
nnoremap <silent><A-j> :set paste<CR>m`o<Esc>``:set nopaste<CR>
nnoremap <silent><A-k> :set paste<CR>m`O<Esc>``:set nopaste<CR>
```

Example: Example (Neovim)

```
-- Delete blank line below/above
vim.keymap.set('n', '<C-j>', function()
  vim.cmd('silent +g/^\s*$/d')
  vim.cmd('noh')
end, { silent = true })
```

```
vim.keymap.set('n', '<C-k>', function()
  vim.cmd('silent -g/^\s*$/d')
  vim.cmd('noh')
end, { silent = true })

-- Insert blank line below/above
vim.keymap.set('n', '<A-j>', function()
  vim.o.paste = true
  vim.cmd('normal! o')
  vim.cmd('normal! `[')
  vim.o.paste = false
end, { silent = true })

vim.keymap.set('n', '<A-k>', function()
  vim.o.paste = true
  vim.cmd('normal! O')
  vim.cmd('normal! `[')
  vim.o.paste = false
end, { silent = true })
```

Source: vim.fandom.com

22.104 Quickly Change Words and Lines

Category: editing
Tags: text-editing, key-mapping, productivity

Provides efficient ways to change words, lines, or text objects with simplified mappings

Example: Example (Vim)

```
nmap pw :inoremap <lt>Space> <lt>Space><lt>Esc>:iunmap
↪  <lt>lt>Space><lt>CR><CR> cw

nmap p$ :inoremap <lt>CR> <lt>CR><lt>Esc>:iunmap <lt>lt>CR><lt>CR><CR> c$
```

Example: Example (Neovim)

```
-- Lua equivalent for quick word/line change
-- Note: This can be achieved more simply with native Neovim keymaps
vim.keymap.set('n', 'pw', function()
  vim.cmd('normal! cw')
  vim.api.nvim_feedkeys(vim.api.nvim_replace_termcodes('<Space>', true,
  ↪  false, true), 'n', true)
end)

vim.keymap.set('n', 'p$', function()
  vim.cmd('normal! c$')
  vim.api.nvim_feedkeys(vim.api.nvim_replace_termcodes('<CR>', true, false,
  ↪  true), 'n', true)
```

```
end)
```

Source: vim.fandom.com

22.105 Quickly Comment Lines in Visual Mode

Category: editing
Tags: visual-mode, commenting, text-manipulation

Easily add comments to multiple lines using visual block mode and substitute command

Example: Example (Vim)

```
:'<,'>s/^/%/  " Add comment at start of selected lines
:'<,'>s/^/<your comment here>/
```

Example: Example (Neovim)

```
-- In visual mode, use:'<,'>s/^/%/ to add comment
-- Or use blockwise visual mode and I to prepend comment
```

Source: vim.fandom.com

22.106 Quickly Insert Current Date or Time

Category: editing
Tags: shortcuts, date-time, insert-mode

Create quick mappings to insert current date or time while editing

Example: Example (Vim)

```
" Insert current date
map <F5> a<C-R>=strftime('%Y-%m-%d')<CR>
" Insert current time
map <F6> a<C-R>=strftime('%H:%M:%S')<CR>
```

Example: Example (Neovim)

```
-- Insert date mapping
vim.keymap.set('n', '<F5>', function()
  vim.api.nvim_put({os.date('%Y-%m-%d')}, 'c', true, true)
end, { desc = 'Insert current date' })

-- Insert time mapping
vim.keymap.set('n', '<F6>', function()
  vim.api.nvim_put({os.date('%H:%M:%S')}, 'c', true, true)
```

```
end, { desc = 'Insert current time' })
```

Source: vim.fandom.com

22.107 Reflow Paragraphs Quickly

Category: editing
Tags: text-editing, formatting, paragraph

Quickly reformat a paragraph to optimal line length using a single command

Example: Example (Vim)

```
" Reflow the inner paragraph
gqip
```

Example: Example (Neovim)

```
-- Reflow the inner paragraph (works the same in Neovim)
vim.cmd('normal gqip')
```

Source: vim.fandom.com

22.108 Remove All Blank Lines in a File

Category: editing
Tags: text-manipulation, command-line

Use global command to delete all blank lines from a file

Example: Example (Vim)

```
:g/^\s*$/d
```

Example: Example (Neovim)

```
-- In command mode
vim.cmd('g/^\s*$/d')
```

Source: vim.fandom.com

22.109 Remove Automatic Comment Leaders Quickly

Category: editing
Tags: insert-mode, comments, text-editing

Quickly remove automatically inserted comment leaders in insert mode without affecting indentation

Example: Example (Vim)

```
" In Insert mode, press Ctrl-U to remove the comment leader
```

Example: Example (Neovim)

```
-- No direct Lua equivalent, but can be configured in autocmd
vim.keymap.set('i', '<C-u>', function()
  -- Remove comment leader while preserving indent
  vim.cmd('norm! d0')
end, { desc = 'Remove comment leader' })
```

Source: vim.fandom.com

22.110 Remove Diacritical Signs from Characters

Category: editing
Tags: text-transformation, character-replacement, internationalization

Efficiently replace characters with diacritical marks (like á, ç) with their base characters, useful for compatibility or normalization

Example: Example (Vim)

```
function! s:RemoveDiacritics(line1, line2)
  let diacs = 'áâãàçéêíóôõüú'
  let repls = 'aaaaceeiooouu'
  let diacs .= toupper(diacs)
  let repls .= toupper(repls)
  let all = join(getline(a:line1, a:line2), "\n")
  call setline(a:line1, split(tr(all, diacs, repls), "\n"))
endfunction
command! -range=% RemoveDiacritics call s:RemoveDiacritics(<line1>, <line2>)
```

Example: Example (Neovim)

```
local function remove_diacritics(line1, line2)
  local diacs = 'áâãàçéêíóôõüú'
  local repls = 'aaaaceeiooouu'
  diacs = diacs .. diacs:upper()
  repls = repls .. repls:upper()

  local lines = vim.api.nvim_buf_get_lines(0, line1-1, line2, false)
  local transformed_lines = vim.tbl_map(function(line)
    return line:gsub('[' .. diacs .. ']', function(char)
      local index = diacs:find(char, 1, true)
      return index and repls:sub(index, index) or char
```

```
    end)
  end, lines)

  vim.api.nvim_buf_set_lines(0, line1-1, line2, false, transformed_lines)
end

vim.api.nvim_create_user_command('RemoveDiacritics', function(opts)
  remove_diacritics(opts.line1, opts.line2)
end, { range = true })
```

Source: vim.fandom.com)

22.111 Remove Empty Lines Efficiently

Category: editing
Tags: text-manipulation, clean-up, search-replace

Quickly remove empty or whitespace-only lines using global commands in Vim/Neovim

Example: Example (Vim)

```
" Remove completely empty lines
:g/^$/d
:v/./d

" Remove lines with only whitespace
:g/^\s*$/d
:v/\S/d
```

Example: Example (Neovim)

```
-- Remove empty lines
vim.cmd('g/^$/d')
vim.cmd('v/./d')

-- Remove lines with only whitespace
vim.cmd('g/^\s*$/d')
vim.cmd('v/\S/d')
```

Source: vim.fandom.com

22.112 Repeat Last Change with Dot Command

Category: editing
Tags: editing, productivity

Press '.' to repeat the last edit or change, saving keystrokes and improving efficiency

Example: Example (Neovim)

```
-- Simply press . in normal mode to repeat last change
```

Source: vim.fandom.com

22.113 Restore Cursor Position in Previous Session

Category: editing
Tags: persistence, cursor-position, workflow

Automatically return to the last cursor position when reopening a file, maintaining your editing context

Example: Example (Vim)

```
augroup RestoreCursor
  autocmd BufReadPost *
    \ if line("'\"") > 0 && line("'\"") <= line("$") |
    \   exe "normal! g`\"" |
    \ endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    local last_pos = vim.fn.line("'\"")
    if last_pos > 0 and last_pos <= vim.fn.line('$') then
      vim.cmd('normal! g`"')
    end
  end
})
```

Source: vim.fandom.com

22.114 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: Example (Vim)

```
`"0     " jump to last exit position
'"0     " jump to last exit position (line start)
:normal `"0  " execute from script/mapping
```

Source: Community contributed

22.115 Reverse Lines Using External Command

Category: editing
Tags: external-tools, text-manipulation

Use external 'tac' utility to reverse lines, works with specific ranges

Example: Example (Vim)

```
:%!tac  # Reverse entire buffer
:100,150!tac  # Reverse lines 100-150
```

Example: Example (Neovim)

```
-- Reverse entire buffer
vim.cmd(':%!tac')

-- Reverse specific line range
vim.cmd(':100,150!tac')
```

Source: vim.fandom.com

22.116 Reverse Lines in Buffer

Category: editing
Tags: text-manipulation, global-command

Quickly reverse the order of lines in a Vim buffer using a global command, useful for reordering lists or log files

Example: Example (Vim)

```
:g/^/m0  # Reverse all lines in buffer
:100,150g/^/m99  # Reverse lines 100-150
```

Example: Example (Neovim)

```
-- Reverse all lines
vim.cmd('g/^/m0')

-- Reverse specific line range
vim.cmd('100,150g/^/m99')
```

Source: vim.fandom.com

22.117 Reverse Lines in a Specific Range

Category: editing
Tags: line-manipulation, range-operations, text-editing

Reverse lines within a specific line range, useful for targeted line order manipulation

Example: Example (Vim)

```
:100,150g/^/m99
```

Example: Example (Neovim)

```
-- Reverse lines 100-150
vim.cmd('100,150g/^/m99')
```

Source: vim.fandom.com

22.118 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: Example (Vim)

```
" 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{
```

Source: Community contributed

22.119 Smart Word Replacement Across Files

Category: editing
Tags: registers, text-manipulation, replace

Efficiently replace words with previously yanked text using different methods, including repeatable changes and register preservation

Example: Example (Vim)

```
" Replace word with yanked text
xnoremap p "_dP

" Stamping technique
nnoremap S diw"0P
```

Example: Example (Neovim)

```
-- Replace word with yanked text
vim.keymap.set('x', 'p', '"_dP', { desc = 'Paste without overwriting
↪  register' })

-- Stamping technique
vim.keymap.set('n', 'S', 'diw"0P', { desc = 'Replace word with last yanked
↪  text' })
```

Source: vim.fandom.com

22.120 Start Editing Blank Lines with Correct Indent

Category: editing
Tags: indentation, editing, productivity

Use cc or S to start editing a blank line with correct indentation instead of i or A

Example: Example (Vim)

```
" No direct vimscript needed, just use cc or S commands
```

Example: Example (Neovim)

```
-- Tip: Use 'cc' or 'S' to start editing
-- These commands are built-in and work with filetype indentation
```

Source: vim.fandom.com

22.121 Substitute character

Category: Editing
Tags: substitute, character, change

Use `s` to substitute character (delete and enter insert mode), `S` for entire line.

Example: Example (Vim)

```
s  " substitute character
S  " substitute line
```

Source: Community contributed

22.122 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: Example (Vim)

```
: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
```

Source: Community contributed

22.123 Swap Characters and Words Easily

Category: editing
Tags: text-manipulation, quick-edit

Quick normal mode commands to swap adjacent characters, words, or lines with minimal keystrokes

Example: Example (Vim)

```
" Swap current character with next
xp

" Swap current word with next
dawwP
```

Example: Example (Neovim)

```
-- Note: These are direct normal mode commands
-- Lua equivalents would typically use vim.api.nvim_input()
-- for similar character/word swapping functionality
```

Source: vim.fandom.com

22.124 Temporarily Disable Auto-Indent for Pasting

Category: editing
Tags: paste, indentation, clipboard

Quickly disable auto-indentation when pasting pre-indented code to prevent double indentation

Example: Example (Vim)

```
:set paste
:set nopaste
```

Example: Example (Neovim)

```
-- Lua equivalent
vim.opt.paste = true  -- Enable paste mode
vim.opt.paste = false  -- Disable paste mode
```

Source: vim.fandom.com

22.125 Temporarily Disable Auto-Indentation for Pasting

Category: editing
Tags: paste, indentation, clipboard

Use paste mode to prevent auto-indentation when inserting pre-indented code

Example: Example (Vim)

```
:set paste
:set nopaste
```

Example: Example (Neovim)

```
-- Temporarily disable auto-indentation
vim.o.paste = true
-- After pasting
vim.o.paste = false
```

Source: vim.fandom.com

22.126 Toggle Paste Mode Easily

Category: editing
Tags: paste-mode, key-mapping, clipboard

Add a quick toggle for paste mode to prevent auto-indentation issues when pasting text

Example: Example (Vim)

```
map <F11> :set invpaste<CR>
set pastetoggle=<F11>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F11>', ':set invpaste<CR>', { desc = 'Toggle paste
↪  mode' })
vim.o.pastetoggle = '<F11>'
```

Source: vim.fandom.com

22.127 Trigger Omni Completion in Insert Mode

Category: editing
Tags: auto-completion, insert-mode, productivity

Use Ctrl-X Ctrl-O to open context-aware completion popup menu in insert mode

Example: Example (Vim)

```
" In insert mode, press Ctrl-X Ctrl-O to trigger omni completion
" Navigate popup with Ctrl-N and Ctrl-P
```

Example: Example (Neovim)

```
-- Omni completion is triggered in insert mode with Ctrl-X Ctrl-O
-- Use built-in Vim mapping, no direct Lua equivalent needed
```

Source: vim.fandom.com

22.128 Use Omni Completion Shortcuts

Category: editing
Tags: autocomplete, insert-mode, productivity

Trigger omni completion and navigate suggestions using keyboard shortcuts

Example: Example (Vim)

```
" Trigger omni completion: Ctrl-X Ctrl-O
" Navigate suggestions: Ctrl-N (next), Ctrl-P (previous)
```

Example: Example (Neovim)

```
-- Use built-in Vim completion mappings
-- <C-x><C-o> to trigger
-- <C-n> and <C-p> to navigate suggestions
```

Source: vim.fandom.com

22.129 Use Text Filters to Process Line Ranges

Category: editing
Tags: text-processing, shell-commands, filtering

Demonstrates using Vim's filter functionality to process selected text ranges with shell commands, enabling powerful text transformation

Example: Example (Vim)

```
" Visual selection followed by ! allows filtering text through shell
↪  commands
" Example: Select lines and use 'cut' to extract specific CSV fields
```

Example: Example (Neovim)

```
-- While this is primarily a Vim editing technique, in Neovim you can
-- achieve similar results with lua functions or external command processing
-- Example filter function:
function ProcessSelectedLines(cmd)
  local start_line = vim.fn.line("'<")
  local end_line = vim.fn.line("'>")
  local lines = vim.fn.getline(start_line, end_line)
  local result = vim.fn.systemlist(cmd, lines)
  vim.api.nvim_buf_set_lines(0, start_line-1, end_line, false, result)
end
```

Source: vim.fandom.com

22.130 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: Example (Vim)

```
" 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
```

Source: Community contributed

22.131 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: Example (Vim)

```
yy  " yank entire line
yw  " yank word
y$  " yank to end of line
```

Source: Community contributed

CHAPTER 23

Ex commands (advanced)

23.1 Advanced substitute flags

Category: Search Replace
Tags: substitute, flags, options, advanced, ex

Use various flags with `:substitute` for advanced replacement options.

Example: Example (Vim)

```
: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
```

23.2 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: Example (Vim)

```
:5a                 " append after line 5
:append             " append after current line
This is new text
.                   " end with dot on empty line
```

23.3 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: Example (Vim)

```
: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
```

23.4 Center align text

Category: Formatting
Tags: center, align, format, text, ex

Use `:center` or `:ce` to center-align text within specified width.

Example: Example (Vim)

```
:center             " center current line (default width)
:ce 80              " center in 80-character width
:1,5ce 60           " center lines 1-5 in 60 chars
```

23.5 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: Example (Vim)

```
:5c                 " change line 5
:1,3c               " change lines 1-3
New replacement text
.                   " end with dot on empty line
```

23.6 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: Example (Vim)

```
: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
```

23.7 Command history navigation

Category: Command Line
Tags: history, navigate, command, previous, ex

Use history navigation to recall and modify previous Ex commands.

Example: Example (Vim)

```
:<Up>                " previous command in history
:<Down>              " next command in history
:<C-p>               " previous command (alternative)
:<C-n>               " next command (alternative)
:his                 " show command history
```

23.8 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. Opposite of `:global` - executes on non-matching lines.

Example: Example (Vim)

```
: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
```

23.9 Global with range

Category: Text Manipulation
Tags: global, range, lines, scope, ex

Use `:global` with line ranges to limit scope of global operations.

Example: Example (Vim)

```
: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 //
```

23.10 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: Example (Vim)

```
:5i                  " insert before line 5
:insert              " insert before current line
New text here
.                    " end with dot on empty line
```

23.11 Join lines together

Category: Text Editing
Tags: join, lines, merge, combine, ex

Use `:join` or `:j` to join lines, removing line breaks.

Example: Example (Vim)

```
: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
```

23.12 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: Example (Vim)

```
:keepjumps normal! G     " go to end without jump entry
:keepjumps /pattern      " search without jump entry
```

23.13 Keep marks during operation

Category: Marks
Tags: keepmarks, marks, preserve, maintain, ex

Use `:keepmarks` to preserve mark positions during range operations.

Example: Example (Vim)

```
:keepmarks 1,5d     " delete lines 1-5 but keep marks
:kee s/old/new/g    " substitute preserving marks
```

23.14 Left align text

Category: Formatting
Tags: left, align, format, text, ex

Use `:left` or `:le` to left-align text, removing leading whitespace.

Example: Example (Vim)

```
:left               " left-align current line
:le 4               " left-align with 4-space indent
:1,10left 0         " remove all leading whitespace from lines 1-10
```

23.15 List old files

Category: File History
Tags: oldfiles, recent, history, files, ex

Use `:oldfiles` to show list of recently edited files. Files are numbered; use `:e #<` to edit by number.

Example: Example (Vim)

```
:oldfiles           " show recently edited files
:ol                 " shorter version
:browse oldfiles    " browse old files with dialog (GUI)
```

23.16 Load saved view

Category: View Management
Tags: loadview, view, restore, position, ex

Use `:loadview` to restore previously saved window view.

Example: Example (Vim)

```
:loadview           " load view with automatic name
:lo 1               " load from view slot 1
:loadview ~/my.vim  " load view from specific file
```

23.17 Lock marks during operation

Category: Marks
Tags: lockmarks, marks, preserve, lock, ex

Use `:lockmarks` to prevent commands from changing mark positions.

Example: Example (Vim)

```
:lockmarks normal! dd    " delete line without affecting marks
:loc s/old/new/g         " substitute without moving marks
```

23.18 Make session file

Category: Session Management
Tags: mksession, session, save, workspace, ex

Use `:mksession` to save current editing session to file. Restore with `:source Session.vim` or `nvim -S Session.vim`.

Example: Example (Vim)

```
:mksession          " create Session.vim in current dir
:mks ~/my.vim       " save session to specific file
:mks!               " overwrite existing session file
```

23.19 Nested global commands

Category: Text Manipulation
Tags: global, nested, complex, pattern, ex

Use nested `:global` commands for complex pattern operations. Inner global operates on lines found by outer global.

Example: Example (Vim)

```
:g/function/+1,/^}/g/TODO/p   " find TODO in function bodies
:g/class/.,/^$/v/def/d        " delete non-def lines in classes
```

23.20 Put register contents

Category: Registers
Tags: put, paste, register, insert, ex

Use `:put` to insert register contents after current line.

Example: Example (Vim)

```
: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
```

23.21 Quit with error code

Category: Exit
Tags: cquit, quit, error, code, ex

Use `:cquit` or `:cq` to quit Vim with error exit code. Useful in shell scripts to indicate failure.

Example: Example (Vim)

```
:cquit              " quit with error code
:cq                 " shorter version
:cq 2               " quit with specific error code
```

23.22 Range with patterns

Category: Text Manipulation
Tags: range, pattern, search, scope, ex

Use patterns as range specifiers in Ex commands.

Example: Example (Vim)

```
:/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
```

23.23 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. Historical command, rarely needed in modern Neovim.

Example: Example (Vim)

```
:visual             " return to Normal mode
:vi                 " shorter version
```

23.24 Right align text

Category: Formatting
Tags: right, align, format, text, ex

Use `:right` or `:ri` to right-align text within specified width.

Example: Example (Vim)

```
:right              " right-align current line
:ri 80              " right-align in 80-character width
:1,5ri 60           " right-align lines 1-5 in 60 chars
```

23.25 Save current view

Category: View Management
Tags: mkview, view, save, position, ex

Use `:mkview` to save current window view (cursor position, folds, etc.).

Example: Example (Vim)

```
:mkview             " save view with automatic name
:mkv 1              " save to view slot 1
:mkview ~/my.vim    " save view to specific file
```

23.26 Sort lines alphabetically

Category: Text Manipulation
Tags: sort, alphabetical, lines, order, ex

Use `:sort` to sort lines in various ways.

Example: Example (Vim)

```
: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
```

23.27 Substitute confirmation

Category: Search Replace
Tags: substitute, confirm, interactive, replace, ex

Use the `c` flag with `:substitute` for interactive confirmation of each replacement. Prompts: `yes`, `no`, `all`, `quit`, `last`.

Example: Example (Vim)

```
: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
```

23.28 Substitute with backreferences

Category: Search Replace
Tags: substitute, backreference, capture, group, ex

Use `\(` and `\)` to capture groups, reference with `\1`, `\2`, etc.

Example: Example (Vim)

```
:s/\(word1\) \(word2\)/\2 \1/    " swap two words
:s/\(\w\+\)\s\+\(\w\+\)/\2, \1/  " swap and add comma
:%s/\(.*\)/"\1"/                 " quote all lines
```

23.29 Substitute with expressions

Category: Search Replace
Tags: substitute, expression, function, dynamic, ex

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

Example: Example (Vim)

```
:s/\d\+/\=submatch(0)*2/g    " double all numbers
:s/.*/\=line('.').' '.submatch(0)/  " add line number prefix
:%s/$/\=', line: '.line('.')/       " add line number suffix
```

23.30 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: Example (Vim)

```
:wqall              " write all modified and quit all
:wqa                " shorter version
:xa                 " write all modified and exit (alternative)
```

23.31 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. More efficient than `:wq` since it only writes when necessary.

Example: Example (Vim)

```
:xit                " write if modified and exit
:x                  " shorter version
:5,10x file.txt     " write lines 5-10 to file and exit
```

23.32 Yank lines to register

Category: Registers
Tags: yank, copy, register, lines, ex

Use `:yank` or `:y` to copy lines to a register.

Example: Example (Vim)

```
: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 24

Ex commands (comprehensive)

24.1 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. Status indicators: `%` current, `#` alternate, `+` modified, `x` read errors.

Example: Example (Vim)

```
:ls                  " list all buffers
:buffers            " same as :ls (alternative)
:ls!                " list all buffers including unlisted
```

24.2 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. Useful for cleaning up complex window layouts quickly.

Example: Example (Vim)

```
:only               " close all other windows
:on                 " shorter version
```

24.3 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: Example (Vim)

```
: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)
```

24.4 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: Example (Vim)

```
:enew               " create new empty buffer
:new                " create new buffer in split window
:vnew               " create new buffer in vertical split
```

24.5 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: Example (Vim)

```
: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
```

24.6 Delete specific lines

Category: Text Manipulation
Tags: delete, lines, range, remove, ex

Use `:delete` or `:d` to delete specific lines or line ranges.

Example: Example (Vim)

```
: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
```

24.7 Execute normal mode commands

Category: Command Execution
Tags: normal, execute, mode, command, ex

Use `:normal commands` to execute normal mode commands from Ex mode. Use `!` to avoid mappings: `:normal! dd`

Example: Example (Vim)

```
:normal dd         " delete current line
:5,10normal A;     " append semicolon to lines 5-10
:%normal I//       " comment all lines with //
```

24.8 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. Set your path with `:set path+=directory` to include custom directories.

Example: Example (Vim)

```
:find config.vim   " find and edit config.vim in path
:fin *.py          " find Python files (with tab completion)
```

24.9 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: Example (Vim)

```
:first             " edit first file in argument list
:rewind            " same as :first
:last              " edit last file in argument list
```

24.10 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: Example (Vim)

```
:buffer 3          " go to buffer number 3
:b 3               " shorter version
:b filename        " go to buffer by partial filename match
```

24.11 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). Use `j` flag to avoid jumping to first match immediately.

Example: Example (Vim)

```
:vimgrep /pattern/j *.py   " search in Python files
:vim /\cTODO/ **/*.js      " case-insensitive recursive search
:vimgrep /\<word\>/ %      " search for whole word in current file
```

24.12 Jump to tag definition

Category: Navigation
Tags: tag, jump, definition, ctags, ex

Use `:tag tagname` to jump to tag definition (requires tags file). Generate tags with `ctags -R .` in your project root.

Example: Example (Vim)

```
:tag function_name " jump to tag definition
:ta MyClass        " jump to MyClass tag
:tag /pattern      " search for tags matching pattern
```

24.13 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. Useful for debugging configuration issues and seeing load order.

Example: Example (Vim)

```
:scriptnames       " list all sourced scripts
:scr               " shorter version
```

24.14 Location list navigation

Category: Search
Tags: location, list, navigate, lnext, ex

Use `:lnext`, `:lprev` to navigate location list (window-local quickfix).

Example: Example (Vim)

```
: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
```

24.15 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: Example (Vim)

```
: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
```

24.16 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. See argument list with `:args`, set with `nvim file1 file2 file3`.

Example: Example (Vim)

```
:next                " edit next file
:n                   " shorter version
:2next               " skip 2 files forward
```

24.17 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: Example (Vim)

```
:previous            " edit previous file
:prev                " shorter version
:2prev               " skip 2 files backward
```

24.18 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. Use after jumping to tags with `:tag` or `Ctrl+]`.

Example: Example (Vim)

```
:pop                 " go back in tag stack
:po                  " shorter version
:2pop                " go back 2 positions
```

24.19 Quickfix list navigation

Category: Search
Tags: quickfix, navigate, error, jump, ex

Use `:cnext`, `:cprev` to navigate quickfix list (global error list).

Example: Example (Vim)

```
: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
```

24.20 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: Example (Vim)

```
:qall               " quit all windows/buffers
:qa                 " shorter version
:qa!                " quit all without saving changes
:wqa                " save all and quit
```

24.21 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. Vim creates `.swp` files for crash recovery. Use this after unexpected shutdowns.

Example: Example (Vim)

```
:recover file.txt   " recover file from swap
:rec                " recover current file
```

24.22 Repeat last Ex command

Category: Command History
Tags: repeat, last, command, history, ex

Use `@:` to repeat the last Ex command, `@@` to repeat last `@` command. Useful for repeating complex commands without retyping.

Example: Example (Vim)

```
@:                  " repeat last Ex command
5@:                 " repeat last Ex command 5 times
@@                  " repeat the last @ command
```

24.23 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. Results appear in quickfix list. Use `:cn` and `:cp` to navigate.

Example: Example (Vim)

```
: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
```

24.24 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: Example (Vim)

```
:wall              " write all modified buffers
:wa                " shorter version
:wqa               " write all and quit
:xa                " write all modified and exit
```

24.25 Set local options

Category: Configuration
Tags: set, local, options, buffer, window, ex

Use `:setlocal` to set options only for current buffer/window.

Example: Example (Vim)

```
: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
```

24.26 Show argument list

Category: File Navigation
Tags: args, argument, list, files, ex

Use `:args` to display current argument list with current file highlighted.

Example: Example (Vim)

```
:args               " show argument list
:args *.py          " set argument list to all Python files
:args **/*.js       " recursively find all JavaScript files
```

24.27 Show version information

Category: System Information
Tags: version, info, build, features, ex

Use `:version` to display Neovim version, build info, and compiled features. Shows version number, build date, features, and compilation options.

Example: Example (Vim)

```
:version            " show full version information
:ve                 " shorter version
```

24.28 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: Example (Vim)

```
: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 25

Ex commands (extended)

25.1 Add buffer to list

Category: Buffer Management
Tags: badd, buffer, add, list, ex

Use `:badd` to add file to buffer list without editing it. Useful for preparing a list of files to work with.

Example: Example (Vim)

```
:badd file.txt     " add file to buffer list
:badd *.py         " add all Python files
```

25.2 AutoGroup management

Category: Scripting
Tags: augroup, autocmd, group, event, ex

Use `:augroup` to group autocommands and manage them collectively.

Example: Example (Vim)

```
:augroup MyGroup    " start group definition
:autocmd!           " clear existing autocmds in group
:autocmd BufRead * echo "File read"
:augroup END        " end group definition
```

25.3 Call functions

Category: Scripting
Tags: call, function, execute, script, ex

Use `:call` to execute functions and discard their return value.

Example: Example (Vim)

```
:call search('pattern')  " call search function
:call setline('.', 'new text')  " replace current line
:call cursor(10, 5)      " move cursor to line 10, column 5
```

25.4 Change working directory

Category: Navigation
Tags: directory, cd, path, working, ex

Use `:cd` to change current working directory, `:pwd` to show current directory.

Example: Example (Vim)

```
: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
```

25.5 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. Useful for debugging include paths in C/C++ projects.

Example: Example (Vim)

```
:checkpath          " check all included files
:checkp!            " show files that cannot be found
```

25.6 Conditional execution

Category: Scripting
Tags: if, condition, branch, script, ex

Use `:if`, `:else`, `:endif` for conditional execution in scripts.

Example: Example (Vim)

```
:if &filetype == 'python'
  echo "Python file"
:else
  echo "Not Python"
```

```
:endif
```

25.7 Create abbreviations

Category: Text Input
Tags: abbreviate, abbr, shortcut, expand, ex

Use `:abbreviate` or `:abbr` to create text shortcuts that expand automatically.

Example: Example (Vim)

```
: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
```

25.8 Define variables

Category: Scripting
Tags: let, variable, assign, define, ex

Use `:let` to define and assign values to variables.

Example: Example (Vim)

```
: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
```

25.9 Echo text and expressions

Category: Scripting
Tags: echo, print, expression, debug, ex

Use `:echo` to print text or evaluate expressions in command line.

Example: Example (Vim)

```
:echo "Hello World" " print text
:echo &tabstop      " show value of tabstop option
:echo expand('%')   " show current filename
:echo line('.')     " show current line number
```

25.10 File information

Category: Information
Tags: file, info, status, buffer, ex

Use `:file` to show file information and optionally rename buffer.

Example: Example (Vim)

```
:file              " show file info (name, lines, position)
:file newname.txt  " rename current buffer
:f                 " shorter version
```

25.11 Function definition

Category: Scripting
Tags: function, define, script, procedure, ex

Use `:function` to define custom functions in Vim script.

Example: Example (Vim)

```
:function! MyFunc()
  echo "Hello from function"
endfunction

:call MyFunc()      " call the function
:delfunc MyFunc     " delete function
```

25.12 Help search

Category: Help
Tags: helpgrep, help, search, documentation, ex

Use `:helpgrep` to search through all help files for patterns.

Example: Example (Vim)

```
:helpgrep pattern   " search help for pattern
:helpg autocmd      " search for autocmd info
:cn                 " next help match
:cp                 " previous help match
```

25.13 Include jump

Category: Navigation
Tags: ijump, include, file, search, ex

Use `:ijump` to jump to first line containing identifier in include files.

Example: Example (Vim)

```
:ijump printf       " jump to printf definition in includes
:ij MyFunc          " jump to MyFunc in include files
```

25.14 Include list

Category: Navigation
Tags: ilist, include, search, show, ex

Use `:ilist` to list all lines containing identifier in include files.

Example: Example (Vim)

```
:ilist printf       " list all printf occurrences
:il /pattern/       " list lines matching pattern
:il! MyFunc         " list including header files
```

25.15 Introduction screen

Category: Interface
Tags: intro, welcome, screen, startup, ex

Use `:intro` to show the Neovim introduction/welcome screen. Useful after clearing the screen or when you want to see version info.

Example: Example (Vim)

```
:intro              " show introduction screen
```

25.16 Key mapping

Category: Mapping
Tags: map, key, mapping, shortcut, ex

Use `:map` to create key mappings, `:noremap` for non-recursive mappings.

Example: Example (Vim)

```
: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
```

25.17 Language settings

Category: Configuration
Tags: language, locale, encoding, international, ex

Use `:language` to set language for messages and time.

Example: Example (Vim)

```
:language messages en_US.UTF-8  " set message language
:language time C                " set time language
:language                       " show current settings
```

25.18 Make and build

Category: Development
Tags: make, build, compile, external, ex

Use `:make` to run external make command and capture errors. Errors appear in quickfix list. Use `:cn` to navigate.

Example: Example (Vim)

```
:make              " run make command
:make clean        " run make with clean target
:make -j4          " run make with 4 parallel jobs
```

25.19 Match highlighting

Category: Display
Tags: match, highlight, pattern, visual, ex

Use `:match` to highlight patterns with specific colors in current window.

Example: Example (Vim)

```
:match ErrorMsg /TODO/     " highlight TODO in red
:match Search /\<word\>/   " highlight whole word
:match none                " clear all matches
```

```
:2match Comment /pattern/  " second match group
```

25.20 Menu creation

Category: Interface
Tags: menu, gui, interface, create, ex

Use `:menu` to create menu items (GUI mode).

Example: Example (Vim)

```
:menu File.Save :w<CR>           " create Save menu item
:menu Edit.Find :promptfind<CR>  " create Find menu item
:unmenu File.Save                " remove menu item
```

25.21 Neovim health check

Category: Diagnostics
Tags: checkhealth, health, diagnostic, status, ex

Use `:checkhealth` to run diagnostic checks on Neovim installation and plugins.

Example: Example (Vim)

```
:checkhealth        " check all health
:checkhealth nvim   " check only Neovim core health
:checkhealth vim    " check Vim compatibility
```

25.22 Print lines

Category: Display
Tags: print, lines, show, output, ex

Use `:print` or `:p` to print lines to command area.

Example: Example (Vim)

```
:print             " print current line
:1,5p              " print lines 1 through 5
:.,$p              " print from current line to end
:p #               " print with line numbers
```

25.23 Runtime file loading

Category: Configuration
Tags: runtime, load, path, script, ex

Use `:runtime` to load script files from runtime path directories.

Example: Example (Vim)

```
:runtime! plugin/**/*.vim  " load all plugins
:runtime syntax/python.vim " load Python syntax
:ru macros/matchit.vim     " load matchit macro
```

25.24 Show all marks

Category: Navigation
Tags: marks, list, show, position, ex

Use `:marks` to display all marks with their line numbers and text.

Example: Example (Vim)

```
: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
```

25.25 Show all messages

Category: Information
Tags: messages, history, errors, warnings, ex

Use `:messages` to display message history including errors and warnings.

Example: Example (Vim)

```
:messages           " show all messages
:mes                " shorter version
:messages clear     " clear message history
```

25.26 Show digraphs

Category: Text Input
Tags: digraphs, special, characters, unicode, ex

Use `:digraphs` to show available two-character combinations for special characters. In

insert mode, use `Ctrl+k` followed by two characters (e.g., `Ctrl+k a'` for á).

Example: Example (Vim)

```
:digraphs           " show all digraphs
:dig                " shorter version
```

25.27 Show jump list

Category: Navigation
Tags: jumps, list, history, navigation, ex

Use `:jumps` to display jump list with positions you can return to. Use `Ctrl+o` to go back, `Ctrl+i` to go forward in jump list.

Example: Example (Vim)

```
:jumps              " show jump list
:ju                 " shorter version
```

25.28 Show registers content

Category: Registers
Tags: registers, show, content, clipboard, ex

Use `:registers` or `:reg` to display contents of all registers.

Example: Example (Vim)

```
:registers          " show all registers
:reg abc            " show only registers a, b, and c
:reg "              " show default register
:reg +              " show system clipboard register
```

25.29 Spell checking commands

Category: Text Editing
Tags: spell, check, dictionary, correction, ex

Use spell-related commands to manage spell checking.

Example: Example (Vim)

```
: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
```

25.30 Tag selection

Category: Navigation
Tags: tselect, tag, multiple, choose, ex

Use `:tselect` when multiple tag matches exist to choose from a list.

Example: Example (Vim)

```
: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
```

25.31 Unlet variables

Category: Scripting
Tags: unlet, variable, delete, remove, ex

Use `:unlet` to delete variables and free memory.

Example: Example (Vim)

```
:unlet g:my_var    " delete global variable
:unlet! b:temp     " delete if exists (no error)
:unlet $HOME       " delete environment variable
```

CHAPTER 26

Exit

26.1 Exit with error code

Category: Exit
Tags: quit, exit, error, git, abort

Use `:cq` to exit Vim with a non-zero error code. Useful for aborting git operations like `git rebase` or `git commit`. When used during `git rebase -i` or `git commit`, this aborts the operation instead of completing it.

Example: Example (Vim)

```
:cq  " exit with error code (non-zero status)
```

26.2 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: Example (Vim)

```
:q   " quit
:q!  " quit without saving
:wq  " write and quit
ZZ   " save and exit
ZQ   " quit without saving
```

CHAPTER 27

Extmarks

27.1 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: Example (Neovim)

```
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
})
```

27.2 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: Example (Neovim)

```
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",
})
```

27.3 Build a simple word abbreviation system

Category: Extmarks
Tags: extmarks, abbreviations, conceal, practical

Create a complete system to visually abbreviate long keywords using extmarks with conceal and inline virtual text.

Example: Example (Neovim)

```
-- 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

  -- Enable concealing (required)
  vim.opt.conceallevel = 2

  -- 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,
            conceal = "",  -- Hide original text
            virt_text = {{abbrev, "Keyword"}},
            virt_text_pos = "inline",  -- Insert replacement
          })
        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
```

27.4 Clear extmarks from buffer

Category: Extmarks
Tags: extmarks, clear, cleanup, namespace

Use `nvim_buf_clear_namespace()` to remove extmarks. Essential for cleanup and refreshing decorations.

Example: Example (Neovim)

```
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)
```

27.5 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: Example (Neovim)

```
-- 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])
```

27.6 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: Example (Neovim)

```
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
```

27.7 Create inline diagnostics with extmarks

Category: Extmarks
Tags: extmarks, diagnostics, lsp, inline

Combine extmarks features to create rich inline diagnostics like modern IDEs.

Example: Example (Neovim)

```
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")
```

27.8 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: Example (Neovim)

```
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",
})
```

27.9 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: Example (Neovim)

```
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,
})
```

27.10 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: Example (Neovim)

```
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}, {})
```

27.11 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: Example (Neovim)

```
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 "f" visually
vim.api.nvim_buf_set_extmark(0, ns_id, 10, 0, {
  end_col = 8,
  conceal = "f",  -- show "f" instead
})

-- Note: Make sure conceallevel is set
vim.opt.conceallevel = 2
```

27.12 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: Example (Neovim)

```
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
})
```

27.13 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: Example (Neovim)

```
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
})
```

27.14 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: Example (Neovim)

```
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,
})
```

27.15 Replace keywords visually with extmarks

Category: Extmarks
Tags: extmarks, conceal, replace, keywords

Use extmarks with `conceal` and `virt_text_pos = "inline"` to visually replace text like ”return” with ”ret” without changing the buffer. Using `overlay` alone will overlay on top rather than replace.

Example: Example (Neovim)

```
local ns_id = vim.api.nvim_create_namespace('abbreviations')

-- Enable concealing (required for conceal to work)
vim.opt.conceallevel = 2

-- 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,
      conceal = "",  -- Hide the original text
      virt_text = {{"ret", "Keyword"}},
      virt_text_pos = "inline",  -- Insert replacement text
    })
    col = finish + 1
  end
```

```
end
```

CHAPTER 28
File operations

28.1 Add Modeline to Permanently Protect File

Category: file_operations
Tags: file-protection, modeline, configuration

Use a modeline to make a file non-modifiable across editing sessions

Example: Example (Vim)

```
// vim:noma
```

Example: Example (Neovim)

```
-- Add the following at the end of the file
-- vim:noma
```

Source: vim.fandom.com

28.2 Add Vim to Windows Send To Menu

Category: file_operations
Tags: windows, file-management, productivity

Easily add Vim to Windows Send To context menu for quick file opening with custom options

Example: Example (Vim)

```
" Example shortcut creation in SendTo folder
" Use gvim.exe with various launch options
" e.g., gvim.exe -d (diff mode)
" e.g., gvim.exe -R (read-only mode)
```

Example: Example (Neovim)

```
-- In Neovim, this is more about file associations and keymaps
-- Lua equivalent would be configuring file handlers
vim.g.neovide_input_use_logo = true  -- Example of custom file handling
```

Source: vim.fandom.com

28.3 Associate Extensionless Files with Vim in Windows

Category: file_operations
Tags: windows, file-association, file-handling

Automatically open files without extensions in Vim using Windows command prompt or registry

Example: Example (Vim)

```
assoc .=txtfile
```

Example: Example (Neovim)

```
-- For Neovim, this is more of a Windows system configuration
-- Typically handled through external system settings
-- Lua equivalent would involve using vim.fn to interact with system
```

Source: vim.fandom.com

28.4 Auto-Detect and Reload Mixed Line Endings

Category: file_operations
Tags: file-format, autocmd, line-endings

Automatically detect and reload files with mixed line endings, converting from Unix to DOS format when CR characters are detected at line ends

Example: Example (Vim)

```
autocmd BufReadPost * nested
      \ if !exists('b:reload_dos') && !&binary && &ff=='unix' && (0 <
      ↪  search('\r$', 'nc')) |
      \   let b:reload_dos = 1 |
      \   e ++ff=dos |
      \ endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  nested = true,
  callback = function()
    local buf = vim.api.nvim_get_current_buf()
    if not vim.b[buf].reload_dos and
       not vim.o.binary and
       vim.o.fileformat == 'unix' and
       vim.fn.search('\r$', 'nc') > 0 then
      vim.b[buf].reload_dos = true
      vim.cmd('e ++ff=dos')
    end
  end
})
```

Source: vim.fandom.com

28.5 Automatic Remote File Synchronization

Category: file_operations
Tags: remote-editing, file-sync, network

Automatically transfer and save remote files when editing, eliminating manual upload steps

Example: Example (Vim)

```
" Write remote file back
:Nwrite scp://user@hostname//path/to/file
```

Example: Example (Neovim)

```
-- Write remote file back in Neovim
vim.cmd('Nwrite scp://user@hostname//path/to/file')
```

Source: vim.fandom.com

28.6 Automatically Create Backup and Tmp Directories

Category: file_operations
Tags: backup, directory-management, cross-platform

Automatically creates backup and temporary directories across different operating systems, ensuring Vim has proper directories for storing backup and swap files

Example: Example (Vim)

```
function! InitBackupDir()
  if has('win32') || has('win32unix') "windows/cygwin
    let l:separator = '_'
  else
    let l:separator = '.'
  endif
  let l:parent = $HOME . '/' . l:separator . 'vim/'
  let l:backup = l:parent . 'backup/'
  let l:tmp = l:parent . 'tmp/'
  if exists('*mkdir')
    if !isdirectory(l:parent)
      call mkdir(l:parent)
    endif
    if !isdirectory(l:backup)
      call mkdir(l:backup)
    endif
    if !isdirectory(l:tmp)
      call mkdir(l:tmp)
    endif
  endif
  let l:missing_dir = 0
  if isdirectory(l:tmp)
    execute 'set backupdir=' . escape(l:backup, ' ') . '/,.'
  else
    let l:missing_dir = 1
  endif
  if isdirectory(l:backup)
    execute 'set directory=' . escape(l:tmp, ' ') . '/,.'
  else
    let l:missing_dir = 1
  endif
  if l:missing_dir
    echo 'Warning: Unable to create backup directories:' l:backup 'and'
    ↪  l:tmp
    echo 'Try: mkdir -p' l:backup
    echo 'and: mkdir -p' l:tmp
    set backupdir=.
    set directory=.
  endif
endfunction
call InitBackupDir()
```

Example: Example (Neovim)

```
local function init_backup_dir()
  local separator = vim.fn.has('win32') == 1 and '_' or '.'
  local parent = vim.fn.expand('$HOME') .. '/' .. separator .. 'vim/'
  local backup = parent .. 'backup/'
  local tmp = parent .. 'tmp/'

  if vim.fn.exists('*mkdir') then
    if not vim.fn.isdirectory(parent) then
      vim.fn.mkdir(parent, 'p')
```

```
    end
    if not vim.fn.isdirectory(backup) then
      vim.fn.mkdir(backup, 'p')
    end
    if not vim.fn.isdirectory(tmp) then
      vim.fn.mkdir(tmp, 'p')
    end
  end

  local missing_dir = false
  if vim.fn.isdirectory(tmp) then
    vim.opt.backupdir = backup .. ',.'
  else
    missing_dir = true
  end

  if vim.fn.isdirectory(backup) then
    vim.opt.directory = tmp .. ',.'
  else
    missing_dir = true
  end

  if missing_dir then
    print('Warning: Unable to create backup directories: ' .. backup .. '
    ↪  and ' .. tmp)
    print('Try: mkdir -p ' .. backup)
    print('and: mkdir -p ' .. tmp)
    vim.opt.backupdir = '.'
    vim.opt.directory = '.'
  end
end

init_backup_dir()
```

Source: vim.fandom.com

28.7 Automatically Create Directories When Editing Files

Category: file_operations
Tags: directory-creation, autocommand, workflow-enhancement

Automatically create parent directories when opening a new file that doesn't exist, saving manual directory creation steps

Example: Example (Vim)

```
au BufNewFile * :exe ': !mkdir -p ' .
↪  escape(fnamemodify(bufname('%'),':p:h'),'#% \')
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufNewFile', {
  callback = function()
    local dir = vim.fn.fnamemodify(vim.api.nvim_buf_get_name(0), ':p:h')
    vim.fn.mkdir(dir, 'p')
  end
})
```

Source: vim.fandom.com

28.8 Batch Convert Line Endings Across Files

Category: file_operations
Tags: file-format, batch-processing, line-endings

Convert line endings for multiple files in a single operation using argdo or bufdo

Example: Example (Vim)

```
:args *.c *.h
:argdo set ff=unix|update
```

Example: Example (Neovim)

```
-- Convert all .c and .h files to unix line endings
vim.cmd('args *.c *.h')
vim.cmd('argdo set ff=unix|update')
```

Source: vim.fandom.com

28.9 Batch Convert Line Endings for Multiple Files

Category: file_operations
Tags: batch-processing, file-format, line-endings

Convert line endings for multiple files in a single directory using argument list commands

Example: Example (Vim)

```
:args *.c *.h
:argdo set ff=unix|update
```

Example: Example (Neovim)

```
-- Convert .c and .h files to Unix line endings
vim.cmd('args *.c *.h')
vim.cmd('argdo set ff=unix|update')
```

Source: vim.fandom.com

28.10 Batch Convert Line Endings in Multiple Files

Category: file_operations
Tags: file-format, batch-processing, line-endings

Convert line endings for multiple files matching a pattern in one command

Example: Example (Vim)

```
:args *.c *.h
:argdo set ff=unix|update
```

Example: Example (Neovim)

```
-- Convert specified file types to unix line endings
vim.cmd('args *.c *.h')
vim.cmd('argdo set ff=unix|update')
```

Source: vim.fandom.com

28.11 Batch Edit Multiple Files from File List

Category: file_operations
Tags: batch-editing, macro, command-line

Efficiently edit multiple files using a Vim macro and command-line file loading

Example: Example (Vim)

```
vim -s scriptin `cat file-containing-files`

# In scriptin macro file:
qq/pattern^Mdd:wn^M^M@qq@q
```

Example: Example (Neovim)

```
-- Open multiple files from a list
-- Lua equivalent would typically use vim.fn.argv() or a plugin like
↪  telescope
-- For macro recording:
```

```
vim.fn.setreg('q', 'search_pattern_and_delete_macro')
-- Execute macro across files
vim.cmd('argdo norm! @q')
```

Source: vim.fandom.com

28.12 Binary File Editing Mode

Category: file_operations
Tags: binary-files, hex-editing, file-modes

Open files in binary mode with hex display and line wrapping

Example: Example (Vim)

```
vim -b myfile.bin
:setlocal display=uhex
:setlocal wrap
```

Example: Example (Neovim)

```
-- Open in binary mode
vim.o.binary = true
vim.o.display = 'uhex'
vim.o.wrap = true
```

Source: vim.fandom.com

28.13 Break Hard Links When Editing Files

Category: file_operations
Tags: hard-links, file-backup, unix

Automatically break hard links when editing a file, creating a separate copy instead of modifying all linked files

Example: Example (Vim)

```
set backupcopy=auto,breakhardlink
```

Example: Example (Neovim)

```
vim.opt.backupcopy = "auto,breakhardlink"
```

Source: vim.fandom.com

28.14 Browse and Edit JAR Files Directly in Vim/Neovim

Category: file_operations
Tags: zip, java, file-browsing

Treat JAR files as zip archives, allowing direct browsing and editing of compressed file contents within Vim/Neovim

Example: Example (Vim)

```
au BufRead,BufNewFile *.jar,*.war,*.ear,*.sar,*.rar set filetype=zip

" Alternative method
au BufReadCmd *.jar call zip#Browse(expand("<amatch>"))
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({"BufRead", "BufNewFile"}, {
  pattern = {"*.jar", "*.war", "*.ear", "*.sar", "*.rar"},
  callback = function()
    vim.bo.filetype = "zip"
  end
})

-- Alternative method for browsing
-- Requires zipPlugin to be installed
vim.api.nvim_create_autocmd("BufReadCmd", {
  pattern = "*.jar",
  callback = function()
    vim.fn['zip#Browse'](vim.fn.expand('<amatch>'))
  end
})
```

Source: vim.fandom.com

28.15 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: Example (Vim)

```
: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
```

Source: Community contributed

28.16 Bulk Create Directories from Text List

Category: file_operations
Tags: mkdir, batch-operations, file-management

Create multiple directories from a list of names in a text file using a single Vim command

Example: Example (Vim)

```
%g/\<\w\+\>/ y A | exe ' !mkdir '. shellescape(substitute(substitute(@a,
↪  '\n\+\s*', '', ''), '\s*\n\+', '', '')) | let @a = ""
```

Example: Example (Neovim)

```
-- Lua equivalent for creating directories from a list
local function create_dirs_from_list()
  local lines = vim.fn.getline(1, '$')
  for _, line in ipairs(lines) do
    local dir = line:match('%w+')
    if dir then
      vim.fn.mkdir(dir, 'p')
    end
  end
end

-- Usage: Call create_dirs_from_list() in Neovim
```

Source: vim.fandom.com

28.17 Bulk Open Files from Filename List

Category: file_operations
Tags: file-management, macro, bulk-editing

Efficiently open multiple files from a text file containing filenames using Vim's built-in macro and quickfix list functionality

Example: Example (Vim)

```
" Method 1: Using macro
q0gf:rew<CR><DOWN-ARROW>q
@0

" Method 2: Using quickfix list
:setlocal efm=%f
:execute 'cfile '.expand('%')
```

Example: Example (Neovim)

```
-- Method 1: Using vim.cmd for macro
vim.cmd('q0gf:rew<CR><DOWN-ARROW>q')
vim.cmd('@0')

-- Method 2: Using quickfix list
vim.opt.errorformat = '%f'
vim.cmd('cfile ' .. vim.fn.expand('%'))

-- Additional Lua-friendly approach
local function open_files_from_list(filename)
  local files = {}
  for line in io.lines(filename) do
    table.insert(files, line)
  end
  for _, file in ipairs(files) do
    vim.cmd('edit ' .. file)
  end
end
```

Source: vim.fandom.com

28.18 Bulk Rename Files Using Vim

Category: file_operations
Tags: file-manipulation, shell-integration, bulk-rename

Efficiently rename multiple files by generating rename commands in Vim and executing them via shell

Example: Example (Vim)

```
# List files
:ls | vim -

# Rename to lowercase
:%s/.*/mv -i '&' \L'&'/g

# Execute rename commands
:w !sh
```

Example: Example (Neovim)

```
-- Equivalent in Neovim can use vim.fn and vim.cmd
-- Open file list
vim.cmd('read !ls')

-- Create rename commands
vim.cmd(':%s/.*/mv -i "&" \L"&"/g')

-- Execute shell commands
```

```
vim.cmd(':w !sh')
```

Source: vim.fandom.com

28.19 Capture Change Locations

Category: file_operations
Tags: change-tracking, file-export, logging

Redirect change list to a file for tracking editing history across sessions

Example: Example (Vim)

```
" Export change list to file
:set nomore
:redir > changes.txt
:changes
:redir END
:set more
:e changes.txt
```

Example: Example (Neovim)

```
-- Export change list to file (Lua equivalent)
vim.o.more = false
vim.api.nvim_command('redir! > changes.txt')
vim.api.nvim_command('changes')
vim.api.nvim_command('redir END')
vim.o.more = true
vim.cmd('edit changes.txt')
```

Source: vim.fandom.com

28.20 Central Backup Directory with Directory Creation

Category: file_operations
Tags: backup, file-management, directory-handling

Automatically create backup directories for files across different drives/paths

Example: Example (Vim)

```
let g:root_backup_dir = 'f:\\vim_backups'
" Create backup directory if it doesn't exist
if !filewritable(g:root_backup_dir)
  silent execute '!mkdir ' . g:root_backup_dir
endif
set backupdir=g:root_backup_dir
```

Example: Example (Neovim)

```
local root_backup_dir = vim.fn.expand('f:/vim_backups')

-- Ensure backup directory exists
if vim.fn.filewritable(root_backup_dir) == 0 then
  vim.fn.mkdir(root_backup_dir, 'p')
end

vim.o.backupdir = root_backup_dir
```

Source: vim.fandom.com

28.21 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: Example (Vim)

```
" In Vim script:
if filereadable('config.vim')
  source config.vim
endif

" Read file into list:
let lines = readfile('data.txt')
```

Source: Community contributed

28.22 Compile LaTeX with Rubber and Make in Vim

Category: file_operations
Tags: latex, compilation, quickfix

Easily compile LaTeX documents using Rubber and Make, with automatic error handling and quickfix integration

Example: Example (Vim)

```
if filereadable('Makefile')
  setlocal makeprg=make
else
  exec "setlocal makeprg=make\ -f\ ~/academic/tools/latex.mk\ " .
  ↪  substitute(bufname("%"),"tex$","pdf", "")
endif

setlocal errorformat=%f:%l:\ %m,%f:%l-%\d%\+:\ %m
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '*.tex',
  callback = function()
    if vim.fn.filereadable('Makefile') == 1 then
      vim.opt_local.makeprg = 'make'
    else
      vim.opt_local.makeprg = string.format('make -f
      ↪  ~/academic/tools/latex.mk %s',
        vim.fn.expand('%:r') .. '.pdf')
    end
    vim.opt_local.errorformat = '%f:%l: %m,%f:%l-%d: %m'
  end
})
```

Source: vim.fandom.com

28.23 Compress Vim Documentation Files

Category: file_operations
Tags: disk-space, documentation, compression

Compress Vim documentation files to save disk space while maintaining readability using gzip

Example: Example (Vim)

```
# Compress all .txt files in doc directory
:!gzip *.txt

# Uncompress help.txt separately
:!gunzip help.txt
```

Example: Example (Neovim)

```
-- Compress files using Lua system command
vim.fn.system('gzip ' .. vim.o.runtimepath .. '/doc/*.txt')
vim.fn.system('gunzip ' .. vim.o.runtimepath .. '/doc/help.txt')
```

Source: vim.fandom.com

28.24 Convert File Line Endings Easily

Category: file_operations
Tags: file-format, line-endings, conversion

Quickly convert file line endings between unix, dos, and mac formats with simple Vim commands

Example: Example (Vim)

```
:e ++ff=dos
:setlocal ff=unix
:w
```

Example: Example (Neovim)

```
-- Convert current file to unix line endings
vim.cmd('e ++ff=dos')
vim.cmd('setlocal ff=unix')
vim.cmd('w')
```

Source: vim.fandom.com

28.25 Convert OEM to ANSI Character Encoding

Category: file_operations
Tags: encoding, character-conversion, file-handling

Easily convert text files from OEM (DOS) to ANSI charset in Windows, ensuring proper character rendering

Example: Example (Vim)

```
:set encoding=cp437
:set fileencoding=latin1
:wq
```

Example: Example (Neovim)

```
-- Convert OEM to ANSI encoding
vim.opt.encoding = 'cp437'
vim.opt.fileencoding = 'latin1'
vim.cmd.wq()
```

Source: vim.fandom.com

28.26 Convert PDF to Text in Vim

Category: file_operations
Tags: pdf, external-tools, text-conversion

Create a custom command to read PDF contents into Vim buffer using pdftotext utility

Example: Example (Vim)

```
:command! -complete=file -nargs=1 Rpdf :r !pdftotext -nopgbrk <q-args> -
:command! -complete=file -nargs=1 Rpdf :r !pdftotext -nopgbrk <q-args> -
↪ |fmt -csw78
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Rpdf', function(opts)
  vim.cmd('read !pdftotext -nopgbrk ' .. opts.args .. ' -')
end, { nargs = 1, complete = 'file' })
```

Source: vim.fandom.com

28.27 Create Tar Archive of Open Buffers

Category: file_operations
Tags: buffer-management, file-archiving

Quickly create a tar archive of all files currently open in Vim buffers, preserving file paths

Example: Example (Vim)

```
:silent bufdo !tar -rvf archive.tar %:p
```

Example: Example (Neovim)

```
vim.api.nvim_command('silent bufdo !tar -rvf archive.tar ' ..
↪ vim.fn.expand('%:p'))
```

Source: vim.fandom.com

28.28 Create Timestamped Temporary File Copy

Category: file_operations
Tags: file-management, backup, mapping

Quickly create a timestamped copy of the current file in a temporary directory for safe editing

Example: Example (Vim)

```
:map zs :exe "sav $TMP/" . expand("%:t") . strftime("-%Y-%m-%d_%H%M%S")<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'zs', function()
  local filename = vim.fn.expand('%:t')
  local timestamp = os.date('%Y-%m-%d_%H%M%S')
  local tmp_dir = os.getenv('TMP') or '/tmp'
  local new_filename = string.format('%s/%s-%s', tmp_dir, filename,
  ↪  timestamp)
  vim.cmd('saveas ' .. new_filename)
end, { desc = 'Save file with timestamp in temp directory' })
```

Source: vim.fandom.com

28.29 Create Vim Patches Easily

Category: file_operations
Tags: diff, patch-creation, development

Technique for creating patches from source file modifications using diff command

Example: Example (Vim)

```
set patchmode=.orig
:!diff -u src/file.orig src/file > /tmp/file.diff
```

Example: Example (Neovim)

```
-- Set patch mode for automatic backups
vim.opt.patchmode = '.orig'

-- Example patch creation command (run in shell)
-- diff -u src/file.orig src/file > /tmp/file.diff
```

Source: vim.fandom.com

28.30 Create a Personal Knowledge Base in Vim

Category: file_operations
Tags: knowledge-management, file-navigation, organization

Efficiently manage a personal knowledge base using Vim's built-in file navigation and search capabilities

Example: Example (Vim)

```
" Use gf to jump between files
" Create lowercase files for consistent naming
" Use directories for large topics
```

Example: Example (Neovim)

```
-- Use vim.cmd('gf') to jump between files
-- Recommended: Create a consistent file naming strategy
-- Use directories to organize large topics

-- Example of setting up a knowledge base directory
vim.g.kb_path = '~/knowledge_base'

-- Function to quickly open or create a knowledge base file
vim.keymap.set('n', '<leader>kn', function()
  local filename = vim.fn.input('Enter knowledge base file name: ')
  vim.cmd('edit ' .. vim.g.kb_path .. '/' .. filename:lower())
end)
```

Source: vim.fandom.com

28.31 Create and Restore Vim Sessions Easily

Category: file_operations
Tags: session-management, workspace, productivity

Save and restore Vim editing sessions, including window layouts, open buffers, and project state with a single command

Example: Example (Vim)

```
" Save session
:mksession projectname.vis

" Restore session
gvim -S projectname.vis
```

Example: Example (Neovim)

```
-- Save session
vim.cmd('mksession projectname.vis')

-- Restore session
-- Can be done via command-line or in Lua
require('nvim-session-manager').save_current_session()
require('nvim-session-manager').load_session()
```

Source: vim.fandom.com

28.32 Create and Submit Vim Patches Easily

Category: file_operations
Tags: version-control, contributing, diff

Learn how to generate patches for Vim source files using diff and prepare them for submission to the vim-dev mailing list

Example: Example (Vim)

```
:set patchmode=.orig
```

Example: Example (Neovim)

```
vim.o.patchmode = '.orig'
```

Source: vim.fandom.com

28.33 Cross-Platform File Directory Opener

Category: file_operations
Tags: cross-platform, file-navigation, multi-os

Flexible function to open current file's directory across different platforms and GUI environments

Example: Example (Vim)

```
func! OpenCWD()
  if has("gui_running")
    if has("win32")
      !start explorer.exe %:p:h
    elseif has("mac") && has("unix")
      execute '!open ' . substitute(expand("%:p:h"), " ", "\\ ", "g")
    endif
  endif
endfunc
```

Example: Example (Neovim)

```
function _G.open_current_directory()
  local current_dir = vim.fn.expand('%:p:h')
  if vim.fn.has('win32') == 1 then
    vim.fn.system('start explorer.exe ' .. current_dir)
  elseif vim.fn.has('mac') == 1 then
    vim.fn.system('open ' .. current_dir:gsub(' ', '\\ '))
  end
end

vim.api.nvim_create_user_command('OpenCWD', _G.open_current_directory, {})
```

Source: vim.fandom.com

28.34 Customize Netrw Directory Listing Style

Category: file_operations
Tags: file-explorer, netrw, configuration

Set Netrw to display a tree-style directory listing, which provides a more comprehensive view of file structures

Example: Example (Vim)

```
let g:netrw_liststyle=3
```

Example: Example (Neovim)

```
vim.g.netrw_liststyle = 3
```

Source: vim.fandom.com

28.35 Decompile and View Java Class Files in Vim

Category: file_operations
Tags: java, decompiler, external-tools

Automatically decompile Java .class files when opening them in Vim using the JAD decompiler

Example: Example (Vim)

```
augroup class
  au!
  au bufreadpost,filereadpost *.class %!jad.exe -noctor -ff -i -p %
  au bufreadpost,filereadpost *.class set readonly
  au bufreadpost,filereadpost *.class set ft=java
  au bufreadpost,filereadpost *.class normal gg=G
  au bufreadpost,filereadpost *.class set nomodified
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('JavaClassDecompile', { clear = true })

vim.api.nvim_create_autocmd({'BufReadPost', 'FileReadPost'}, {
  group = 'JavaClassDecompile',
  pattern = '*.class',
  callback = function()
    vim.cmd('%!jad -noctor -ff -i -p %')
    vim.opt_local.readonly = true
    vim.opt_local.filetype = 'java'
    vim.cmd('normal! gg=G')
```

```
    vim.opt_local.modified = false
  end
})
```

Source: vim.fandom.com

28.36 Delete Current File and Close Buffer

Category: file_operations
Tags: buffer-management, file-delete

Quick command to delete the current file and close its buffer in one step

Example: Example (Vim)

```
call delete(expand('%')) | bdelete!
```

Example: Example (Neovim)

```
vim.fn.delete(vim.fn.expand('%')) vim.cmd('bdelete!')
```

Source: vim.fandom.com

28.37 Delete Files Directly from Vim

Category: file_operations
Tags: file-management, command-line

Create a custom command to delete files using Vim's delete() function, with filename completion and success/failure confirmation

Example: Example (Vim)

```
command! -complete=file -nargs=1 Remove :echo 'Remove: '.'<f-args>'.'
↪  '.(delete(<f-args>) == 0 ? 'SUCCEEDED' : 'FAILED')
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Remove', function(opts)
  local filename = opts.args
  local result = vim.fn.delete(filename)
  print(string.format('Remove: %s %s', filename, result == 0 and 'SUCCEEDED'
  ↪  or 'FAILED'))
end, { nargs = 1, complete = 'file' })
```

Source: vim.fandom.com

28.38 Drag and Drop Files into Vim Command Line

Category: file_operations
Tags: file-navigation, command-line, drag-and-drop

Easily insert file paths into Vim command line by drag and dropping from file explorer or file manager

Example: Example (Neovim)

```
-- Drag and drop files works natively in Vim/Neovim
-- In command line, type :edit and then drag file
-- Or simply drag file directly into Vim window
```

Source: vim.fandom.com

28.39 Easy File Encoding Switching

Category: file_operations
Tags: encoding, file-handling, utility

Create a function to quickly cycle through or select different file encodings when a file is loaded incorrectly

Example: Example (Vim)

```
let g:enc_index = 0
function! ChangeFileencoding()
  let encodings = ['cp1251', 'koi8-u', 'cp866']
  execute 'e ++enc='.encodings[g:enc_index].' %:p'
  if g:enc_index >=2
          let g:enc_index = 0
  else
          let g:enc_index = g:enc_index + 1
  endif
endf
nmap <F8> :call ChangeFileencoding()<CR>
```

Example: Example (Neovim)

```
local M = {}
M.enc_index = 0

function M.change_file_encoding()
  local encodings = {'cp1251', 'koi8-u', 'cp866'}
  vim.cmd('e ++enc=' .. encodings[M.enc_index + 1] .. ' %:p')

  M.enc_index = (M.enc_index >= 2) and 0 or M.enc_index + 1
end
```

```
vim.keymap.set('n', '<F8>', M.change_file_encoding, { desc = 'Cycle file
↪  encodings' })
```

Source: vim.fandom.com

28.40 Edit Files on FTP Server with Custom Port

Category: file_operations
Tags: remote-editing, ftp, file-access

Open files on an FTP server using a non-standard port by escaping the # character

Example: Example (Vim)

```
:e ftp://ftp.server\#2121/path/to/file/filename
```

Example: Example (Neovim)

```
-- Use netrw to open FTP files with custom port
vim.cmd('e ftp://ftp.server\#2121/path/to/file/filename')
```

Source: vim.fandom.com

28.41 Edit Files on Non-Standard Port FTP Server

Category: file_operations
Tags: remote-editing, ftp, file-access

Edit remote files on an FTP server using a non-standard port by escaping the port number

Example: Example (Vim)

```
:e ftp://ftp.server\#2121/path/to/file/filename
```

Example: Example (Neovim)

```
-- Neovim uses the same syntax as Vim for remote file editing
vim.cmd('e ftp://ftp.server\#2121/path/to/file/filename')
```

Source: vim.fandom.com

28.42 Edit Remote Files Seamlessly with Netrw

Category: file_operations
Tags: remote-editing, network, scp, file-transfer

Edit remote files locally using Vim's built-in network file transfer capabilities, supporting SCP, RCP, and FTP protocols without manual copying

Example: Example (Vim)

```
" Open remote file via SCP
:e scp://user@hostname//path/to/file
```

Example: Example (Neovim)

```
-- Open remote file via SCP in Neovim
vim.cmd('edit scp://user@hostname//path/to/file')
```

Source: vim.fandom.com

28.43 Edit Remote Files via Network Protocols

Category: file_operations
Tags: remote-editing, network, scp, file-transfer

Directly edit remote files using built-in Vim network protocols without manual file transfers

Example: Example (Vim)

```
vim scp://username@host//path/to/file
:Nread scp://username@host//path/to/file
:Nwrite scp://username@host//path/to/file
```

Example: Example (Neovim)

```
-- Open remote file
vim.cmd('edit scp://username@host//path/to/file')

-- Alternatively using Lua
vim.api.nvim_cmd({cmd = 'edit', args =
↪  {'scp://username@host//path/to/file'}}, {})
```

Source: vim.fandom.com

28.44 Edit Remote Files via SCP

Category: file_operations
Tags: remote-editing, network, ssh

Open and edit remote files directly through SCP using Vim's netrw plugin, specifying remote user and absolute path

Example: Example (Vim)

```
vim scp://remoteuser@server.tld//absolute/path/to/document

" Alternative methods:
:e scp://remoteuser@server.tld//absolute/path/to/document
:tabe scp://remoteuser@server.tld//absolute/path/to/document
```

Example: Example (Neovim)

```
-- Neovim supports the same SCP syntax
-- Open remote file in current window
vim.cmd('e scp://remoteuser@server.tld//absolute/path/to/document')

-- Open remote file in new tab
vim.cmd('tabe scp://remoteuser@server.tld//absolute/path/to/document')
```

Source: vim.fandom.com

28.45 Edit Remote Files with SCP in Vim

Category: file_operations
Tags: remote-editing, ssh, networking

Easily edit configuration files across multiple remote servers using Vim's netrw plugin and SCP protocol without multiple logins

Example: Example (Vim)

```
" Project file example for remote config editing
" Create a .project file with SCP paths
" Edit files across multiple servers transparently
```

Example: Example (Neovim)

```
-- Lua equivalent would use vim.fn.system() or a plugin like plenary.nvim
↪  for remote file handling
-- Example remote file opening
require('plenary.nvim').curl.get('scp://server1.domain.com///opt/myapp/etc/
↪  myapprc')
```

Source: vim.fandom.com

28.46 Encrypt Files Securely in Vim

Category: file_operations
Tags: encryption, security, file-management

Use :X command to encrypt files and prevent unauthorized access, with additional settings to minimize trace evidence

Example: Example (Vim)

```
" Encrypt file
:X

" Prevent leaving traces in viminfo
set viminfo='0,"/0,/0,:0,f0

" Disable swap and backup files
set noswapfile
set nobackup
set nowritebackup
```

Example: Example (Neovim)

```
-- Encryption is initiated interactively via :X command

-- Prevent trace evidence
vim.opt.viminfo = '0,"/0,/0,:0,f0'
vim.opt.swapfile = false
vim.opt.backup = false
vim.opt.writebackup = false
```

Source: vim.fandom.com

28.47 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: Example (Vim)

```
" Vimscript:
: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
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.readonly = true  -- make buffer read-only
vim.opt.modifiable = false  -- prevent any modifications
vim.opt.fileformat = 'unix'  -- set Unix line endings
vim.opt.fileformat = 'dos'   -- set DOS line endings
```

Source: Community contributed

28.48 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: Example (Vim)

```
: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
```

Source: Community contributed

28.49 Flexible File Find Function

Category: file_operations
Tags: file-search, custom-function, wildcard

Custom function to find files using partial name matches across directory tree

Example: Example (Vim)

```
function! Find(name)
  let l:_name = substitute(a:name, "\s", "*", "g")
  let l:list=system("find . -iname '*".l:_name."*' -not -name \"*.class\"
  ↪  -and -not -name \"*.swp\" | perl -ne 'print "$.\t$_"'")
  " ... rest of original function ...
endfunction
command! -nargs=1 Find :call Find("<args>")
```

Example: Example (Neovim)

```
function _G.flexible_file_find(name)
  local escaped_name = name:gsub('%s', '*')
  local cmd = string.format("find . -iname '*%s*' -not -name '*.class' -and
  ↪  -not -name '*.swp'", escaped_name)
  local result = vim.fn.system(cmd)
  -- Implement selection logic similar to Vimscript version
end

vim.api.nvim_create_user_command('Find', function(opts)
  _G.flexible_file_find(opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

28.50 Flexible Project File Search with Ignore Patterns

Category: file_operations
Tags: file-search, project-management, ignore-patterns

Create a file search function that allows configurable ignore patterns for files and directories

Example: Example (Vim)

```
function! Find(...)
    if a:0==2
        let path=a:1
        let query=a:2
    else
        let path="./"
        let query=a:1
    endif

    if !exists("g:FindIgnore")
        let ignore = ""
    else
        let ignore = " | egrep -v '".join(g:FindIgnore, "|")."'"
    endif

    let l:list=system("find ".path." -type f -iname '*".query."*'".ignore)
    let l:num=strlen(substitute(l:list, "[^\n]", "", "g"))

    if l:num < 1
        echo "'".query."' not found"
        return
    endif

    if l:num == 1
        exe "open " . substitute(l:list, "\n", "", "g")
    else
        let tmpfile = tempname()
```

```
        exe "redir! > " . tmpfile
        silent echon l:list
        redir END
        let old_efm = &efm
        set efm=%f

        if exists(":cgetfile")
            execute "silent! cgetfile " . tmpfile
        else
            execute "silent! cfile " . tmpfile
        endif

        let &efm = old_efm
        botright copen
        call delete(tmpfile)
    endif
endfunction

command! -nargs=* Find :call Find(<f-args>)
```

Example: Example (Neovim)

```
function _G.flexible_file_search(...)
    local path = './'
    local query

    if select('#', ...) == 2 then
        path = select(1, ...)
        query = select(2, ...)
    else
        query = select(1, ...)
    end

    local ignore_patterns = vim.g.FindIgnore or {}
    local ignore_cmd = ignore_patterns and #ignore_patterns > 0
        and ' | egrep -v "' .. table.concat(ignore_patterns, '|') .. '"' or
        ↪  ''

    local find_cmd = string.format('find %s -type f -iname "*%s*"%s', path,
    ↪  query, ignore_cmd)
    local result = vim.fn.system(find_cmd)
    local num_results = select(2, result:gsub('\n', '\n'))

    if num_results == 0 then
        print("'" .. query .. "' not found")
        return
    end

    if num_results == 1 then
        vim.cmd('edit ' .. result:gsub('\n', ''))
    else
        local tmpfile = vim.fn.tempname()
        vim.fn.writefile(vim.split(result, '\n'), tmpfile)
        vim.cmd('cfile ' .. tmpfile)
```

```
        vim.cmd('botright copen')
        vim.fn.delete(tmpfile)
    end
end

vim.api.nvim_create_user_command('Find', function(opts)
    _G.flexible_file_search(unpack(opts.fargs))
end, { nargs = '*' })
```

Source: vim.fandom.com

28.51 Flexible Vim File Launch Options

Category: file_operations
Tags: windows, file-launch, custom-options

Create shortcuts to launch Vim with specific modes like diff or read-only

Example: Example (Vim)

```
" Launch options for different file editing scenarios
" -d: Diff mode
" -R: Read-only mode
" -p: Open files in tabs
```

Example: Example (Neovim)

```
-- Lua approach to similar file launch configurations
-- Can be done via autocmds or file type associations
vim.api.nvim_create_autocmd('BufRead', {
  pattern = '*',
  callback = function()
    -- Custom file handling logic
  end
})
```

Source: vim.fandom.com

28.52 Generate C Source from Binary Files

Category: file_operations
Tags: binary-conversion, c-source, file-utility

Convert binary files into C-compatible byte arrays using xxd

Example: Example (Vim)

```
:r !xxd -i sample.bin
```

Example: Example (Neovim)

```
-- Generate C source array from binary file
vim.cmd('r !xxd -i sample.bin')
```

Source: vim.fandom.com

28.53 Handle Filenames with Spaces in Path

Category: file_operations
Tags: file-navigation, path-handling

Configure Vim to handle file paths with spaces by modifying isfname option

Example: Example (Vim)

```
" Allow spaces in filenames
set isfname+=32
```

Example: Example (Neovim)

```
-- Allow spaces in filenames
vim.opt.isfname:append(32)
```

Source: vim.fandom.com

28.54 Hex Dump Binary Files

Category: file_operations
Tags: hex-editing, binary-files, file-inspection

Convert binary files to hex dump for inspection and editing using xxd utility

Example: Example (Vim)

```
:r !xxd sample.bin
:%!xxd
:%!xxd -r
```

Example: Example (Neovim)

```
-- Read binary file as hex dump
vim.cmd('r !xxd sample.bin')

-- Convert buffer to hex mode
vim.cmd(':%!xxd')

-- Convert hex back to binary
```

```
vim.cmd(':%!xxd -r')
```

Source: vim.fandom.com

28.55 Hex Dump Binary Files with xxd

Category: file_operations
Tags: hex-editing, binary-files, utilities

Convert binary files to hex dump and back using Vim's built-in xxd utility, enabling hex editing of binary files

Example: Example (Vim)

```
" Convert file to hex dump
:%!xxd

" Convert hex dump back to binary
:%!xxd -r
```

Example: Example (Neovim)

```
-- Convert file to hex dump
vim.cmd(':%!xxd')

-- Convert hex dump back to binary
vim.cmd(':%!xxd -r')
```

Source: vim.fandom.com

28.56 Incremental Backups with Timestamped Files

Category: file_operations
Tags: backup, file-management, automation

Automatically create timestamped backup files in a central directory, preserving original file structure

Example: Example (Vim)

```
" Incremental backup configuration
augroup backup
    autocmd!
    autocmd BufWritePre,FileWritePre * let &l:backupext = '~' .
    ↪ strftime('%F') . '~'
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('incremental_backup', { clear = true })
vim.api.nvim_create_autocmd({'BufWritePre', 'FileWritePre'}, {
  group = 'incremental_backup',
  callback = function()
    vim.o.backupext = '~' .. os.date('%Y-%m-%d') .. '~'
  end
})
```

Source: vim.fandom.com

28.57 Incremental File Backups with Unique Timestamps

Category: file_operations
Tags: backup, file-management, timestamps

Automatically create unique, timestamped backup files when editing documents, ensuring you never lose previous versions

Example: Example (Vim)

```
set backupdir=c:\temp\vim_backup
let myvar = strftime("(%y%m%d)[%Hh%M]")
let myvar = "set backupext=_". myvar
execute myvar
```

Example: Example (Neovim)

```
vim.opt.backupdir = '/tmp/vim_backup'

-- Create backup with timestamp
local backup_ext = string.format("_(%s)[%s]",
  os.date("%y%m%d"),
  os.date("%Hh%M"))

vim.opt.backupext = backup_ext
```

Source: vim.fandom.com

28.58 Insert File or Command Output into Buffer

Category: file_operations
Tags: file-manipulation, command-execution

Quickly insert file contents or command output at cursor or specific buffer positions

Example: Example (Vim)

```
:r foo.txt    " Insert file below cursor
:0r foo.txt   " Insert file before first line
:r !ls        " Insert directory listing
:$r !pwd      " Insert current working directory
```

Example: Example (Neovim)

```
-- Insert file
vim.cmd('r foo.txt')

-- Insert command output
vim.cmd('r !ls')
vim.cmd('$r !pwd')
```

Source: vim.fandom.com

28.59 Insert Template Files into Buffer

Category: file_operations
Tags: file-templates, text-insertion, macros

Dynamically insert template files into the current buffer using a custom command that reads files specified by a special marker

Example: Example (Vim)

```
command! -range=% Refile <line1>,<line2>g/^#refile=/exe ":r " .
↪  strpart(getline("."), 8) | normal! kdd
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Refile', function(opts)
  local current_line = vim.fn.line('.')
  local last_line = opts.line2

  for line = current_line, last_line do
    local line_text = vim.fn.getline(line)
    if line_text:match('^#refile=') then
      local file_path = line_text:sub(9)
      vim.cmd('read ' .. file_path)
      vim.cmd('normal! kdd')
    end
  end
end, { range = true })
```

Source: vim.fandom.com

28.60 Insert current date

Category: File Operations
Tags: date, insert, command

Use `:r !date` to insert current date at cursor position.

Example: Example (Vim)

```
:r !date  " insert current date
```

Source: Community contributed

28.61 Launch Files with Default Application

Category: file_operations
Tags: windows, file-launch

Open the current file with its default system handler, similar to double-clicking in Windows Explorer

Example: Example (Vim)

```
nmap <Leader>x :silent ! start "1" "%:p"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>x', function()
  vim.cmd('silent !start "1" "' .. vim.fn.expand('%:p') .. '"')
end, { silent = true })
```

Source: vim.fandom.com

28.62 List Files in Current Directory Tree

Category: file_operations
Tags: file-listing, directory-navigation, custom-function

Create a function to recursively list all files in the current directory tree, displaying them in a new buffer for easy browsing and reference

Example: Example (Vim)

```
function! ListTree(dir)
  new
  set buftype=nofile
  set bufhidden=hide
```

```
  set noswapfile
  normal i.
  while 1
    let file = getline(".")
    if (file == '')
      normal dd
    elseif (isdirectory(file))
      normal dd
      let @" = glob(file . "/*")
      normal O
      normal P
      let @" = glob(file . "/.[^.]*")
      if (@" != '')
        normal O
        normal P
      endif
    else
      if (line('.') == line('$'))
        return
      else
        normal j
      endif
    endif
  endwhile
endfunction

map <Leader>L :call ListTree('.')<CR>
```

Example: Example (Neovim)

```
function _G.list_tree(dir)
  vim.cmd('new')
  vim.bo.buftype = 'nofile'
  vim.bo.bufhidden = 'hide'
  vim.bo.swapfile = false

  vim.fn.append(0, '.')
  vim.cmd('normal! dd')

  local function process_file(file)
    if vim.fn.isdirectory(file) == 1 then
      local files = vim.fn.glob(file .. '/*')
      local hidden_files = vim.fn.glob(file .. '/.[^.]*')

      vim.fn.append('$', vim.split(files, '\n'))
      if hidden_files ~= '' then
        vim.fn.append('$', vim.split(hidden_files, '\n'))
      end
    end
  end

  local current_line = 1
  while current_line <= vim.fn.line('$') do
    local file = vim.fn.getline(current_line)
```

```
    if file ~= '' then
      process_file(file)
    end
    current_line = current_line + 1
  end
end

vim.keymap.set('n', '<Leader>L', _G.list_tree, { desc = 'List files in
↪  current directory tree' })
```

Source: vim.fandom.com

28.63 Modifier Keys for Drag and Drop in Vim

Category: file_operations
Tags: file-navigation, window-management, shortcuts

Use modifier keys to control how files are opened when drag and dropped

Example: Example (Neovim)

```
-- Drag and drop behaviors with modifier keys:
-- Ctrl: Open file in new split window
-- Shift: Change to file's directory
-- Default: Open file in current window
```

Source: vim.fandom.com

28.64 Move Files Directly in Vim File Explorer

Category: file_operations
Tags: file-management, navigation, file-explorer

Use Vim's built-in file explorer to rename/move files without leaving the editor

Example: Example (Vim)

```
:Explore
:Sex
:Sex!
Press 'R' to rename file
```

Example: Example (Neovim)

```
-- Open file explorer
vim.cmd.Explore()

-- Split window and open explorer
```

```
vim.cmd.Sex()

-- Vertically split explorer
vim.cmd.Sex({'bang': true})

-- Note: Use 'R' key to rename/move files in explorer
```

Source: vim.fandom.com

28.65 Open All Files in Tabs by Default

Category: file_operations
Tags: tab-management, autocmd, workflow

Automatically open command-line arguments in separate tabs and create new tabs for each buffer

Example: Example (Vim)

```
autocmd VimEnter * tab all
autocmd BufAdd * exe 'tablast | tabe "' . expand('<afile>') . '"'
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('TabManagement', { clear = true })

vim.api.nvim_create_autocmd('VimEnter', {
  group = 'TabManagement',
  callback = function()
    vim.cmd('tab all')
  end
})

vim.api.nvim_create_autocmd('BufAdd', {
  group = 'TabManagement',
  callback = function()
    vim.cmd('tablast | tabe ' .. vim.fn.expand('<afile>'))
  end
})
```

Source: vim.fandom.com

28.66 Open Binary Files in Vim

Category: file_operations
Tags: binary-editing, file-modes

Open binary files with specific settings to prevent formatting issues

Example: Example (Vim)

```
" Open in binary mode
vim -b myfile.bin

" Display non-printable characters in hex
:setlocal display=uhex
:setlocal wrap
```

Example: Example (Neovim)

```
-- Open in binary mode
-- Use command line argument or set buffer options
vim.opt.binary = true
vim.opt.display = 'uhex'
vim.opt.wrap = true
```

Source: vim.fandom.com

28.67 Open Current File's Directory in File Explorer

Category: file_operations
Tags: file-navigation, explorer, windows

Quickly open the directory of the current file in Windows file explorer from Vim/Neovim

Example: Example (Vim)

```
map <C-e> :silent !explorer %:p:h:gs?/?\\\?<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-e>', function()
  vim.fn.system('explorer ' .. vim.fn.expand('%:p:h'):gsub('/', '\\'))
end, { silent = true })
```

Source: vim.fandom.com

28.68 Open Current Visual Studio File in Vim

Category: file_operations
Tags: external-tools, workflow, integration

Configure Visual Studio to open the current file in Vim with cursor position preserved

Example: Example (Vim)

```
Tools > External Tools > Add:
Title: &Vim
Command: C:\Vim\vim73\gvim.exe
Arguments: --servername gVimStudio --remote-silent +"execute 'normal!
↪  $(CurLine)G$(CurCol)|'" "$(ItemFileName)$(ItemExt)"
```

Example: Example (Neovim)

```
-- Lua equivalent would be to create a similar external tool configuration
-- This is mostly an IDE integration tip, so direct Lua translation isn't
↪  straightforward
```

Source: vim.fandom.com

28.69 Open File at Specific Line

Category: file_operations
Tags: cli-integration, file-opening

Launch Neovim and jump directly to a specific line number when opening a file

Example: Example (Vim)

```
vim file.txt +123
```

Example: Example (Neovim)

```
-- Same CLI behavior works in Neovim
-- Launch with: nvim file.txt +123
```

Source: vim.fandom.com

28.70 Open File with Clipboard Paste

Category: file_operations
Tags: clipboard, file-open, paste

Open a new file and immediately paste clipboard contents

Example: Example (Vim)

```
gvim.exe -c 'normal ggdG"*p' file.txt
```

Example: Example (Neovim)

```
-- Equivalent approach in Neovim
-- Can be implemented via command-line or lua function
vim.cmd('enew')
vim.cmd('normal "*p')
```

Source: vim.fandom.com

28.71 Open File's Directory in Vim

Category: file_operations
Tags: file-navigation, directory-browsing

Quickly open the directory of the current file using Netrw built-in commands

Example: Example (Vim)

```
" Netrw directory exploration commands
:Explore   " Open current file's directory
:Sexplore  " Split horizontally
:Vexplore  " Split vertically
```

Example: Example (Neovim)

```
-- Using Neovim's built-in commands
-- These are the same as Vim, but can be wrapped in Lua if needed
vim.cmd('Explore')    -- Open current file's directory
vim.cmd('Sexplore')   -- Split horizontally
vim.cmd('Vexplore')   -- Split vertically
```

Source: vim.fandom.com

28.72 Open Files in Existing Vim Instance

Category: file_operations
Tags: remote-editing, multi-instance, cli-integration

Quickly open files in an already running Vim instance without launching multiple windows

Example: Example (Vim)

```
gvim --remote first_file
gvim --remote +split first_file
```

Example: Example (Neovim)

```
-- Requires Vim with client-server enabled
-- Use shell command: nvim --remote first_file
-- Or create a function in init.lua
function open_in_existing_nvim(file)
  vim.fn.remote_send('<ESC>:e ' .. file .. '<CR>')
end
```

Source: vim.fandom.com

28.73 Open Files in New Tabs Across Desktop

Category: file_operations
Tags: file-handling, tabs, desktop-integration

Automatically open files in new tabs instead of new Vim instances, works with file managers and across desktops

Example: Example (Vim)

```
vim -p --remote-tab-silent filename
```

Example: Example (Neovim)

```
-- Equivalent in Neovim can be achieved via shell script or vim.cmd
-- Example shell script approach:
-- vim.cmd('silent tabedit ' .. filename)
```

Source: vim.fandom.com

28.74 Open Files in New Tabs Across Desktops

Category: file_operations
Tags: desktop-integration, file-management, tab-handling

Automatically open files in new tabs in the current Vim instance, even across different desktops

Example: Example (Vim)

```
gvim -p --remote-tab-silent %F
```

Example: Example (Neovim)

```
-- Lua equivalent for shell script
local function open_file_in_desktop_tab(file)
```

```
  local desktop = vim.fn.system('xprop -root -notype _NET_CURRENT_DESKTOP |
  ↪  perl -pe "s/.*?= (\d+)/$1/"'):gsub('\n', '')
  vim.cmd(string.format('edit ++tab +set\ servername=%s %s', desktop, file))
end
```

Source: vim.fandom.com

28.75 Open Files in New Tabs Automatically

Category: file_operations
Tags: tabs, file-management, windows

Automatically open new files in tabs using an autocmd, which is useful for maintaining a clean workspace when opening multiple files

Example: Example (Vim)

```
autocmd BufReadPost * tab ball

" Conditional version for diff mode
if (&diff==0)
    :autocmd BufReadPost * tab ball
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    if vim.o.diff == false then
      vim.cmd('tab ball')
    end
  end
})
```

Source: vim.fandom.com

28.76 Open Files in New Tabs via Remote Command

Category: file_operations
Tags: windows, cli, file-management

Use remote tab silent command to open files in tabs from command line or file associations

Example: Example (Vim)

```
"C:\Program Files\Vim\vim72\gvim.exe" --remote-tab-silent "%1"
```

Example: Example (Neovim)

```
-- Equivalent in Neovim terminal or external script
vim.fn.system({'nvim', '--remote-tab-silent', vim.fn.expand('%')})
```

Source: vim.fandom.com

28.77 Open Files in Tabs Across Multiple Instances

Category: file_operations
Tags: file-handling, tabs, windows

Automatically open files in new tabs within the same Vim/Neovim instance using remote tab commands

Example: Example (Vim)

```
gvim.exe --remote-tab-silent "%1"
```

Example: Example (Neovim)

```
-- For Neovim, this would typically be handled by terminal or file manager
↪  integration
-- Can be configured in init.lua with autocmds or external scripts
```

Source: vim.fandom.com

28.78 Open Files in Tabs from Command Line

Category: file_operations
Tags: cli, tabs, file-opening

Use -p flag to open multiple files, each in its own tab when launching Vim

Example: Example (Vim)

```
# Command line usage: gvim -p *.py
```

Example: Example (Neovim)

```
-- Use the same command-line flag when launching Neovim
-- nvim -p *.py
```

Source: vim.fandom.com

28.79 Open Files in Tabs with Path Search

Category: file_operations
Tags: tabs, file-search, path

Configure Vim to search paths when opening files in new tabs, including subdirectories

Example: Example (Vim)

```
" Configure path for file searching
set path=.,,**

" Use :tabfind to open files by searching path
:tabfind myfile.txt
```

Example: Example (Neovim)

```
-- Configure path for file searching
vim.opt.path = '.,,**'

-- Note: :tabfind is still a command, can be executed via vim.cmd
```

Source: vim.fandom.com

28.80 Open Files via URLs in Vim

Category: file_operations
Tags: file-handling, autocmd, url-parsing

Allows opening files from URLs, specifically file:/// URLs on Windows by using an autocmd to handle file URL parsing

Example: Example (Vim)

```
au BufReadCmd file:///* exe "bd!|edit
↪  ".substitute(expand("<afile>"),"file:/*","","")
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufReadCmd', {
  pattern = 'file:///*',
  callback = function()
    local file_path = vim.fn.substitute(vim.fn.expand('<afile>'), 'file:/*',
    ↪  '', '')
    vim.cmd('bd!')
    vim.cmd('edit ' .. file_path)
  end
})
```

Source: vim.fandom.com

28.81 Open Files with Wildcard Patterns

Category: file_operations
Tags: file-management, wildcard, globbing

Use wildcard patterns to open multiple files matching a specific pattern

Example: Example (Vim)

```
gvim *\2*\xx.txt
```

Example: Example (Neovim)

```
vim.cmd('args *\2*\xx.txt')
```

Source: vim.fandom.com

28.82 Open Finder for macOS Users

Category: file_operations
Tags: file-navigation, macos, finder

Quickly open Finder at the directory of the current buffer

Example: Example (Vim)

```
nmap <Leader>f :!open %:p:h<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>f', function()
  vim.fn.system('open ' .. vim.fn.expand('%:p:h'))
end, { desc = 'Open Finder in current directory' })
```

Source: vim.fandom.com

28.83 Open Multiple Files from List File

Category: file_operations
Tags: file-management, command-line, bulk-editing

Efficiently open multiple files specified in a list file using Vim command-line arguments

Example: Example (Vim)

```
gvim "+gl/^/exec 'badd '.getline('.')" +bdel +nohls +"cd %:h" LISTFILE
```

Example: Example (Neovim)

```
vim.cmd('args ' .. vim.fn.expand('mylist'))
```

Source: vim.fandom.com

28.84 Open Multiple Files in Same Vim Instance on macOS

Category: file_operations
Tags: macos, applescript, file-management

Use AppleScript to open multiple files in the same Vim window on macOS, avoiding multiple Vim instances

Example: Example (Vim)

```
on run argv
   tell application "Vim"
      repeat with n from 1 to (count of argv)
         set theUnixPath to item n of argv
         set theMacPath to (POSIX file theUnixPath) as string
         open file theMacPath
      end repeat
      activate
   end tell
end run
```

Example: Example (Neovim)

```
-- Lua equivalent would typically use Neovim's built-in commands
-- For macOS file opening, you might use vim.cmd or external script
vim.cmd('args ' .. table.concat(files, ' '))
```

Source: vim.fandom.com

28.85 Open Multiple Files in Tabs Automatically

Category: file_operations
Tags: file-handling, tabs, windows

Automatically open files in new tabs when launched from Windows, using remote tab functionality

Example: Example (Vim)

```
"C:\Program Files\Vim\vim72\gvim.exe" --remote-tab-silent "%1"
```

Example: Example (Neovim)

```
-- Neovim equivalent (typically configured via file associations)
-- Use :edit or :tabedit to open files in new tabs
vim.keymap.set('n', '<leader>nt', ':tabedit ', { desc = 'Open new tab' })
```

Source: vim.fandom.com

28.86 Open Windows Explorer from Current Buffer

Category: file_operations
Tags: file-navigation, windows, explorer

Quickly open Windows Explorer at the directory of the current buffer, optionally selecting the current file

Example: Example (Vim)

```
" Open Explorer in current buffer's directory
nmap <F11> :!start explorer /e,,%:p:h<CR>
imap <F11> <Esc><F11>

" Open Explorer and select current file
nmap <F11> :!start explorer /select,%:p
```

Example: Example (Neovim)

```
-- Open Explorer in current buffer's directory
vim.keymap.set('n', '<F11>', function()
  vim.fn.system('start explorer /e,,' .. vim.fn.expand('%:p:h'))
end, { desc = 'Open Explorer in current directory' })

-- Open Explorer and select current file
vim.keymap.set('n', '<F11>', function()
  vim.fn.system('start explorer /select,' .. vim.fn.expand('%:p'))
end, { desc = 'Open Explorer and select current file' })
```

Source: vim.fandom.com

28.87 Open or Jump to File in New Tab

Category: file_operations
Tags: file-handling, tabs, navigation

Use :drop to edit a file or jump to its existing tab/window, with option to open in new tab

Example: Example (Vim)

```
" Open file in new tab or jump to existing
:tab drop filename.txt

" Custom command to simplify file opening
command! -nargs=1 -complete=file O tab drop <args>
```

Example: Example (Neovim)

```
-- Open file in new tab or jump to existing
vim.cmd('tab drop ' .. vim.fn.expand('%:p'))

-- Create custom command in Lua
vim.api.nvim_create_user_command('O', function(opts)
  vim.cmd('tab drop ' .. opts.args)
end, { nargs = 1, complete = 'file' })
```

Source: vim.fandom.com

28.88 Optimize Large File Loading Performance

Category: file_operations
Tags: performance, file-handling, optimization

Automatically adjust Vim settings for large files to improve loading speed and reduce memory usage

Example: Example (Vim)

```
" file is large from 10mb
let g:LargeFile = 1024 * 1024 * 10
augroup LargeFile
  au!
  autocmd BufReadPre * let f=getfsize(expand("<afile>")) | if f >
  ↪  g:LargeFile || f == -2 | call LargeFile() | endif
augroup END

function! LargeFile()
 set eventignore+=FileType
 setlocal bufhidden=unload
 setlocal buftype=nowrite
 setlocal undolevels=-1
 autocmd VimEnter * echo "The file is larger than " . (g:LargeFile / 1024 /
 ↪  1024) . " MB, so some options are changed"
endfunction
```

Example: Example (Neovim)

```
vim.g.LargeFile = 1024 * 1024 * 10

vim.api.nvim_create_augroup('LargeFile', { clear = true })

vim.api.nvim_create_autocmd('BufReadPre', {
  group = 'LargeFile',
  callback = function()
    local file_size = vim.fn.getfsize(vim.fn.expand('<afile>'))
    if file_size > vim.g.LargeFile or file_size == -2 then
      vim.opt.eventignore:append('FileType')
      vim.bo.bufhidden = 'unload'
      vim.bo.buftype = 'nowrite'
      vim.bo.undolevels = -1

      vim.notify(string.format('The file is larger than %d MB, so some
      ↪  options are changed', vim.g.LargeFile / 1024 / 1024))
    end
  end
})
```

Source: vim.fandom.com

28.89 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: Example (Vim)

```
" Convert backslashes to forward slashes:
:%s/\\/\//g

" Convert forward slashes to backslashes:
:%s/\//\\/g

" Or use built-in function:
:echo substitute(@%, '\\', '/', 'g')
```

Source: Community contributed

28.90 Preserve Missing End-of-Line When Saving Files

Category: file_operations
Tags: file-formatting, eol, file-saving

Automatically preserve files without an end-of-line character, maintaining the original file format across different operating systems

Example: Example (Vim)

```
" Preserve noeol (missing trailing eol) when saving file
augroup automatic_noeol
  au!
  au BufWritePre  * call <SID>TempSetBinaryForNoeol()
  au BufWritePost * call <SID>TempRestoreBinaryForNoeol()
augroup END

function! s:TempSetBinaryForNoeol()
  let s:save_binary = &binary
  if ! &eol && ! &binary
    let s:save_view = winsaveview()
    setlocal binary
    if &ff == "dos" || &ff == "mac"
      if line('$') > 1
        undojoin | exec "silent 1,$-1normal! A\<C-V>\<C-M>"
      endif
    endif
    if &ff == "mac"
      undojoin | %join!
    endif
  endif
endfunction

function! s:TempRestoreBinaryForNoeol()
  if ! &eol && ! s:save_binary
    if &ff == "dos"
      if line('$') > 1
        silent! undojoin | silent 1,$-1s/\r$//e
      endif
    elseif &ff == "mac"
      silent! undojoin | silent %s/\r/\r/ge
    endif
    setlocal nobinary
    call winrestview(s:save_view)
  endif
endfunction
```

Example: Example (Neovim)

```
local function temp_set_binary_for_noeol()
  local save_binary = vim.o.binary
  if not vim.o.eol and not vim.o.binary then
    local save_view = vim.fn.winsaveview()
    vim.o.binary = true
    local fileformat = vim.o.ff

    if fileformat == 'dos' or fileformat == 'mac' then
      if vim.fn.line('$') > 1 then
        vim.cmd('undojoin')
        vim.cmd('silent 1,$-1normal! A\<C-V>\<C-M>')
      end
    end
```

```
    if fileformat == 'mac' then
      vim.cmd('undojoin')
      vim.cmd('%join!')
    end

    return save_view, save_binary
  end
  return nil, save_binary
end

local function temp_restore_binary_for_noeol(save_view, save_binary)
  if save_view and not vim.o.eol and not save_binary then
    local fileformat = vim.o.ff

    if fileformat == 'dos' then
      if vim.fn.line('$') > 1 then
        vim.cmd('silent! undojoin')
        vim.cmd('silent 1,$-1s/\r$//e')
      end
    elseif fileformat == 'mac' then
      vim.cmd('silent! undojoin')
      vim.cmd('silent %s/\r/\r/ge')
    end

    vim.o.binary = false
    vim.fn.winrestview(save_view)
  end
end

-- Autocmd setup
vim.api.nvim_create_augroup('automatic_noeol', { clear = true })
vim.api.nvim_create_autocmd('BufWritePre', {
  group = 'automatic_noeol',
  callback = function()
    local save_view, save_binary = temp_set_binary_for_noeol()
    vim.b.noeol_save_view = save_view
    vim.b.noeol_save_binary = save_binary
  end
})

vim.api.nvim_create_autocmd('BufWritePost', {
  group = 'automatic_noeol',
  callback = function()
    temp_restore_binary_for_noeol(vim.b.noeol_save_view,
    ↪  vim.b.noeol_save_binary)
  end
})
```

Source: vim.fandom.com

28.91 Preserve Modified Flag After Writing to File

Category: file_operations
Tags: file-writing, buffer-management, autocmds

Prevents the modified flag from being reset when writing to a different file, maintaining the original buffer's modification state

Example: Example (Vim)

```
function! BufWrite()
  let fileName = expand('<afile>')
  if fileName =~ 'ftp://\|rcp://\|scp://\|dav://\|rync://\|sftp://'
    return
  endif
  let _modified = &modified
  exec 'w'.(v:cmdbang?'!':'') v:cmdarg fileName
  if expand('%') !=# fileName
    let &modified = _modified
  endif
endfunction
```

Example: Example (Neovim)

```
local function buf_write()
  local file_name = vim.fn.expand('<afile>')
  local protocol_pattern =
  ↪  'ftp://\|rcp://\|scp://\|dav://\|rync://\|sftp://'

  if file_name:match(protocol_pattern) then
    return
  end

  local was_modified = vim.o.modified
  vim.cmd('write' .. (vim.v.cmdbang and '!' or '') .. ' ' .. vim.v.cmdarg ..
  ↪  ' ' .. file_name)

  if vim.fn.expand('%') ~= file_name then
    vim.o.modified = was_modified
  end
end

-- Set up autocmd
vim.api.nvim_create_autocmd('BufWriteCmd', {
  callback = buf_write
})
```

Source: vim.fandom.com

28.92 Project File Search with Wildmenu

Category: file_operations
Tags: file-search, navigation, ui-enhancement

Enable wildmenu to display multiple file matches when using :find, allowing easy file selection

Example: Example (Vim)

```
set wildmenu
set path=$PWD/**
```

Example: Example (Neovim)

```
vim.opt.wildmenu = true
vim.opt.path = vim.fn.getcwd() .. '/**'
```

Source: vim.fandom.com

28.93 Protect Reference Files from Accidental Changes

Category: file_operations
Tags: file-protection, read-only, safety

Prevent accidental modifications to reference or comparison files by setting them as non-modifiable and changing the color scheme for visual distinction

Example: Example (Vim)

```
:set nomodifiable
:colorscheme peachpuff
```

Example: Example (Neovim)

```
vim.opt.modifiable = false
vim.cmd.colorscheme('peachpuff')
```

Source: vim.fandom.com

28.94 Quick Backup of Current File with Timestamp

Category: file_operations
Tags: backup, file-management, timestamp

Quickly create a timestamped backup of the current file before making modifications, preserving the original file's view and extension

Example: Example (Vim)

```
map ;s :up | saveas!
↪  %:p:r-<C-R>=strftime("%y%m%d-%H:%M")<CR>-bak.<C-R>=expand("%:e")<CR> |
↪  3sleep | e #<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', ';s', function()
  -- Update current file
  vim.cmd('update')

  -- Create timestamped backup with original extension
  local original_file = vim.fn.expand('%:p')
  local timestamp = os.date('%y%m%d-%H:%M')
  local backup_file = string.format('%s-%s-bak.%s',
    vim.fn.fnamemodify(original_file, ':r'),
    timestamp,
    vim.fn.expand('%:e'))

  -- Save backup and return to original file
  vim.cmd('saveas! ' .. backup_file)
  vim.cmd('sleep 3')
  vim.cmd('edit #')
end, { desc = 'Create timestamped backup' })
```

Source: vim.fandom.com

28.95 Quick Directory Exploration in Vim

Category: file_operations
Tags: navigation, file-explorer, directory

Easily explore directories and navigate file system directly from Vim using built-in commands

Example: Example (Vim)

```
:Explore
:e /home/user
:e ..
:e <dir_path>
```

Example: Example (Neovim)

```
-- Open file explorer
vim.cmd('Explore')

-- Open specific directory
vim.cmd('e /home/user')

-- Go to parent directory
vim.cmd('e ..')
```

Source: vim.fandom.com

28.96 Quick File Backup Mapping

Category: file_operations
Tags: backup, key-mapping, file-handling

Easily create a backup of the current file with a single keypress, temporarily enabling backup option

Example: Example (Vim)

```
" Quick backup mapping
nnoremap <Leader>b :let x=&backup<Bar>set backup<Bar>write<Bar>let
↪  &backup=x<Bar>unlet x<Cr>
```

Example: Example (Neovim)

```
-- Lua equivalent for quick file backup
vim.keymap.set('n', '<Leader>b', function()
  local backup_orig = vim.o.backup
  vim.o.backup = true
  vim.cmd.write()
  vim.o.backup = backup_orig
end, { desc = 'Quick file backup' })
```

Source: vim.fandom.com

28.97 Quick File Copy Within Vim

Category: file_operations
Tags: file-management, command, productivity

Create a custom command to easily copy the current file to another location without leaving Vim

Example: Example (Vim)

```
command Cpage silent !cp '%:p' "c:/Progra~1/Tomcat/webapps/console/pages/%"
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Cpage', function()
  local current_file = vim.fn.expand('%:p')
  local dest_path = 'c:/Progra~1/Tomcat/webapps/console/pages/'
  vim.fn.system('cp "' .. current_file .. '" "' .. dest_path .. '"')
end, {})
```

Source: vim.fandom.com

28.98 Quick File Encoding Detection in Bash

Category: file_operations
Tags: bash, encoding, file-detection

Create a Bash alias to quickly detect file encoding using Vim without fully opening the file

Example: Example (Vim)

```
alias vimenc='vim -c \'let $enc = &fileencoding | execute "!echo Encoding:
↪  $enc" | q\''

# Usage: vimenc filename.xml
```

Example: Example (Neovim)

```
-- In your shell config (e.g., .bashrc or .zshrc)
-- Use the same alias as in Vimscript
-- Or create a Lua-based function
function _G.detect_file_encoding(filename)
  local encoding = vim.fn.readfile(filename, '', 1)
  print('Encoding: ' .. vim.o.fileencoding)
end
```

Source: vim.fandom.com

28.99 Quick File Extension Manipulation

Category: file_operations
Tags: file-management, command-line

Easily add or change file extensions using custom commands that leverage Vim's expand() function

Example: Example (Vim)

```
command! -nargs=1 AddExt execute "saveas ".expand("%:p").<q-args>
command! -nargs=1 ChgExt execute "saveas ".expand("%:p:r").<q-args>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('AddExt', function(opts)
  vim.cmd('saveas ' .. vim.fn.expand('%:p') .. opts.args)
end, { nargs = 1 })

vim.api.nvim_create_user_command('ChgExt', function(opts)
  vim.cmd('saveas ' .. vim.fn.expand('%:p:r') .. opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

28.100 Quick File and Editing Commands

Category: file_operations
Tags: file-editing, command-line, save-quit

Efficient commands for editing, saving, and quitting files

Example: Example (Neovim)

```
-- File operations
vim.keymap.set('n', '<leader>q', ':q!<CR>', { desc = 'Quit without saving'
↪  })
vim.keymap.set('n', 'ZZ', ':wq<CR>', { desc = 'Save and quit' })
vim.keymap.set('n', ':e!', ':e!<CR>', { desc = 'Reload file' })
```

Source: vim.fandom.com

28.101 Quick HTML File Preview in Browser

Category: file_operations
Tags: html, preview, browser-integration

Easily preview HTML files in a browser directly from Vim/Neovim with cross-platform support

Example: Example (Vim)

```
" Open current HTML file in default browser
nnoremap <F5> :update<Bar>silent !xdg-open %:p &<CR>

" Open URL under cursor in browser
nnoremap <F8> :silent !xdg-open <cfile> &<CR>
```

Example: Example (Neovim)

```
-- Open current file in default browser
vim.keymap.set('n', '<F5>', function()
  vim.cmd.update()
  vim.fn.system({'xdg-open', vim.fn.expand('%:p')} )
end, { desc = 'Preview current file in browser' })

-- Open URL under cursor in browser
vim.keymap.set('n', '<F8>', function()
  vim.fn.system({'xdg-open', vim.fn.expand('<cfile>')})
end, { desc = 'Open URL under cursor' })
```

Source: vim.fandom.com

28.102 Quick Note Taking with Vim Alias

Category: file_operations
Tags: alias, note-taking, command-line

Create a shell alias to open a notes file, go to the last line, add a blank line, and start insert mode quickly

Example: Example (Vim)

```
alias note="gvim -c $ +foldopen +put_ +startinsert ~/path-to-file/notes.md"
```

Example: Example (Neovim)

```
-- Lua equivalent for Neovim
-- In your shell configuration or init.lua
vim.api.nvim_create_user_command('Note', function()
  vim.cmd('edit ~/path-to-file/notes.md')
  vim.cmd('normal! G')
  vim.cmd('normal! o')
  vim.cmd('startinsert')
end, {})
```

Source: vim.fandom.com

28.103 Quick Source/Header File Switching

Category: file_operations
Tags: navigation, file-switching, programming

Easily switch between source and header files in C/C++ projects with a flexible function that handles multiple file extensions

Example: Example (Vim)

```
function! SwitchSourceHeader()
  if (expand("%:e") == "cpp")
    find %:t:r.h
  else
    find %:t:r.cpp
  endif
endfunction

nmap ,s :call SwitchSourceHeader()<CR>
```

Example: Example (Neovim)

```
function _G.switch_source_header()
  local ext = vim.fn.expand('%:e')
  if ext == 'cpp' then
    vim.cmd('find ' .. vim.fn.expand('%:t:r') .. '.h')
  else
    vim.cmd('find ' .. vim.fn.expand('%:t:r') .. '.cpp')
  end
end

vim.keymap.set('n', ',s', _G.switch_source_header, { desc = 'Switch between
↪  source and header' })
```

Source: vim.fandom.com

28.104 Quick Vim Directory Navigation

Category: file_operations
Tags: navigation, windows, file-explorer

Create a Vim shortcut that opens directly to root directory for quick file access without using Windows Explorer

Example: Example (Vim)

```
"C:\Program Files\gvim\vim62\gvim.exe" "c:/"
```

Example: Example (Neovim)

```
-- Lua equivalent for launching Neovim in root directory
vim.fn.system('nvim c:/')
```

Source: vim.fandom.com

28.105 Quickly Find and Edit Files Across Large Codebases

Category: file_operations
Tags: file-navigation, tags, search

Generate a tag file for quickly finding and opening files in large directory structures using ctags or custom find scripts

Example: Example (Vim)

```
set tags=c:\files.tags

" Use :tj Foo<tab> to autocomplete and open files
```

Example: Example (Neovim)

```
-- Equivalent in Neovim
vim.opt.tags = 'c:\\files.tags'

-- Use built-in ctags functionality
-- Generate tags: ctags -R --file-tags=yes c:/
```

Source: vim.fandom.com

28.106 Quickly Switch Between Source and Header Files

Category: file_operations
Tags: file-navigation, c-development, productivity

Easily toggle between corresponding source and header files in C/C++ projects

Example: Example (Vim)

```
" Map key combination to switch between .c/.cpp and .h files
map <F4> :e %:p:s,.h$,.X123X,:s,.cpp$,.h,:s,.X123X$,.cpp,<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', function()
  local current_file = vim.fn.expand('%:p')
  local new_file = current_file:gsub('\.h$', '.X123X'):gsub('\.cpp$',
  ↪  '.h'):gsub('\.X123X$', '.cpp')
  vim.cmd.edit(new_file)
end, { desc = 'Switch between source and header' })
```

Source: vim.fandom.com

28.107 Recover Files After Crash or Power Failure

Category: file_operations
Tags: file-recovery, crash-recovery, backup

Easily recover unsaved files after system crashes using Vim's built-in recovery mode

Example: Example (Vim)

```
vim -r <filename>  # Recover specific file
vim -r ""  # Recover unnamed file
vim -r  # List recoverable files
```

Example: Example (Neovim)

```
-- In terminal/command mode
-- Use same commands as Vimscript
-- Recommended to use :help recovery in Neovim for more details
```

Source: vim.fandom.com

28.108 Recover Overwritten File Using Swap File

Category: file_operations
Tags: file-recovery, swap-file, backup

Recover a file that was accidentally overwritten by using Vim's built-in recover command when a swap file exists

Example: Example (Vim)

```
:recover
```

Example: Example (Neovim)

```
-- In Neovim, use the same command
vim.cmd('recover')
```

Source: vim.fandom.com

28.109 Recursive File Finding in Vim

Category: file_operations
Tags: file-search, navigation, project-management

Create a custom function to find files recursively in subdirectories with interactive selection

Example: Example (Vim)

```
function! Find(name)
  let l:list=system("find . -name '".a:name."' | perl -ne 'print
  ↪  \"$.\        $_"'")
  let l:num=strlen(substitute(l:list, "[^\n]", "", "g"))
  if l:num < 1
    echo "'".a:name."' not found"
    return
  endif
  if l:num != 1
    echo l:list
    let l:input=input("Which ? (CR=nothing)\n")
    if strlen(l:input)==0
```

```
      return
    endif
    let l:line=matchstr("\n".l:list, "\n".l:input."\        [^\n]*")
  else
    let l:line=l:list
  endif
  let l:line=substitute(l:line, "^[^\        ]*\        ./", "", "")
  execute ":e ".l:line
endfunction
command! -nargs=1 Find :call Find("<args>")
```

Example: Example (Neovim)

```
function _G.find_file(name)
  local list = vim.fn.system('find . -name "' .. name .. '" | perl -ne
  ↪  "print qq{$.\t$_}"')
  local num = #vim.split(list, '\n')

  if num < 1 then
    print("'" .. name .. "' not found")
    return
  end

  if num > 1 then
    print(list)
    local input = vim.fn.input('Which ? (CR=nothing)\n')

    if input == '' then
      return
    end

    local selected_line = vim.split(list, '\n')[tonumber(input)]
    vim.cmd('edit ' .. selected_line:gsub('^%d+\        %./', ''))
  else
    vim.cmd('edit ' .. list:gsub('^%d+\        %./', ''))
  end
end

vim.api.nvim_create_user_command('Find', function(opts)
  _G.find_file(opts.args)
end, { nargs = 1 })
```

Source: vim.fandom.com

28.110 Reload File with Different Encoding

Category: file_operations
Tags: encoding, file-handling, charset

Quickly reload a file using a specific character encoding when Vim fails to detect the correct one

Example: Example (Vim)

```
" Reload file with specific encoding
:e ++enc=utf-8

" Quick mapping to switch to UTF-8
nnoremap <F12> :e ++enc=utf-8<CR>
```

Example: Example (Neovim)

```
-- Reload file with specific encoding
vim.cmd('e ++enc=utf-8')

-- Quick mapping to switch to UTF-8
vim.keymap.set('n', '<F12>', ':e ++enc=utf-8<CR>', { desc = 'Reload file as
↪  UTF-8' })
```

Source: vim.fandom.com

28.111 Reload file from disk

Category: File Operations
Tags: reload, file, refresh

Use `:e!` to reload current file from disk, discarding unsaved changes.

Example: Example (Vim)

```
:e!  " reload file from disk
```

Source: Community contributed

28.112 Restore Last Edited File and Cursor Position

Category: file_operations
Tags: session-management, cursor-restoration, startup

Automatically open the last edited file and restore cursor position when launching Vim/Neovim without arguments

Example: Example (Vim)

```
" Go to last file if invoked without arguments
autocmd VimEnter * nested if
  \ argc() == 0 &&
  \ bufname("%") == "" &&
  \ bufname(2) != "" |
  \   exe "normal! `0" |
  \ endif
```

```
" Restore cursor position
autocmd BufReadPost *
  \ if line("'\"") > 1 && line("'\"") <= line("$") |
  \   exe "normal! g`\"" |
  \ endif
```

Example: Example (Neovim)

```
-- Restore last edited file and cursor position
vim.api.nvim_create_autocmd('VimEnter', {
  nested = true,
  callback = function()
    if vim.fn.argc() == 0 and vim.fn.bufname('%') == '' and
    ↪  vim.fn.bufname(2) ~= '' then
      vim.cmd('normal! `0')
    end
  end
})

-- Restore cursor position
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    local last_pos = vim.fn.line('\'"')
    if last_pos > 1 and last_pos <= vim.fn.line('$') then
      vim.cmd('normal! g`"')
    end
  end
})
```

Source: vim.fandom.com

28.113 Save and Restore Session Automatically

Category: file_operations
Tags: session-management, persistence

Automatically save and restore Vim/Neovim sessions when launching without arguments

Example: Example (Vim)

```
" Save session on exit
autocmd VimLeave * nested if (!isdirectory($HOME . "/.vim")) |
    \ call mkdir($HOME . "/.vim") |
    \ endif |
    \ execute "mksession! " . $HOME . "/.vim/Session.vim"

" Restore session on startup
autocmd VimEnter * nested if argc() == 0 && filereadable($HOME .
↪  "/.vim/Session.vim") |
```

```
    \ execute "source " . $HOME . "/.vim/Session.vim"
```

Example: Example (Neovim)

```
-- Save session on exit
vim.api.nvim_create_autocmd('VimLeave', {
  nested = true,
  callback = function()
    local vim_dir = vim.fn.expand('$HOME') .. '/.vim'
    if vim.fn.isdirectory(vim_dir) == 0 then
      vim.fn.mkdir(vim_dir, 'p')
    end
    vim.cmd('mksession! ' .. vim_dir .. '/Session.vim')
  end
})

-- Restore session on startup
vim.api.nvim_create_autocmd('VimEnter', {
  nested = true,
  callback = function()
    local session_file = vim.fn.expand('$HOME') .. '/.vim/Session.vim'
    if vim.fn.argc() == 0 and vim.fn.filereadable(session_file) == 1 then
      vim.cmd('source ' .. session_file)
    end
  end
})
```

Source: vim.fandom.com

28.114 Save as

Category: File
Tags: save, file

Use `:sav[eas] filepath` to save file under a different name

Example: Example (Vim)

```
:sav ~/tmp/work.txt
```

Source: Community contributed

28.115 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: Example (Vim)

```
:w              " save current file
:w newfile.txt " save as new file
:wall           " save all files
```

Source: Community contributed

28.116 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: Example (Vim)

```
: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
```

Source: Community contributed

28.117 Secure File Encryption in Vim

Category: file_operations
Tags: encryption, security, file-management

Encrypt files using built-in Vim encryption methods, with options for different encryption strengths

Example: Example (Vim)

```
" Encrypt file with Blowfish2 (strongest method)
:X  " Prompt for encryption key
:setlocal cm=blowfish2  " Set strongest encryption method
```

Example: Example (Neovim)

```
-- Lua equivalent for encryption
vim.cmd('X')  -- Prompt for encryption key
vim.opt.cryptmethod = 'blowfish2'  -- Set strongest encryption method
```

Source: vim.fandom.com

28.118 Secure GPG File Editing in Vim/Neovim

Category: file_operations
Tags: encryption, gpg, security

Automatically handle GPG encrypted files with secure buffer and write operations

Example: Example (Vim)

```
augroup encrypted
  au!
  autocmd BufReadPre,FileReadPre *.gpg
    \ setlocal noswapfile bin
  autocmd BufReadPost,FileReadPost *.gpg
    \ execute "'[,']!gpg --decrypt --default-recipient-self" |
    \ setlocal nobin |
    \ execute "doautocmd BufReadPost " . expand("%:r")
  autocmd BufWritePre,FileWritePre *.gpg
    \ setlocal bin |
    \ '[,']!gpg --encrypt --default-recipient-self
  autocmd BufWritePost,FileWritePost *.gpg
    \ silent u |
    \ setlocal nobin
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('encrypted', { clear = true })

vim.api.nvim_create_autocmd({'BufReadPre', 'FileReadPre'}, {
  group = 'encrypted',
  pattern = '*.gpg',
  callback = function()
    vim.opt_local.swapfile = false
    vim.opt_local.binary = true
  end
})

vim.api.nvim_create_autocmd({'BufReadPost', 'FileReadPost'}, {
  group = 'encrypted',
  pattern = '*.gpg',
  callback = function()
    vim.cmd("'[,']!gpg --decrypt --default-recipient-self")
    vim.opt_local.binary = false
    vim.cmd("doautocmd BufReadPost " .. vim.fn.expand('%:r'))
  end
})

vim.api.nvim_create_autocmd({'BufWritePre', 'FileWritePre'}, {
  group = 'encrypted',
  pattern = '*.gpg',
  callback = function()
    vim.opt_local.binary = true
    vim.cmd("'[,']!gpg --encrypt --default-recipient-self")
```

```
  end
})

vim.api.nvim_create_autocmd({'BufWritePost', 'FileWritePost'}, {
  group = 'encrypted',
  pattern = '*.gpg',
  callback = function()
    vim.cmd('silent u')
    vim.opt_local.binary = false
  end
})
```

Source: vim.fandom.com

28.119 Set Working Directory to Current File

Category: file_operations
Tags: navigation, file-management, workflow

Automatically change Vim's working directory to the directory of the current file, making file operations more convenient

Example: Example (Vim)

```
set autochdir

" Alternative manual mapping
nnoremap <leader>cd :cd %:p:h<CR>:pwd<CR>
```

Example: Example (Neovim)

```
-- Set working directory automatically
vim.opt.autochdir = true

-- Or create a manual mapping
vim.keymap.set('n', '<leader>cd', function()
  vim.cmd('cd %:p:h')
  print(vim.fn.getcwd())
end, { desc = 'Change to current file directory' })
```

Source: vim.fandom.com

28.120 Sync Disk After Every Write to Prevent Data Loss

Category: file_operations
Tags: data-protection, file-safety, crash-prevention

Automatically run sync command after each buffer write to reduce chances of file cor-

ruption during unexpected system crashes or power failures

Example: Example (Vim)

```
" disk sync after every write
au BufWritePost * silent !sync
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePost', {
  callback = function()
    vim.fn.system('sync')
  end
})
```

Source: vim.fandom.com

28.121 Sync Vim Runtime Files with rsync

Category: file_operations
Tags: file-sync, runtime-files, system-management

Efficiently synchronize Vim runtime files from a remote FTP server using rsync, keeping your local runtime environment up to date

Example: Example (Vim)

```
# Rsync command to sync Vim runtime files
rsync -avzcP --delete --exclude='/dos/' ftp.nluug.nl::Vim/runtime/
↪  ./runtime/
```

Example: Example (Neovim)

```
-- While rsync is a shell command, you can trigger it in Lua using
↪  vim.fn.system()
vim.fn.system('rsync -avzcP --delete --exclude="/dos/"
↪  ftp.nluug.nl::Vim/runtime/ ./runtime/')
```

Source: vim.fandom.com

28.122 Track File Open Dialog to Current File's Directory

Category: file_operations
Tags: file-navigation, configuration

Automatically set the file open dialog to start in the current buffer's directory, making file navigation more convenient

Example: Example (Vim)

```
set browsedir=buffer
```

Example: Example (Neovim)

```
vim.opt.browsedir = 'buffer'
```

Source: vim.fandom.com

28.123 Transparently Edit Encrypted GPG Files in Vim

Category: file_operations
Tags: encryption, security, file-handling

Automatically decrypt and encrypt GPG files when opening and saving, simplifying secure file editing

Example: Example (Vim)

```
if has("autocmd")
  augroup GPGASCII
    au!
    au BufReadPost *.asc :%!gpg -q -d
    au BufReadPost *.asc |redraw
    au BufWritePre *.asc :%!gpg -q -e -a
    au BufWritePost *.asc u
    au VimLeave *.asc :!clear
  augroup END
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('GPGEncryption', { clear = true })

vim.api.nvim_create_autocmd({'BufReadPost'}, {
  group = 'GPGEncryption',
  pattern = '*.asc',
  callback = function()
    vim.cmd('!gpg -q -d')
    vim.cmd('redraw')
  end
})

vim.api.nvim_create_autocmd({'BufWritePre'}, {
  group = 'GPGEncryption',
  pattern = '*.asc',
  callback = function()
    vim.cmd('!gpg -q -e -a')
  end
})
```

Source: vim.fandom.com

28.124 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: Example (Vim)

```
:update            " save only if file is modified
:map <F2> :update<CR>  " map F2 to conditional save
```

Source: Community contributed

28.125 Windows File Associations with Vim/Neovim

Category: file_operations
Tags: windows, file-associations, configuration

Configure Windows file associations to open specific file types in Vim/Neovim, with support for remote editing and custom actions

Example: Example (Vim)

```
" Vim command to associate .php files with Vim
assoc .php=PHPFile
ftype PHPFile="C:\Program Files\Vim\vim82\gvim.exe" --remote-silent "%1"
```

Example: Example (Neovim)

```
-- Lua equivalent (conceptual, as this is a Windows registry operation)
-- Can use vim.fn.system() to run system commands if needed
-- Example of registering file type programmatically
vim.cmd('assoc .php=PHPFile')
vim.cmd('ftype PHPFile="C:\\Program Files\\Vim\\vim82\\gvim.exe"
↪  --remote-silent "%1"')
```

Source: vim.fandom.com

28.126 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: Example (Vim)

```
:write ++p
```

Source: Community contributed

28.127 Zip Open Buffers Quickly

Category: file_operations
Tags: buffer-management, file-archiving

Create a zip archive of all files currently open in Vim buffers

Example: Example (Vim)

```
:silent bufdo !zip proj.zip %
```

Example: Example (Neovim)

```
vim.api.nvim_command('silent bufdo !zip proj.zip ' .. vim.fn.expand('%'))
```

Source: vim.fandom.com

CHAPTER 29

Filetype specific tips

29.1 Binary and hex file editing

Category: File Type Specific
Tags: binary, hex, xxd, hexedit, file

Edit binary files using xxd hex editor integration.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.2 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: Example (Vim)

```
" 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>
```

Source: Community contributed

29.3 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: Example (Vim)

```
" 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\}/
```

Source: Community contributed

29.4 Configuration file syntax highlighting

Category: File Type Specific
Tags: config, conf, ini, properties, syntax

Enable proper syntax highlighting for various configuration formats.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.5 Detect Inform Header Files in Vim

Category: filetype
Tags: file-detection, custom-filetypes, syntax-detection

Automatically detect Inform header files and set correct filetype for syntax highlighting

Example: Example (Vim)

```
" In filetype.vim
au BufNewFile,BufRead *.h call FTCheck_inform()

fun! FTCheck_inform()
  if getline(1) =~ "^!"
    setfiletype inform
  elseif getline(2) =~ "^!"
    setfiletype inform
  else
    let s:colnum = col('.')
    let s:linenum = line('.')
    call cursor(500, 1)
    if search('\[\(\s*\I\i*\)*\s*;', 'bW') > 0
      setfiletype inform
    endif
    call cursor(s:linenum, s:colnum)
  endif
endfun
```

Example: Example (Neovim)

```
-- In init.lua or filetype detection
vim.api.nvim_create_autocmd({"BufNewFile", "BufRead"}, {
  pattern = "*.h",
  callback = function()
    local lines = vim.api.nvim_buf_get_lines(0, 0, 2, false)
    if lines[1]:match("^!") or lines[2]:match("^!") then
      vim.bo.filetype = "inform"
```

```
      return
    end

    local cursor_pos = vim.api.nvim_win_get_cursor(0)
    vim.api.nvim_win_set_cursor(0, {500, 0})

    if vim.fn.search('\[\(\s*\I\i*\)*\s*;', 'bW') > 0 then
      vim.bo.filetype = "inform"
    end

    vim.api.nvim_win_set_cursor(0, cursor_pos)
  end
})
```

Source: vim.fandom.com

29.6 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: Example (Vim)

```
" 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
```

Source: Community contributed

29.7 Git commit message formatting

Category: File Type Specific
Tags: git, commit, message, format, conventional

Improve git commit message writing with templates and formatting.

Example: Example (Vim)

```
" 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:
```

Source: Community contributed

29.8 Go language specific features

Category: File Type Specific
Tags: go, golang, gofmt, import, build

Configure Go development workflow with formatting and building.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

29.9 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: Example (Vim)

```
" 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
```

Source: Community contributed

29.10 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: Example (Vim)

```
" 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>
```

Source: Community contributed

29.11 Java class and package navigation

Category: File Type Specific
Tags: java, class, package, import, navigation

Streamline Java development with class and package utilities.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.12 JavaScript/TypeScript development setup

Category: File Type Specific
Tags: javascript, typescript, js, ts, node, format

Optimize settings for JavaScript and TypeScript development.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

29.13 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: Example (Vim)

```
" 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/
```

Source: Community contributed

29.14 Lua script configuration

Category: File Type Specific
Tags: lua, script, neovim, config, development

Configure Lua development for Neovim scripting and general development.

Example: Example (Vim)

```
" 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.
```

Source: Community contributed

29.15 Markdown writing and formatting

Category: File Type Specific
Tags: markdown, md, writing, format, preview

Enhance Markdown writing experience with formatting and navigation.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

29.16 Python indentation and formatting

Category: File Type Specific
Tags: python, indent, format, pep8, filetype

Configure Python-specific settings for proper indentation and formatting.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.17 Rust development optimization

Category: File Type Specific
Tags: rust, cargo, rustfmt, clippy, build

Set up efficient Rust development environment and shortcuts.

Example: Example (Vim)

```
" 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()
```

Source: Community contributed

29.18 SQL query formatting and execution

Category: File Type Specific
Tags: sql, query, format, database, execute

Enhance SQL development with formatting and execution capabilities.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

29.19 Shell script development

Category: File Type Specific
Tags: shell, bash, sh, script, executable

Streamline shell script development with proper settings and shortcuts.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

29.20 Template file creation

Category: File Type Specific
Tags: template, skeleton, file, creation, boilerplate

Automatically insert templates for new files based on file type.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.21 XML and configuration file handling

Category: File Type Specific
Tags: xml, config, plist, format, validate

Handle XML and various configuration file formats effectively.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

29.22 YAML configuration editing

Category: File Type Specific
Tags: yaml, yml, config, indent, validate

Optimize YAML editing with proper indentation and validation.

Example: Example (Vim)

```
" 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>-
```

Source: Community contributed

CHAPTER 30

Folding

30.1 Advanced Folding Techniques in Vim/Neovim

Category: folding
Tags: folding, code-navigation, productivity

Provides flexible folding methods to improve code readability and navigation by hiding complex sections of code

Example: Example (Vim)

```
" Indent folding with manual override
augroup vimrc
  au BufReadPre * setlocal foldmethod=indent
  au BufWinEnter * if &fdm == 'indent' | setlocal foldmethod=manual | endif
augroup END
```

Example: Example (Neovim)

```
-- Indent folding with manual override
vim.api.nvim_create_augroup('folding', { clear = true })
vim.api.nvim_create_autocmd({'BufReadPre'}, {
  group = 'folding',
  callback = function()
    vim.wo.foldmethod = 'indent'
  end
})

vim.api.nvim_create_autocmd({'BufWinEnter'}, {
  group = 'folding',
  callback = function()
    if vim.wo.foldmethod == 'indent' then
      vim.wo.foldmethod = 'manual'
    end
  end
})
```

Source: vim.fandom.com

30.2 Advanced Syntax Folding for Java Files

Category: folding
Tags: syntax-folding, code-organization, java

Custom syntax folding for Java files that intelligently folds methods, constructors, enums, and comments with support for annotations

Example: Example (Vim)

```
" Place in ~/.vim/after/syntax/java.vim

" Fold method definitions with annotations
syn region javaFuncDef start="^\z(\s*\)\%(@[A-Z]\k*\%((\_.\{-}\))\?\s*\)*\%(.\+\)\s\+\%([a-zA-Z$0-9_][$A-Za-z0-9_]*\)\s*(\_.*)\s*{" end="^\z1}\s*$" keepend transparent fold

" Fold multi-line comments
syn region javaMultiLineComment start="/[*]\{1,}" end="[*]/" keepend transparent fold
```

Example: Example (Neovim)

```
-- Use in init.lua or after/syntax/java.lua

local api = vim.api

-- Create custom syntax folding for Java
api.nvim_create_autocmd('FileType', {
  pattern = 'java',
  callback = function()
    vim.cmd[[
      syn region javaFuncDef start="^\z(\s*\)\%(@[A-Z]\k*\%((\_.\{-}\))\?\s*\)*\%(.\+\)\s\+\%([a-zA-Z$0-9_][$A-Za-z0-9_]*\)\s*(\_.*)\s*{" end="^\z1}\s*$" keepend transparent fold
      syn region javaMultiLineComment start="/[*]\{1,}" end="[*]/" keepend transparent fold
    ]]
    vim.opt.foldmethod = 'syntax'
  end
})
```

Source: vim.fandom.com

30.3 Automatically Fold Perl Subroutines

Category: folding
Tags: perl, syntax-folding, code-organization

Enable automatic folding for Perl subroutines using built-in syntax folding

Example: Example (Vim)

```
let perl_fold = 1
```

Example: Example (Neovim)

```
vim.g.perl_fold = 1
```

Source: vim.fandom.com

30.4 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: Example (Vim)

```
zf5j  " create fold 5 lines down
zf    " create fold from visual selection
```

Source: Community contributed

30.5 Custom Fold Text for Comments

Category: folding
Tags: fold-text, custom-display, comments

Customize fold display to show comment start and number of lines

Example: Example (Vim)

```
set foldtext=MyFoldText()
function MyFoldText()
  let line = getline(v:foldstart)
  let sub = substitute(line, '^[\t ]*', '', '')
  let nlines = v:foldend - v:foldstart + 1
  return "+-" . v:folddashes . nlines . ": " . sub
endfunction
```

Example: Example (Neovim)

```
function _G.custom_fold_text()
  local first_line = vim.api.nvim_buf_get_lines(0, vim.v.foldstart - 1,
  ↪  vim.v.foldstart, false)[1]
```

```
  local line_count = vim.v.foldend - vim.v.foldstart + 1
  return string.format("+-(%d lines): %s", line_count,
  ↪  first_line:gsub("^%s*", ""))
end

vim.o.foldtext = 'v:lua.custom_fold_text()'
```

Source: vim.fandom.com

30.6 Custom Function Folding in C/C++

Category: folding
Tags: folding, custom-functions, syntax

Create a custom fold expression that includes function prototypes and handles brace placement in C/C++ files

Example: Example (Vim)

```
function FoldBrace()
  if getline(v:lnum+1)[0] == '{'
    return '>1'
  endif
  if getline(v:lnum)[0] == '}'
    return '<1'
  endif
  return foldlevel(v:lnum-1)
endfunction
set foldexpr=FoldBrace()
set foldmethod=expr
```

Example: Example (Neovim)

```
function _G.custom_fold()
  local next_line = vim.fn.getline(vim.v.lnum + 1)
  local current_line = vim.fn.getline(vim.v.lnum)

  if next_line:sub(1,1) == '{' then
    return '>1'
  end

  if current_line:sub(1,1) == '}' then
    return '<1'
  end

  return vim.fn.foldlevel(vim.v.lnum - 1)
end

vim.o.foldexpr = 'v:lua.custom_fold()'
vim.o.foldmethod = 'expr'
```

Source: vim.fandom.com

30.7 Custom Java Folding with Expression Function

Category: folding
Tags: custom-folding, java, code-organization

Create intelligent fold levels for Java code based on comments and code structure

Example: Example (Vim)

```
function! MyFoldLevel(lineNumber)
  let thisLine = getline(a:lineNumber)
  if (thisLine =~ '\%(\%(/\*\*\).*\%(\*/\)\)\|\%({.*}\)')
    return '='
  elseif (thisLine =~ '\%(^\s*/\*\*\s*$\)\|{'
    return "a1"
  elseif (thisLine =~ '\%(^\s*\*/\s*$\)\|}'
    return "s1"
  endif
  return '='
endfunction
setlocal foldexpr=MyFoldLevel(v:lnum)
setlocal foldmethod=expr
```

Example: Example (Neovim)

```
function _G.java_fold_level(linenum)
  local line = vim.fn.getline(linenum)
  if line:match('(/\*\*.*\*/)|({.*})')
    return '='
  elseif line:match('^%s*/\*\*%s*$') or line:match('{')
    return 'a1'
  elseif line:match('^%s*\*/') or line:match('}')
    return 's1'
  end
  return '='
end

vim.wo.foldexpr = 'v:lua.java_fold_level(v:lnum)'
vim.wo.foldmethod = 'expr'
```

Source: vim.fandom.com

30.8 Custom Perl Folding with Fold Expression

Category: folding
Tags: custom-folding, perl, advanced-folding

Create a custom fold expression to intelligently fold Perl subroutines

Example: Example (Vim)

```
function GetPerlFold()
  if getline(v:lnum) =~ '^\s*sub\s'
    return ">1"
  elseif getline(v:lnum) =~ '\}\s*$'
    let my_perlnum = v:lnum
    let my_perlmax = line("$")
    while (1)
      let my_perlnum = my_perlnum + 1
      if my_perlnum > my_perlmax
        return "<1"
      endif
      let my_perldata = getline(my_perlnum)
      if my_perldata =~ '^\s*\(\#.*\)\?$'
        " do nothing
      elseif my_perldata =~ '^\s*sub\s'
        return "<1"
      else
        return "="
      endif
    endwhile
  else
    return "="
  endif
endfunction
setlocal foldexpr=GetPerlFold()
setlocal foldmethod=expr
```

Example: Example (Neovim)

```
local function get_perl_fold(lnum)
  local line = vim.fn.getline(lnum)
  if line:match('^%s*sub%s') then
    return '>1'
  elseif line:match('}%s*$') then
    local perlnum = lnum
    local perlmax = vim.fn.line('$')
    while true do
      perlnum = perlnum + 1
      if perlnum > perlmax then
        return '<1'
      end
      local perldata = vim.fn.getline(perlnum)
      if perldata:match('^%s*(#.*)?$') then
        -- do nothing
      elseif perldata:match('^%s*sub%s') then
        return '<1'
      else
        return '='
      end
    end
  else
    return '='
  end
```

```
end

vim.wo.foldexpr = 'v:lua.get_perl_fold(v:lnum)'
vim.wo.foldmethod = 'expr'
```

Source: vim.fandom.com

30.9 Customize Fold Text with Advanced Display

Category: folding
Tags: folding, display, code-readability

Create a custom fold text function that improves code readability by handling comments and braces intelligently, and right-aligning line count

Example: Example (Vim)

```
function! MyFoldText()
  let line = getline(v:foldstart)
  let n = v:foldend - v:foldstart + 1
  let info = " " . n . " lines"
  let sub = line
  let startbrace = substitute(line, '^.*{[ \t]*$', '{', 'g')
  if startbrace == '{'
    let line = getline(v:foldend)
    let endbrace = substitute(line, '^[ \t]*}\(.*\)$', '}', 'g')
    if endbrace == '}'
      let sub = sub.substitute(line, '^[ \t]*}\(.*\)$', '...}\1', 'g')
    endif
  endif
  return sub . info
endfunction

set foldtext=MyFoldText()
```

Example: Example (Neovim)

```
function _G.custom_fold_text()
  local start_line = vim.fn.getline(vim.v.foldstart)
  local line_count = vim.v.foldend - vim.v.foldstart + 1
  local info = " " .. line_count .. " lines"

  local sub = start_line
  local start_brace = start_line:match('^.*{[ \t]*$') and '{' or start_line

  if start_brace == '{' then
    local end_line = vim.fn.getline(vim.v.foldend)
    local end_brace = end_line:match('^[ \t]*}') and '}' or end_line

    if end_brace == '}' then
      sub = sub .. end_line:gsub('^[ \t]*}(.*)', '...}%1')
```

```
    end
  end

  return sub .. info
end

vim.o.foldtext = 'v:lua.custom_fold_text()'
```

Source: vim.fandom.com

30.10 Double-Click to Toggle Folds

Category: folding
Tags: key-mapping, folding, user-interface

Add a convenient double-click mouse mapping to open and close code folds in Vim/Neovim

Example: Example (Vim)

```
:noremap <2-LeftMouse> za
```

Example: Example (Neovim)

```
vim.keymap.set({'n', 'v'}, '<2-LeftMouse>', 'za', { desc = 'Toggle fold with
↪  double-click' })
```

Source: vim.fandom.com

30.11 Dynamic Folding for Java Code Blocks

Category: folding
Tags: syntax-folding, code-navigation

Implement custom folding for different code structures like methods, loops, try-catch blocks, and comments

Example: Example (Vim)

```
syn region javaMethod
↪  start="^\z(\s*\)\(public\|private\|protected\)\(\_.*)\*{\s*$"
↪  end="^\z1}\s*$" transparent fold keepend
syn region javaLoop start="^\z(\s*\)\(for\|if\|while\|switch\).*{\s*$"
↪  end="^\z1}\s*$" transparent fold keepend
syn region javadoc start="^\s*/\*\*" end="^.*\*/" transparent fold keepend
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'java',
  callback = function()
    vim.cmd([[
      syn region javaMethod
       ↪  start="^\z(\s*\)\(public\|private\|protected\)\(\_.*)\*{\s*$"
       ↪  end="^\z1}\s*$" transparent fold keepend
      syn region javaLoop start="^\z(\s*\)\(for\|if\|while\|switch\).*{\s*$"
       ↪  end="^\z1}\s*$" transparent fold keepend
      syn region javadoc start="^\s*/\*\*" end="^.*\*/" transparent fold
       ↪  keepend
    ]]
  end
})
```

Source: vim.fandom.com

30.12 Dynamic Indent Folding with Manual Override

Category: folding
Tags: autocmd, configuration, code-navigation

Configure Vim to use indent-based folding by default, but allow manual fold creation during editing

Example: Example (Vim)

```
augroup vimrc
  au BufReadPre * setlocal foldmethod=indent
  au BufWinEnter * if &fdm == 'indent' | setlocal foldmethod=manual | endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('folding', { clear = true })
vim.api.nvim_create_autocmd('BufReadPre', {
  group = 'folding',
  callback = function() vim.wo.foldmethod = 'indent' end
})
vim.api.nvim_create_autocmd('BufWinEnter', {
  group = 'folding',
  callback = function()
    if vim.wo.foldmethod == 'indent' then
      vim.wo.foldmethod = 'manual'
    end
  end
})
```

Source: vim.fandom.com

30.13 Dynamic Indent and Manual Folding

Category: folding
Tags: autocmd, folding, configuration

Automatically set initial folding to indent, then allow manual fold creation while editing

Example: Example (Vim)

```
augroup vimrc
  au BufReadPre * setlocal foldmethod=indent
  au BufWinEnter * if &fdm == 'indent' | setlocal foldmethod=manual | endif
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('foldconfig', { clear = true })
vim.api.nvim_create_autocmd({'BufReadPre'}, {
  group = 'foldconfig',
  callback = function()
    vim.opt_local.foldmethod = 'indent'
  end
})
vim.api.nvim_create_autocmd({'BufWinEnter'}, {
  group = 'foldconfig',
  callback = function()
    if vim.opt_local.foldmethod:get() == 'indent' then
      vim.opt_local.foldmethod = 'manual'
    end
  end
})
```

Source: vim.fandom.com

30.14 Flexible Folding Methods in Neovim

Category: folding
Tags: folding, code-navigation, productivity

Vim/Neovim supports multiple folding methods to help organize and navigate code, including syntax, indent, and manual folding techniques.

Example: Example (Vim)

```
" Set folding method
set foldmethod=syntax

" Limit fold nesting
set foldnestmax=3
```

Example: Example (Neovim)

```
-- Set folding method
vim.opt.foldmethod = 'syntax'

-- Limit fold nesting
vim.opt.foldnestmax = 3
```

Source: vim.fandom.com

30.15 Flexible Java Code Folding with Expression

Category: folding
Tags: custom-folding, expression-folding, java

Custom fold expression function that intelligently handles multi-line JavaDoc comments and code blocks

Example: Example (Vim)

```
function! MyFoldLevel(lineNumber)
  let thisLine = getline(a:lineNumber)
  if (thisLine =~ '\%(\%(/\*\*\).*\%(\/\)\)\|\%({.*}\)')
    return '='
  elseif (thisLine =~ '\%(^\s*/\*\*\s*$\)\|{')
    return "a1"
  elseif (thisLine =~ '\%(^\s*\*/\s*$\)\|}'))
    return "s1"
  endif
  return '='
endfunction
setlocal foldexpr=MyFoldLevel(v:lnum)
setlocal foldmethod=expr
```

Example: Example (Neovim)

```
function _G.java_fold_level(line_nr)
  local line = vim.fn.getline(line_nr)
  if line:match('(/\*\*.*%*/)')  -- One-line JavaDoc
    or line:match('{.*}')  -- Single line block
    then
    return '='
  elseif line:match('^%s*/\*\*%s*$') or line:match('{') then
    return 'a1'  -- Start a fold
  elseif line:match('^%s*%*/') or line:match('}') then
    return 's1'  -- End a fold
  end
  return '='
end

vim.cmd.setlocal('foldexpr=v:lua.java_fold_level(v:lnum)')
vim.cmd.setlocal('foldmethod=expr')
```

Source: vim.fandom.com

30.16 Fold Away Lines Not in Quickfix List

Category: folding
Tags: quickfix, error-handling, code-navigation

Automatically fold lines in the current buffer that are not present in the quickfix or location list, helping focus on relevant code sections with errors or matches

Example: Example (Vim)

```
function! s:FoldMisses(list, context)
  setlocal foldmethod=manual
  normal! zE
  let extra = a:context == 99999 ? g:foldmisses_context : a:context
  let last = 0
  for lnum in s:GetHitLineNumbers(a:list)
    let start = last==0 ? 1 : last+1+extra
    call s:Fold(start, lnum-1-extra)
    let last = lnum
  endfor
  call s:Fold(last+1+extra, line('$'))
endfunction

command! -bar -count=99999 FoldMisses call s:FoldMisses(getqflist(),
 ↪  <count>)
```

Example: Example (Neovim)

```
local function fold_misses(list, context)
  vim.wo.foldmethod = 'manual'
  vim.cmd('normal! zE')
  local extra = context == 99999 and vim.g.foldmisses_context or context
  local last = 0

  for _, item in ipairs(list) do
    if item.valid and item.bufnr == vim.fn.bufnr('') then
      local start = last == 0 and 1 or last + 1 + extra
      local end_line = item.lnum - 1 - extra

      if start < end_line then
        vim.cmd(string.format('%d,%dfold', start, end_line))
      end

      last = item.lnum
    end
  end

  if last > 0 then
    vim.cmd(string.format('%d,$fold', last + 1 + extra))
  end
```

```
end

vim.api.nvim_create_user_command('FoldMisses', function(opts)
  fold_misses(vim.fn.getqflist(), opts.count)
end, { bar = true, count = 99999 })
```

Source: vim.fandom.com

30.17 Fold C# Regions Easily

Category: folding
Tags: code-folding, language-specific, navigation

Automatically fold C# #region blocks using matchit and custom matchwords

Example: Example (Vim)

```
let b:match_words = '\s*#\s*region.*$:\s*#\s*endregion'
```

Example: Example (Neovim)

```
vim.g.match_words = '\s*#\s*region.*$:\s*#\s*endregion'
```

Source: vim.fandom.com

30.18 Fold C-Style Comments Automatically

Category: folding
Tags: syntax-folding, c-languages, comments

Automatically create folds for multi-line C-style comments in source code files

Example: Example (Vim)

```
au BufNewFile,BufRead *.cpp,*.c,*.h,*.java syn region myCComment start="/\*"
↪  end="\*/" fold keepend transparent
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd({"BufNewFile", "BufRead"}, {
  pattern = {"*.cpp", "*.c", "*.h", "*.java"},
  callback = function()
    vim.cmd([[syn region myCComment start="/\*" end="\*/" fold keepend
    ↪  transparent]])
  end
})
```

Source: vim.fandom.com

30.19 Fold C/C++ Functions Including Prototypes

Category: folding
Tags: code-folding, c-language, syntax

Custom folding function for C/C++ that includes function prototypes in code folding, allowing more comprehensive code collapse

Example: Example (Vim)

```
function FoldBrace()
  if getline(v:lnum+1)[0] == '{'
    return '>1'
  endif
  if getline(v:lnum)[0] == '}'
    return '<1'
  endif
  return foldlevel(v:lnum-1)
endfunction
set foldexpr=FoldBrace()
set foldmethod=expr
```

Example: Example (Neovim)

```
function _G.custom_fold_expr(lnum)
  local next_line = vim.fn.getline(lnum + 1)
  local current_line = vim.fn.getline(lnum)

  if next_line:sub(1,1) == '{' then
    return '>1'
  end

  if current_line:sub(1,1) == '}' then
    return '<1'
  end

  return vim.fn.foldlevel(lnum - 1)
end

vim.opt.foldexpr = 'v:lua.custom_fold_expr(v:lnum)'
vim.opt.foldmethod = 'expr'
```

Source: vim.fandom.com

30.20 Fold File Listings with Dynamic Hierarchy

Category: folding
Tags: file-browsing, hierarchy, output-visualization

Create dynamic folding for file listings from find, locate, or tar commands, allowing easy navigation of complex directory structures

Example: Example (Vim)

```
set mouse=a
set foldminlines=1 foldcolumn=2 fillchars="+" foldlevel=0
set foldmethod=expr
set foldexpr=FileBrowserFoldExpr()
set foldtext=FileBrowserFoldText()

function FileBrowserFoldExpr()
  let line=getline(v:lnum)
  let n=strlen(substitute(line,'[^/]*','','g'))
  if (line=~'^.*/$')
    return '>'.n
  elseif (strpart(getline(v:lnum+1),0,strlen(line)+1)==line.'/')
    return '>'.(n+1)
  endif
  return n
endfunction

function FileBrowserFoldText()
  return getline(v:foldstart) . '    ... [' . (v:foldend-v:foldstart+1) . '
  ↪  lines]'
endfunction
```

Example: Example (Neovim)

```
vim.opt.mouse = 'a'
vim.opt.foldminlines = 1
vim.opt.foldcolumn = '2'
vim.opt.fillchars = { fold = '+' }
vim.opt.foldlevel = 0
vim.opt.foldmethod = 'expr'

vim.g.FileBrowserFoldExpr = function()
  local line = vim.fn.getline(vim.v.lnum)
  local n = #line:gsub('[^/]*', '')
  if line:match('^.*/$') then
    return '>' .. n
  elseif vim.fn.getline(vim.v.lnum + 1):sub(1, #line + 1) == line .. '/'
  ↪  then
    return '>' .. (n + 1)
  end
  return n
end

vim.opt.foldexpr = 'v:lua.vim.g.FileBrowserFoldExpr()'

vim.g.FileBrowserFoldText = function()
  return vim.fn.getline(vim.v.foldstart) .. '    ... [' .. (vim.v.foldend -
  ↪  vim.v.foldstart + 1) .. ' lines]'
end
```

```
vim.opt.foldtext = 'v:lua.vim.g.FileBrowserFoldText()'
```

Source: vim.fandom.com

30.21 Fold INI File Sections Automatically

Category: folding
Tags: syntax, configuration, file-management

Enable automatic folding for INI file sections, making large configuration files more readable

Example: Example (Vim)

```
syn region dosiniSection start="^\[" end="\(\n\+\[\)\@="
↪  contains=dosiniLabel,dosiniHeader,dosiniComment keepend fold
setlocal foldmethod=syntax
setlocal foldlevel=20
```

Example: Example (Neovim)

```
vim.cmd([[syn region dosiniSection start="^\[" end="\(\n\+\[\)\@="
↪  contains=dosiniLabel,dosiniHeader,dosiniComment keepend fold]])
vim.wo.foldmethod = 'syntax'
vim.wo.foldlevel = 20
```

Source: vim.fandom.com

30.22 Fold Lines Matching Search Pattern

Category: folding
Tags: search, folding, regex

Create a fold that shows only lines matching a search pattern with optional context lines

Example: Example (Vim)

```
nnoremap \z :setlocal foldexpr=(getline(v:lnum)=~@/)?0:(getline(v:lnum-1)=~
↪  @/)||(getline(v:lnum+1)=~@/)?1:2 foldmethod=expr foldlevel=0
↪  foldcolumn=2<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>zf', function()
  vim.wo.foldexpr = 'v:lua.custom_fold_expr()'
  vim.wo.foldmethod = 'expr'
```

```
  vim.wo.foldlevel = 0
  vim.wo.foldcolumn = '2'
end, { desc = 'Fold lines matching search' })

function _G.custom_fold_expr()
  local line = vim.fn.getline(vim.v.lnum)
  local prev_line = vim.fn.getline(vim.v.lnum - 1)
  local next_line = vim.fn.getline(vim.v.lnum + 1)
  local search_pattern = vim.fn.getreg('/')

  if line:match(search_pattern) then
    return 0
  elseif prev_line:match(search_pattern) or next_line:match(search_pattern)
  ↪  then
    return 1
  else
    return 2
  end
end
```

Source: vim.fandom.com

30.23 Fold Multiple Empty Lines in a File

Category: folding
Tags: syntax, folding, whitespace

Automatically fold sequences of two or more empty lines to reduce visual clutter and improve readability

Example: Example (Vim)

```
syn match MyEmptyLines "(^\s*\n)\+" fold
syn sync fromstart
setlocal foldmethod=syntax
```

Example: Example (Neovim)

```
vim.cmd('syn match MyEmptyLines "(^\\s*\\n)\\+" fold')
vim.cmd('syn sync fromstart')
vim.opt.foldmethod = 'syntax'
```

Source: vim.fandom.com

30.24 Fold Text by Regex Search Pattern

Category: folding
Tags: search, folding, regex

Create a custom fold command that hides lines not matching a search pattern, with optional context lines around matches

Example: Example (Vim)

```
nnoremap \z :setlocal foldexpr=(getline(v:lnum)=~@/)?0:(getline(v:lnum-1)=~
↪ @/)\|\|(getline(v:lnum+1)=~@/)?1:2 foldmethod=expr foldlevel=0
↪ foldcolumn=2<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>z', function()
  vim.wo.foldexpr = 'v:lua.custom_fold_expr()'
  vim.wo.foldmethod = 'expr'
  vim.wo.foldlevel = 0
  vim.wo.foldcolumn = '2'
end, { desc = 'Fold by Search Pattern' })

function _G.custom_fold_expr()
  local line = vim.fn.getline(vim.v.lnum)
  local prev_line = vim.fn.getline(vim.v.lnum - 1)
  local next_line = vim.fn.getline(vim.v.lnum + 1)
  local search_pattern = vim.fn.getreg('/')

  if line:match(search_pattern) then
    return 0
  elseif prev_line:match(search_pattern) or next_line:match(search_pattern)
  ↪ then
    return 1
  else
    return 2
  end
end
```

Source: vim.fandom.com

30.25 Fold VB.NET Code with Region Markers

Category: folding
Tags: region-folding, language-specific, code-organization

Create custom folding for Visual Basic .NET files using region markers, allowing collapsible code sections similar to Visual Studio

Example: Example (Vim)

```
fu! VBFold(lnum)
    if getline(a:lnum) =~# '^\s*#Region'
        return '>1'
    elseif getline(a:lnum) =~# '^\s*#End Region'
        return  '<1'
```

```
    else
        return '='
    endif
endfu

fu! VBFoldText()
    return substitute(v:folddashes,'-',' ','g').
     ↪  matchstr(getline(v:foldstart), '^\s*#Region\s\+\zs.*')
endfu

setl foldenable foldmethod=expr foldexpr=VBFold(v:lnum)
 ↪  foldtext=VBFoldText()
```

Example: Example (Neovim)

```
local function vb_fold(lnum)
    local line = vim.fn.getline(lnum)
    if line:match('^%s*#Region') then
        return '>1'
    elseif line:match('^%s*#End Region') then
        return '<1'
    else
        return '='
    end
end

local function vb_fold_text()
    local fold_start_line = vim.fn.getline(vim.v.foldstart)
    local region_name = fold_start_line:match('^%s*#Region%s+(.+)')
    return vim.fn.substitute(vim.v.folddashes, '-', ' ', 'g') ..
     ↪  (region_name or '')
end

vim.opt_local.foldenable = true
vim.opt_local.foldmethod = 'expr'
vim.opt_local.foldexpr = 'v:lua.vb_fold(v:lnum)'
vim.opt_local.foldtext = 'v:lua.vb_fold_text()'
```

Source: vim.fandom.com

30.26 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: Example (Vim)

```
zm  " increase fold level
zr  " reduce fold level
```

Source: Community contributed

30.27 Indent-Based Folding for Plain Text

Category: folding
Tags: folding, text-organization, hierarchy

Create dynamic folds in plain text files based on indentation levels, allowing hierarchical text organization with minimal formatting

Example: Example (Vim)

```
setlocal foldmethod=expr
setlocal foldexpr=(getline(v:lnum)=~'^$')?-1:((indent(v:lnum)<indent(v:lnum
↪  +1))?('>'.indent(v:lnum+1)):indent(v:lnum))
set foldtext=getline(v:foldstart)
set fillchars=fold:\
highlight Folded ctermfg=DarkGreen ctermbg=Black
```

Example: Example (Neovim)

```
vim.opt_local.foldmethod = 'expr'
vim.opt_local.foldexpr = [[v:lua.custom_fold_expr()]]

function _G.custom_fold_expr()
  local line = vim.fn.getline(vim.v.lnum)
  local next_line = vim.fn.getline(vim.v.lnum + 1)

  if line:match('^%s*$') then return -1 end

  local current_indent = vim.fn.indent(vim.v.lnum)
  local next_indent = vim.fn.indent(vim.v.lnum + 1)

  return current_indent < next_indent and '>' .. next_indent or
  ↪  current_indent
end
```

Source: vim.fandom.com

30.28 MoinMoin Wiki Folding with Vim

Category: folding
Tags: file-type, syntax, editing

Create custom folding for MoinMoin wiki files based on section headings, with dynamic fold levels

Example: Example (Vim)

```
" Define folding based on wiki headings; start with all folds open.
setlocal foldlevel=20
setlocal foldmethod=expr
setlocal foldexpr=HeadingLevel(v:lnum)

function HeadingLevel(lnum)
  let n = strlen(substitute(getline(a:lnum), '[^=].*', '', ''))
  return (n == 0) ? '=' : '>' . n
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'moin',
  callback = function()
    vim.wo.foldlevel = 20
    vim.wo.foldmethod = 'expr'
    vim.wo.foldexpr = 'v:lua.MoinHeadingLevel(v:lnum)'
  end
})

_G.MoinHeadingLevel = function(lnum)
  local line = vim.fn.getline(lnum)
  local n = #line:match('^(=+)') or 0
  return n == 0 and '=' or '>' .. n
end
```

Source: vim.fandom.com

30.29 Navigate XML Project Files with Folding

Category: folding
Tags: xml, navigation, project-management

Quickly navigate complex XML project files by creating folds based on a specific attribute and moving between parent nodes

Example: Example (Vim)

```
function! FoldMoveToParentNode()
  normal! zc
  let curNodeIndent = indent(foldclosed('.'))
  let nodeIndent = curNodeIndent
  let curLine = line('.')
  let line = -1
  while nodeIndent >= curNodeIndent
```

```
    normal! zkzc
    let line = line('.')
    if line == curLine
      break
    endif
    let nodeIndent = indent(foldclosed('.'))
  endwhile
endfunction
```

Example: Example (Neovim)

```
function _G.fold_move_to_parent_node()
  vim.cmd('normal! zc')
  local cur_node_indent = vim.fn.indent(vim.fn.foldclosed('.'))
  local node_indent = cur_node_indent
  local cur_line = vim.fn.line('.')
  local line = -1

  while node_indent >= cur_node_indent do
    vim.cmd('normal! zkzc')
    line = vim.fn.line('.')
    if line == cur_line then
      break
    end
    node_indent = vim.fn.indent(vim.fn.foldclosed('.'))
  end
end

-- Optional mapping
vim.keymap.set('n', '<leader>fdp', _G.fold_move_to_parent_node, { desc =
↪  'Move to parent node in XML' })
```

Source: vim.fandom.com

30.30 Navigate XML Project Hierarchies with Folds

Category: folding
Tags: xml, navigation, project-structure

Quickly explore hierarchical XML files by creating folds based on display names and navigating parent nodes

Example: Example (Vim)

```
" Function to move to parent node in XML hierarchy
function! FoldMoveToParentNode()
  normal! zc
  let curNodeIndent = indent(foldclosed('.'))
  let nodeIndent = curNodeIndent
  let curLine = line('.')
  let line = -1
```

```
  while nodeIndent >= curNodeIndent
    normal! zkzc
    let line = line('.')
    if line == curLine
      break
    endif
    let nodeIndent = indent(foldclosed('.'))
  endwhile
endfunction

" Map to parent node navigation
nnoremap \fdp :call FoldMoveToParentNode()<CR>
```

Example: Example (Neovim)

```
function _G.fold_move_to_parent_node()
  vim.cmd('normal! zc')
  local cur_node_indent = vim.fn.indent(vim.fn.foldclosed('.'))
  local node_indent = cur_node_indent
  local cur_line = vim.fn.line('.')
  local line = -1

  while node_indent >= cur_node_indent do
    vim.cmd('normal! zkzc')
    line = vim.fn.line('.')
    if line == cur_line then
      break
    end
    node_indent = vim.fn.indent(vim.fn.foldclosed('.'))
  end
end

-- Map to parent node navigation
vim.keymap.set('n', '<leader>fp', _G.fold_move_to_parent_node, { desc =
↪  'Find Parent Node in XML' })
```

Source: vim.fandom.com

30.31 Open All Folds Automatically on File Open

Category: folding
Tags: folds, file-loading, configuration

Automatically open most folds when opening a file, with flexibility for different file types

Example: Example (Vim)

```
set foldmethod=indent
set foldlevelstart=20

" Open all folds for specific file types
```

```
autocmd Syntax c,cpp,vim,xml,html,xhtml setlocal foldmethod=syntax
autocmd Syntax c,cpp,vim,xml,html,xhtml,perl normal zR
```

Example: Example (Neovim)

```
-- Set fold method and default open level
vim.opt.foldmethod = 'indent'
vim.opt.foldlevelstart = 20

-- Automatically open folds for specific file types
vim.api.nvim_create_augroup('FoldConfig', { clear = true })
vim.api.nvim_create_autocmd('Syntax', {
  group = 'FoldConfig',
  pattern = {'c', 'cpp', 'vim', 'xml', 'html', 'xhtml', 'perl'},
  callback = function()
    vim.opt_local.foldmethod = 'syntax'
    vim.cmd('normal zR')
  end
})
```

Source: vim.fandom.com

30.32 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: Example (Vim)

```
zR  " open all folds
zM  " close all folds
```

Source: Community contributed

30.33 Prevent Syntax Folding Disruptions in Insert Mode

Category: folding
Tags: folding, insert-mode, performance

Temporarily switch to manual folding during insert mode to prevent syntax folding slowdowns and unexpected fold opening

Example: Example (Vim)

```
" Don't disrupt folds when inserting text
autocmd InsertEnter * if !exists('w:last_fdm') | let w:last_fdm=&foldmethod
↪  | setlocal foldmethod=manual | endif
autocmd InsertLeave,WinLeave * if exists('w:last_fdm') | let
↪  &l:foldmethod=w:last_fdm | unlet w:last_fdm | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('FoldMethodToggle', { clear = true })

vim.api.nvim_create_autocmd({'InsertEnter'}, {
  group = 'FoldMethodToggle',
  callback = function()
    if vim.w.last_fdm == nil then
      vim.w.last_fdm = vim.wo.foldmethod
      vim.wo.foldmethod = 'manual'
    end
  end
})

vim.api.nvim_create_autocmd({'InsertLeave', 'WinLeave'}, {
  group = 'FoldMethodToggle',
  callback = function()
    if vim.w.last_fdm ~= nil then
      vim.wo.foldmethod = vim.w.last_fdm
      vim.w.last_fdm = nil
    end
  end
})
```

Source: vim.fandom.com

30.34 Quick Fold Marker Wrapping

Category: folding
Tags: fold-markers, visual-mode, text-editing

Create fold markers around a visual selection quickly, useful for organizing code sections

Example: Example (Vim)

```
vmap <Leader>fold mz:<Esc>'<O// {{{<Esc>'>o// }}}<Esc>`z?{{{<CR>A<Space>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '<Leader>fold', function()
  -- Save current cursor position
  vim.cmd('normal! mz')
```

```
  -- Add fold markers at start and end of visual selection
  vim.cmd('normal! O// {{{\<Esc>j')
  vim.cmd('normal! o// }}}')
  vim.cmd('normal! `z')
  vim.cmd('normal! ?{{{\<CR>A ')
end, { desc = 'Wrap visual selection with fold markers' })
```

Source: vim.fandom.com

30.35 Quick Java API Folding

Category: folding
Tags: code-navigation, folding, source-code

Create a command to fold a Java source file, showing only public/protected/private declarations

Example: Example (Vim)

```
command! Japi Foldsearch public\s\|protected\s\|private\s
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Japi', function()
  vim.cmd('Foldsearch public\\s\\|protected\\s\\|private\\s')
end, {})
```

Source: vim.fandom.com

30.36 Quick Java API View via Folding

Category: folding
Tags: folding, code-navigation, source-analysis

Create a quick overview of methods and variables in a Java source file using regex folding

Example: Example (Vim)

```
command! Japi Foldsearch public\s\|protected\s\|private\s
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Japi', function()
  vim.cmd('Foldsearch public\\s\\|protected\\s\\|private\\s')
end, {})
```

Source: vim.fandom.com

30.37 Smart Folding for Diff Files

Category: folding
Tags: diff, file-parsing, custom-folding

Create a custom folding function for diff files that automatically folds diff headers, file changes, and hunks for better readability

Example: Example (Vim)

```
setlocal foldmethod=expr foldexpr=DiffFold(v:lnum)
function! DiffFold(lnum)
  let line = getline(a:lnum)
  if line =~ '^\(diff\|---\|+++\|@@\) '
    return 1
  elseif line[0] =~ '[-+ ]'
    return 2
  else
    return 0
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'diff',
  callback = function()
    vim.wo.foldmethod = 'expr'
    vim.wo.foldexpr = 'v:lua.DiffFold(v:lnum)'
  end
})

function _G.DiffFold(lnum)
  local line = vim.fn.getline(lnum)
  if line:match('^(diff|---|+++|@@) ') then
    return 1
  elseif line:match('^[-+ ]') then
    return 2
  else
    return 0
  end
end
```

Source: vim.fandom.com

30.38 Toggle fold

Category: Folding
Tags: fold, toggle, code

Use `za` to toggle fold under cursor open/closed.

Example: Example (Vim)

```
za  " toggle fold under cursor
```

Source: Community contributed

30.39 Use Built-in Folding for Outlining

Category: folding
Tags: navigation, text-editing, productivity

Leverage Vim's built-in folding functionality for creating document outlines, with multiple plugin options available

Example: Example (Vim)

```
set foldmethod=indent  # Example folding setup
zf  # Create fold
zo  # Open fold
zc  # Close fold
```

Example: Example (Neovim)

```
vim.opt.foldmethod = 'indent'
-- Fold commands remain the same in Neovim
```

Source: vim.fandom.com

30.40 Use Vim as a Powerful Outline Processor

Category: folding
Tags: outline, folding, indentation

Leverage Vim's built-in folding and indentation features to create and manage outlines efficiently

Example: Example (Vim)

```
set ai
set foldmethod=indent
```

```
" Promote/demote headlines
:map << to promote
:map >> to demote
```

Example: Example (Neovim)

```
vim.opt.autoindent = true
vim.opt.foldmethod = 'indent'

-- Promote/demote can be done with < and > in normal/visual mode
```

Source: vim.fandom.com

30.41 Use Vim-Style Folding in Documentation

Category: folding
Tags: documentation, folding, text-organization

Use marker-based folding to create collapsible sections in text files, making large documents more manageable

Example: Example (Vim)

```
" Modeline for folding
vim:fdm=marker:tw=78:isk=!-~,^*,^\|,^":ts=8:ft=help:norl:

" Example section with folding
==================================================
Commands {{{1
==================================================
```

Example: Example (Neovim)

```
-- Set folding method
vim.o.foldmethod = 'marker'

-- Create a buffer with foldable sections
vim.api.nvim_buf_set_lines(0, 0, -1, false, {
  '==================================================',
  'Commands {{{1',
  '=================================================='
})
```

Source: vim.fandom.com

30.42 VB.NET Style Folding in Vim/Neovim

Category: folding
Tags: folding, language-specific, code-organization

Create custom folding for Visual Basic .NET files based on #Region and #End Region markers

Example: Example (Vim)

```
fu! VBFold(lnum)
    if getline(a:lnum) =~# '^\s*#Region'
        return '>1'
    elseif getline(a:lnum) =~# '^\s*#End Region'
        return  '<1'
    else
        return '='
    endif
endfu

fu! VBFoldText()
    return substitute(v:folddashes,'-',' ','g').
    ↪  matchstr(getline(v:foldstart), '^\s*#Region\s\+\zs.*')
endfu

setl foldenable foldmethod=expr foldexpr=VBFold(v:lnum)
↪  foldtext=VBFoldText()
```

Example: Example (Neovim)

```
local function vb_fold(lnum)
    local line = vim.fn.getline(lnum)
    if line:match('^%s*#Region') then
        return '>1'
    elseif line:match('^%s*#End Region') then
        return '<1'
    else
        return '='
    end
end

local function vb_fold_text()
    local fold_start_line = vim.fn.getline(vim.v.foldstart)
    local region_name = fold_start_line:match('^%s*#Region%s+(.+)')
    return string.gsub(vim.v.folddashes, '-', ' ') .. (region_name or '')
end

vim.wo.foldenable = true
vim.wo.foldmethod = 'expr'
vim.wo.foldexpr = 'v:lua.vb_fold(v:lnum)'
vim.wo.foldtext = 'v:lua.vb_fold_text()'
```

Source: vim.fandom.com

30.43 XML Syntax Folding in Vim

Category: folding
Tags: xml, syntax-folding, file-type

Automatically enable syntax-based folding for XML files

Example: Example (Vim)

```
let g:xml_syntax_folding=1
au FileType xml setlocal foldmethod=syntax
```

Example: Example (Neovim)

```
vim.g.xml_syntax_folding = 1
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'xml',
  callback = function()
    vim.wo.foldmethod = 'syntax'
  end
})
```

Source: vim.fandom.com

30.44 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: Example (Vim)

```
5zF   " create fold for 5 lines
zf3j  " create fold from cursor down 3 lines
zfip  " create fold for inner paragraph
```

Source: Community contributed

CHAPTER 31
Formatting

31.1 Add Line Numbers Using Perl Filter

Category: formatting
Tags: line-numbering, external-filter, text-manipulation

Quickly add line numbers to the entire file using a Perl one-liner within Vim

Example: Example (Vim)

```
:amenu Mo1.Format.NumberLines<Tab>:!perl :1,$!perl -ne
↪  "printf("%3d:%s",$.,$$_);"
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>nl', function()
  vim.cmd(':%!perl -ne "printf("%3d:%s",$.,$$_)"')
end, { desc = 'Number lines' })
```

Source: vim.fandom.com

31.2 Align Numbers at Decimal Point

Category: formatting
Tags: text-alignment, number-formatting, substitution

Automatically align numbers with consistent decimal precision and formatting

Example: Example (Vim)

```
:%s§\([-+]\?\d\+\)\.\?(\d*$)§\1                          !\200000§
:%s§\%15c\s*!§!§
:%s§\(^\S*\)(\s*)!§\2\1.§
:%s§\%21c\d*§§
:%s§\s\(\d\)§+\1§
```

Example: Example (Neovim)

```
-- Lua implementation would typically use a more robust plugin like
↪  'Tabular'
-- Or a custom function with vim.api.nvim_buf_set_lines()
-- Example using Tabular plugin:
-- ':Tabularize /\./r0l0l0'
```

Source: vim.fandom.com

31.3 Align Text into Columns

Category: formatting
Tags: text-alignment, formatting, productivity

Easily align text into columns using an external Perl script, helpful for formatting code or config files

Example: Example (Vim)

```
:vmap <A-a> !perl ~/perl/align -c:=
```

Example: Example (Neovim)

```
-- Note: Requires external Perl script
vim.keymap.set('v', '<A-a>', function()
  vim.cmd('!perl ~/perl/align -c:=')
end, { desc = 'Align text into columns' })
```

Source: vim.fandom.com

31.4 Align Text into Columns Automatically

Category: formatting
Tags: text-alignment, formatting, perl-script

Quickly align text with equal spacing around a delimiter (like '=') using an external Perl script

Example: Example (Vim)

```
:vmap <A-a> !perl ~/perl/align -c:=
```

Example: Example (Neovim)

```
-- Note: Requires external Perl script
-- Can be implemented with more modern Lua alignment plugins
```

```
vim.keymap.set('v', '<A-a>', '!perl ~/perl/align -c:=', { desc = 'Align text
↪  columns' })
```

Source: vim.fandom.com

31.5 Automatic Table Field Alignment

Category: formatting
Tags: text-alignment, tables, editing

Automatically realign table rows to match the heading row's field widths, making table editing more consistent

Example: Example (Vim)

```
" Alignment mapping
map <Leader>ta :AlignTableFields<CR>
```

Example: Example (Neovim)

```
-- Lua conceptual implementation
function _G.align_table_fields()
  -- Implement table field alignment logic
end

vim.keymap.set('n', '<Leader>ta', _G.align_table_fields)
```

Source: vim.fandom.com

31.6 Automatic Text Wrapping for Plain Text

Category: formatting
Tags: text-formatting, writing, configuration

Set text width to automatically break lines at a specified character length, useful for plain text editing

Example: Example (Vim)

```
" Set text width to 60 characters
:setlocal textwidth=60
:setl tw=60
```

Example: Example (Neovim)

```
-- Set text width to 60 characters
vim.opt.textwidth = 60
```

Source: vim.fandom.com

31.7 Automatically Indent XML Files

Category: formatting
Tags: xml, indentation, autocmd

Automatically indent XML files using XSLT or a simple text replacement method

Example: Example (Vim)

```
if version >= 540
  augroup filetype
    autocmd FileType xml '[,']!xsltproc indent.xsl %
  augroup END
endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'xml',
  callback = function()
    -- Use xsltproc to indent XML file
    vim.cmd('!xsltproc indent.xsl %')
  end
})
```

Source: vim.fandom.com

31.8 Chop Long Lines Using fmt Command

Category: formatting
Tags: text-formatting, command-line, external-tools

Easily break long lines to a specified width using the external fmt command in Vim, with multiple application methods

Example: Example (Vim)

```
" Chop lines 1-10 to default width
:1,10!fmt

" In visual block mode
'v' to select block
```

```
'!fmt<Enter>'
```

Example: Example (Neovim)

```
-- For lines 1-10
vim.cmd('1,10!fmt')

-- In Neovim, can use vim.fn.feedkeys for visual block fmt
vim.keymap.set('v', '<leader>f', function()
  vim.fn.feedkeys('!fmt\<CR>', 'n')
end)
```

Source: vim.fandom.com

31.9 Clean Paragraphs for Copy-Paste to Word

Category: formatting
Tags: text-manipulation, copy-paste, formatting

Remove single newlines while preserving paragraph breaks when copying text to Word or other applications

Example: Example (Vim)

```
vmap <C-C> '+y:let @+ = substitute(@+, "\n\n\n*", "±", "g")\|:let @+ =
↪  substitute(@+, "\n", " ", "g")\|:let @+ = substitute(@+, "±", "\n",
↪  "g")<CR>\|'<
```

Example: Example (Neovim)

```
vim.keymap.set('v', '<C-C>', function()
  -- Copy visual selection
  vim.cmd('normal! "+y')
  local text = vim.fn.getreg('+')

  -- Remove single newlines, preserve paragraph breaks
  text = text:gsub('\n\n+', '±')
  text = text:gsub('\n', ' ')
  text = text:gsub('±', '\n')

  vim.fn.setreg('+', text)
end, { desc = 'Clean paragraphs for Word copy' })
```

Source: vim.fandom.com

31.10 Clean Up HTML Files with Tidy

Category: formatting
Tags: html, formatting, external-tool

Use the Tidy external tool to automatically format and clean up HTML and XML files directly in Vim/Neovim

Example: Example (Vim)

```
" Tidy mapping for HTML cleanup
:vmap ,x :!tidy -q -i --show-errors 0<CR>

" Command to tidy entire HTML file
:command Thtml :%!tidy -q -i --show-errors 0
```

Example: Example (Neovim)

```
-- Tidy mapping for HTML cleanup
vim.keymap.set('v', ',x', function()
  vim.cmd('!tidy -q -i --show-errors 0')
end, { desc = 'Tidy HTML formatting' })

-- Command to tidy entire HTML file
vim.api.nvim_create_user_command('Thtml', '%!tidy -q -i --show-errors 0',
↪  {})
```

Source: vim.fandom.com

31.11 Collapse Multi-Line Paragraphs

Category: formatting
Tags: text-manipulation, global-command

Quickly join lines within paragraphs, removing unnecessary line breaks

Example: Example (Vim)

```
:v/^\s*$/norm vipJ
```

Example: Example (Neovim)

```
vim.cmd(':%s/\v(\S+)\n(\S+)/\1 \2/ge')
```

Source: vim.fandom.com

31.12 Comment lines by filetype

Category: Formatting
Tags: comment, filetype, toggle

Automatically comment/uncomment lines based on current file type.

Example: Example (Vim)

```
" Vimscript:
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()
```

Example: Example (Neovim)

```
-- Lua:
local function comment_it()
  local ft = vim.bo.filetype
  if ft == 'vim' then
    vim.keymap.set('v', '+#', ':s/^/"/<CR>', { buffer = true })
    vim.keymap.set('v', '-#', ':s/^"//<CR>', { buffer = true })
  elseif ft == 'python' then
    vim.keymap.set('v', '+#', ':s/^/#/<CR>', { buffer = true })
    vim.keymap.set('v', '-#', ':s/^#//<CR>', { buffer = true })
  end
end

vim.api.nvim_create_autocmd('BufEnter', {
  pattern = '*',
  callback = comment_it,
  desc = 'Setup comment mappings by filetype'
})
```

Source: Community contributed

31.13 Convert CSV to Readable Columns

Category: formatting
Tags: csv, text-manipulation, data-parsing

Easily convert CSV data into more readable columnar format by setting fixed-width columns

Example: Example (Vim)

```
let width = 20
let fill = repeat(' ', width)
:%s/([^,]*),\=/\=strpart(submatch(1).fill, 0, width)/ge
:%s/\s\+$//ge
```

Example: Example (Neovim)

```
local function format_csv_columns(width)
  width = width or 20
  local fill = string.rep(' ', width)
  vim.cmd(':%s/([^,]*),\\=/\\=strpart(submatch(1)..\"' .. fill .. '", 0, ' ..
  ↪  width .. ')/ge')
  vim.cmd(':%s/\\s\\+$//ge')
end

-- Usage: :lua format_csv_columns(20)
```

Source: vim.fandom.com

31.14 Convert Code to Syntax-Highlighted HTML

Category: formatting
Tags: html, syntax-highlighting, code-export

Easily convert Vim/Neovim buffer to HTML with syntax coloring for sharing code in emails or forums

Example: Example (Vim)

```
:TOhtml

" Optional: Switch to light colorscheme first
:colorscheme default
```

Example: Example (Neovim)

```
-- Use built-in :TOhtml command
-- For light background before conversion
vim.cmd('colorscheme default')
vim.cmd('TOhtml')
```

Source: vim.fandom.com

31.15 Convert Spaces to Tabs Efficiently

Category: formatting
Tags: indentation, file-formatting, whitespace

Quickly convert spaces to tabs with consistent indentation across different tab sizes

Example: Example (Vim)

```
:ret! 2
:x
```

Example: Example (Neovim)

```
vim.cmd('ret! 2')
vim.cmd('x')
```

Source: vim.fandom.com

31.16 Environment-Aware Paragraph Formatting in LaTeX

Category: formatting
Tags: latex, text-formatting, mapping

Provides a custom mapping to format LaTeX paragraphs while respecting environments like equations and labels

Example: Example (Vim)

```
map \gq ?^$\|^\s*\(\\begin\|\\end\|\\label\)?1<CR>gq//-1<CR>
omap lp ?^$\|^\s*\(\\begin\|\\end\|\\label\)?1<CR>//-1<CR>.<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>gq', function()
  -- Use Vim's search and formatting commands
  vim.cmd('?^$\|^\s*\(\\begin\|\\end\|\\label\)\|\n?1')
  vim.cmd('normal! gq')
  vim.cmd('/')
end, { desc = 'Format LaTeX paragraph respecting environments' })
```

Source: vim.fandom.com

31.17 Expand Tabs to Spaces in Entire File

Category: formatting
Tags: indentation, whitespace, code-style

Convert all tab indents to spaces in the current file

Example: Example (Vim)

```
:set et|retab
```

Example: Example (Neovim)

```
vim.opt.expandtab = true
vim.cmd('retab!')
```

Source: vim.fandom.com

31.18 Flexible Line Formatting

Category: formatting
Tags: text-formatting, key-mapping

Quickly format a block of text to a specific width using a custom mapping

Example: Example (Vim)

```
map st :set tw=70<CR>v<S-}>gq<End>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'st', function()
  vim.opt.textwidth = 70
  -- Equivalent of visual select block and format
  vim.cmd('normal! v}gq')
end, { desc = 'Format text block to 70 characters' })
```

Source: vim.fandom.com

31.19 Format Code Blocks and Functions Easily

Category: formatting
Tags: text-objects, code-formatting, indentation

Use text objects to quickly format code blocks, functions, or entire files with built-in formatting operators

Example: Example (Vim)

```
=i{  " Format inner code block
gg=G  " Format entire buffer
```

Example: Example (Neovim)

```
vim.cmd('normal =i{')  -- Format inner code block
vim.cmd('normal gg=G')  -- Format entire buffer
```

Source: vim.fandom.com

31.20 Format Only Long Lines in Text

Category: formatting
Tags: text-formatting, text-manipulation, line-wrapping

Intelligently format only lines longer than a specified width without merging short lines

Example: Example (Vim)

```
:g/./ normal gqq

# Alternative for lines 80 chars or longer
:g/\{80,\}/ .!par w70
```

Example: Example (Neovim)

```
-- Format only lines with content
vim.cmd('g/./ normal gqq')

-- Format lines 80 chars or longer using external tool
vim.cmd('g/\{80,\}/ .!par w70')
```

Source: vim.fandom.com

31.21 Format Paragraph Without Moving Cursor

Category: formatting
Tags: text-formatting, cursor-position

Format the current paragraph while keeping the cursor in its original position

Example: Example (Vim)

```
gqap  " Format paragraph
gwap  " Format paragraph and restore cursor
```

Example: Example (Neovim)

```
vim.cmd('normal! gwap')  -- Format paragraph and restore cursor position
```

Source: vim.fandom.com

31.22 Format XML Files Using xmllint

Category: formatting
Tags: xml, external-tools, formatting

Automatically format XML files using xmllint, which can clean up and standardize XML document indentation

Example: Example (Vim)

```
au FileType xml exe ":silent %!xmllint --format --recover - 2>/dev/null"
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'xml',
  callback = function()
    vim.cmd('silent %!xmllint --format --recover - 2>/dev/null')
  end
})
```

Source: vim.fandom.com

31.23 Format with Treesitter

Category: Formatting
Tags: treesitter, format, syntax

Use `=ap` to format syntax-aware regions using Treesitter (when available).

Example: Example (Vim)

```
=ap  " format around paragraph with Treesitter
```

Source: Community contributed

31.24 Poor men's JSON formatter

Category: Formatting
Tags: text, format, json

A poor men's json formatter using `vim.json.decode` + `vim.json.encode`: You can put it in `ftplugin/json.lua`. Only works for the whole file, e.g. with `gggqG` Credits: yochem

Example: Example (Neovim)

```
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()'
```

Source: Community contributed

31.25 Powerful Text Formatting with Par

Category: formatting
Tags: text-processing, external-tool, formatting

Use the 'par' external tool for advanced text formatting and justification within Vim/Neovim

Example: Example (Vim)

```
set formatprg=par\ -w60
map <A-q> {v}!par -jw60<CR>
vmap <A-q> !par -jw60<CR>
```

Example: Example (Neovim)

```
vim.opt.formatprg = 'par -w60'
vim.keymap.set({'n', 'v'}, '<A-q>', function()
  vim.cmd('normal! {v}!par -jw60\<CR>')
end, { desc = 'Format text with Par' })
```

Source: vim.fandom.com

31.26 Pretty Format XML with External Tools

Category: formatting
Tags: xml, external-tools, text-processing

Easily format XML files using external tools like xmllint or Python, with support for both whole file and visual range formatting

Example: Example (Vim)

```
function! DoPrettyXML()
  let l:origft = &ft
  set ft=
  1s/<?xml .*?>//e
  0put ='<PrettyXML>'
  $put ='</PrettyXML>'
  silent %!xmllint --format -
  2d
  $d
  silent %<
  1
  exe "set ft=" . l:origft
endfunction
command! PrettyXML call DoPrettyXML()
```

Example: Example (Neovim)

```
function _G.pretty_xml()
  local orig_ft = vim.bo.filetype
  vim.bo.filetype = ''

  -- Remove XML header
  vim.cmd('1s/<?xml .*?>//e')

  -- Add fake tags
  vim.api.nvim_buf_set_lines(0, 0, 0, false, {'<PrettyXML>'})
  vim.api.nvim_buf_set_lines(0, -1, -1, false, {'</PrettyXML>'})

  -- Format with xmllint
  vim.fn.systemlist('xmllint --format -')

  -- Remove fake tags
  vim.cmd('2d')
  vim.cmd('$d')

  -- Restore filetype
  vim.bo.filetype = orig_ft
end

vim.api.nvim_create_user_command('PrettyXML', _G.pretty_xml, {})
```

Source: vim.fandom.com

31.27 Quick Line Formatting with gqq

Category: formatting
Tags: text-formatting, built-in-commands

Native Vim method to break long lines without external tools or regex

Example: Example (Vim)

```
gqq  " Format current line
```

Example: Example (Neovim)

```
-- Equivalent native Neovim formatting
vim.cmd('normal gqq')

-- Or using keymap
vim.keymap.set('n', '<leader>q', 'gqq', { desc = 'Format current line' })
```

Source: vim.fandom.com

31.28 Quick XML Formatting with Python

Category: formatting
Tags: xml, text-processing, python

Simple one-line command to format XML using Python's minidom, with a convenient mapping

Example: Example (Vim)

```
com! FormatXML :%!python3 -c "import xml.dom.minidom, sys;
↪  print(xml.dom.minidom.parse(sys.stdin).toprettyxml())"

nnoremap = :FormatXML<Cr>
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('FormatXML', function()
  vim.cmd("%!python3 -c 'import xml.dom.minidom, sys;
  ↪  print(xml.dom.minidom.parse(sys.stdin).toprettyxml())'")
end, {})

vim.keymap.set('n', '=', ':FormatXML<CR>', { noremap = true, silent = true
↪  })
```

Source: vim.fandom.com

31.29 Quick XML Indentation Without XSLT

Category: formatting
Tags: xml, text-replacement

Simple method to indent XML/HTML files using search and replace followed by auto-indentation

Example: Example (Vim)

```
:%s/></>
</g
gg=G
```

Example: Example (Neovim)

```
function IndentXML()
  -- Replace ><, add newline, then auto-indent entire file
  vim.cmd('%%s/></>
</g')
  vim.cmd('normal! gg=G')
end

-- Optional: create a command
vim.api.nvim_create_user_command('IndentXML', IndentXML, {})
```

Source: vim.fandom.com

31.30 Quick XML/HTML Indentation Trick

Category: formatting
Tags: xml, indentation, text-manipulation

Simple method to indent XML/HTML files without external tools

Example: Example (Vim)

```
:%s/></>
</g
gg=G
```

Example: Example (Neovim)

```
-- Split tags onto new lines and then indent
vim.api.nvim_command(':%s/></>
</g')
vim.api.nvim_command('gg=G')
```

Source: vim.fandom.com

31.31 Reformat Email Quotations Quickly

Category: formatting
Tags: email, text-formatting, text-objects

Easily reformat long email quotations to a consistent width using Vim's text formatting commands

Example: Example (Vim)

```
" Formatting options for email
set formatoptions+=cq
set textwidth=76

" Formatting methods:
" gqip - Format current paragraph
" gqap - Format around paragraph
" gq} - Format to end of paragraph
```

Example: Example (Neovim)

```
-- Set formatting options for email
vim.opt.formatoptions:append('cq')
vim.opt.textwidth = 76

-- Formatting can be done via text objects
-- Use gqip, gqap, or gq} in normal mode
```

Source: vim.fandom.com

31.32 Regex-Based Text Alignment Function

Category: formatting
Tags: text-alignment, regex, editing

Dynamically align text based on a regex pattern, such as aligning assignment statements or function declarations

Example: Example (Vim)

```
function! AlignSection(regex) range
  let extra = 1
  let sep = empty(a:regex) ? '=' : a:regex
  let maxpos = 0
  let section = getline(a:firstline, a:lastline)
  for line in section
    let pos = match(line, ' *'.sep)
    if maxpos < pos
      let maxpos = pos
    endif
```

```
  endfor
  call map(section, 'AlignLine(v:val, sep, maxpos, extra)')
  call setline(a:firstline, section)
endfunction
```

Example: Example (Neovim)

```
function M.align_section(regex)
  local extra = 1
  local sep = vim.fn.empty(regex) and '=' or regex
  local maxpos = 0
  local section = vim.api.nvim_buf_get_lines(0, vim.fn.line('.')-1,
  ↪  vim.fn.line('v')-1, false)

  for _, line in ipairs(section) do
    local pos = line:find(' *'..sep)
    maxpos = math.max(maxpos, pos or 0)
  end

  -- Implementation would require additional helper functions for full
  ↪  conversion
end
```

Source: vim.fandom.com

31.33 Reindent Entire File or Selected Text

Category: formatting
Tags: indentation, text-formatting, code-cleanup

Quickly fix indentation for entire file or selected text using built-in = operator

Example: Example (Vim)

```
" Reindent entire file
gg=G

" Reindent selected text in visual mode
V=

" Quick mapping to reindent and return to original position
map <F7> gg=G<C-o><C-o>
```

Example: Example (Neovim)

```
-- Reindent entire file
vim.cmd('normal! gg=G')

-- Reindent in visual mode
vim.cmd('normal! gv=')
```

```
-- Quick mapping to reindent file and return to original position
vim.keymap.set('n', '<F7>', function()
  vim.cmd('normal! gg=G')
  vim.cmd('normal! \<C-o>\<C-o>')
end, { desc = 'Reindent entire file' })
```

Source: vim.fandom.com

31.34 Unicode Text Formatting Workaround

Category: formatting
Tags: unicode, text-processing, encoding

Handle Unicode text formatting with iconv when using 'par'

Example: Example (Vim)

```
# Unicode conversion for par formatting
# cat file | iconv -f utf-8 -t <encoding> | par <options> | iconv -f
↪  <encoding> -t utf-8
```

Example: Example (Neovim)

```
-- Unicode formatting function (example)
function format_unicode_text(file, from_encoding, to_encoding, par_options)
  local cmd = string.format(
    'cat %s | iconv -f %s -t %s | par %s | iconv -f %s -t utf-8',
    file, from_encoding, to_encoding, par_options, to_encoding
  )
  return vim.fn.system(cmd)
end
```

Source: vim.fandom.com

31.35 Wrap Lines at 80 Chars Without Breaking Words

Category: formatting
Tags: text-formatting, line-wrap, editing

Format text to wrap at 80 characters without breaking words mid-line, useful for maintaining readable code or text

Example: Example (Vim)

```
set formatoptions+=w
set tw=80
gggqG
```

```
" Mapping to format current paragraph
nnoremap Q gqip
```

Example: Example (Neovim)

```
-- Set text width and format options
vim.opt.formatoptions:append('w')
vim.opt.textwidth = 80

-- Format entire file
vim.cmd('normal! gggqG')

-- Mapping to format current paragraph
vim.keymap.set('n', 'Q', 'gqip', { desc = 'Format current paragraph' })
```

Source: vim.fandom.com

CHAPTER 32

Fun

32.1 Flip a coin

Category: Fun
Tags: fun

Quick decision-making inside Neovim. Could help settle coding debates ("Tabs or spaces?").

Example: Example (Vim)

```
:echo ["Heads","Tails"][rand() % 2]
```

32.2 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: It turns out that this event is not implemented yet, help text is there just for fun. But... if you install plugin 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.

Example: Example (Vim)

```
:h UserGettingBored
```

32.3 Help!

Category: Fun
Tags: easter egg, fun

If you are really, really desperate:

Example: Example (Vim)

```
:help!
```

32.4 Holy Grail

Category: Fun
Tags: easter egg, fun

Find the Holy Grail by typing the following command:

Example: Example (Vim)

```
:help holy-grail
```

32.5 Matrix like effect

Category: Fun
Tags: fun

Poor man's matrix screen :)

Example: Example (Vim)

```
:!yes | awk '{print int(rand()*10)}' | pv -qL 100
```

32.6 Random quote generator:

Category: Fun
Tags: fun, file

Paste random quote into your buffer with the following command:

Example: Example (Vim)

```
: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)] })
```

32.7 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: Example (Vim)

```
:g/^/m0
```

32.8 Show all whitespace, but nicely

Category: Fun
Tags: fun

Use the following commands: Highlights trailing spaces in red. Instant "why did I leave that space there?" effect.

Example: Example (Vim)

```
:highlight ExtraWhitespace ctermbg=red guibg=red
:match ExtraWhitespace /\s\+$/
```

32.9 Shuffle lines

Category: Fun
Tags: fun, shuffle, edit

Use the following commands to randomly rearrange all lines: Perfect for testing or chaos.

Example: Example (Vim)

```
: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)
```

32.10 Smile!

Category: Fun
Tags: easter egg, fun

Check this easter egg:

Example: Example (Vim)

```
:smile
```

32.11 Sort randomly

Category: Fun
Tags: fun, sort

Sort lines randomly by using this simple command: Works for small files only.

Example: Example (Vim)

```
:sort!
```

32.12 Speaking French?

Category: Fun
Tags: easter egg, fun

Well, translate this:

Example: Example (Vim)

```
:h |
```

32.13 Surprise yourself!

Category: Fun
Tags: fun

The following command will print `Neovim is great` until you stop it with `Ctrl+C`:

Example: Example (Vim)

```
:!yes "Neovim is great"
```

32.14 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: Example (Vim)

```
:help 42
```

CHAPTER 33

G commands

33.1 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: Example (Vim)

```
" 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
```

33.2 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: Example (Vim)

```
" 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
```

33.3 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: Example (Vim)

```
gqap  " format around paragraph
gqq   " format current line
gq}   " format until end of paragraph

" In visual mode:
gq    " format selection
```

33.4 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: Example (Vim)

```
gwap  " format around paragraph, cursor stays
gww   " format current line, cursor stays
```

33.5 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: Example (Vim)

```
gt   " go to next tab
gT   " go to previous tab
{n}gt " go to tab number n
```

33.6 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: Example (Vim)

```
" Select multiple lines with same number:
1
1
1

" Press g CTRL-A to get:
1
2
3

" Similarly, g CTRL-X decrements sequentially
```

33.7 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: Example (Vim)

```
gI  " insert at column 1 (ignoring indentation)
I   " insert before first non-blank character (respects indentation)
```

33.8 Join lines without spaces with gJ

Category: Text
Tags: g-commands, join, lines

Use `gJ` to join lines without inserting spaces between them.

Example: Example (Vim)

```
gJ   " join current line with next, no space added
J    " join with space added (standard join)
```

33.9 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: Example (Vim)

```
gm  " jump to middle of screen line
```

33.10 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: Example (Vim)

```
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
```

33.11 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: Example (Vim)

```
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
```

33.12 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: Example (Vim)

```
" Place cursor on URL or file path:
gx  " open with default application (browser for URLs, etc.)
```

33.13 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: Example (Vim)

```
g;  " go to position of previous change (older)
g,  " go to position of next change (newer)
```

33.14 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: Example (Vim)

```
" After making a visual selection and returning to normal mode:
gv  " reselect the same area
```

33.15 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: Example (Vim)

```
gi  " jump to last insertion point and enter INSERT mode
```

33.16 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: Example (Vim)

```
" After searching with /pattern:
gn    " select next match
cgn   " change next match (then use . to repeat)
dgn   " delete next match
```

33.17 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: Example (Vim)

```
" Place cursor on a character:
ga  " shows decimal, hex, and octal values
```

33.18 Swap character case with g~

Category: Text
Tags: g-commands, case, swap, toggle

Use `g~` to swap the case of text (uppercase ↔ lowercase).

Example: Example (Vim)

```
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 34
Global

34.1 Global command - advanced patterns

Category: Global
Tags: global, regex, pattern, advanced

Combine global command with advanced patterns for complex operations.

Example: Example (Vim)

```
: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
```

34.2 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: Example (Vim)

```
:g/error/t$                 " copy all error lines to end of file
:g/TODO/t0                  " copy all TODO lines to top of file
```

34.3 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: Example (Vim)

```
: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)
```

34.4 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: Example (Vim)

```
: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'
```

34.5 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: Example (Vim)

```
: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
```

34.6 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: Example (Vim)

```
: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)
```

34.7 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: Example (Vim)

```
K
```

34.8 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: Example (Vim)

```
:ter
```

CHAPTER 35

Help

35.1 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: Example (Vim)

```
:version      " show Neovim version and build info
:intro        " show introduction message
:messages     " show message history
:checkhealth  " run health diagnostics
```

35.2 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: Example (Vim)

```
: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
```

35.3 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: Example (Vim)

```
:Man ls   "Opens man-page for linux ls command
gO        "Generates TOC in normal mode for the selected man-page
```

35.4 Master help index

Category: Help
Tags: help, index, reference

Use `:h index.txt` to access the master help index with all available commands.

Example: Example (Vim)

```
:h index.txt  " master help index
```

35.5 Search help by pattern

Category: Help
Tags: help, search, pattern

Use `:help pattern` to search help documentation for specific keywords or patterns.

Example: Example (Vim)

```
:help pattern  " search help for 'pattern'
```

CHAPTER 36

Indentation

36.1 Adjust Indent in Insert Mode

Category: indentation
Tags: insert-mode, indentation, editing

Quickly increase or decrease indent while in insert mode

Example: Example (Vim)

```
" Ctrl-D: decrease indent
" Ctrl-T: increase indent
```

Example: Example (Neovim)

```
-- These are built-in Neovim behaviors, but can be explicitly mapped
vim.keymap.set('i', '<C-d>', '<C-d>', { desc = 'Decrease indent' })
vim.keymap.set('i', '<C-t>', '<C-t>', { desc = 'Increase indent' })
```

Source: vim.fandom.com

36.2 Auto indent

Category: Indentation
Tags: indent, auto, format

Use `==` to auto-indent current line, or `{number}==` to auto-indent multiple lines.

Example: Example (Vim)

```
==   " auto-indent current line
5==  " auto-indent 5 lines
```

Source: Community contributed

36.3 Block Indent with Matching Brace

Category: indentation
Tags: code-formatting, text-objects

Indent entire code blocks relative to matching braces quickly

Example: Example (Vim)

```
" Indent block with matching brace
=%

" Increase/decrease block indent
>%
<%
```

Example: Example (Neovim)

```
-- These commands work identically in Neovim
-- Cursor should be on { or } when using
-- =% indents block with matching brace
-- >% increases block indent
-- <% decreases block indent
```

Source: vim.fandom.com

36.4 Block Indentation with Brace Matching

Category: indentation
Tags: code-formatting, matching-pairs, text-objects

Use % to match braces and indent entire code blocks with single commands

Example: Example (Vim)

```
" Indent block including matching brace
=%

" Increase/decrease block indent
>%
<%
```

Example: Example (Neovim)

```
-- Lua equivalents work identically in Neovim
-- These commands leverage Vim's text object and matching capabilities
```

Source: vim.fandom.com

36.5 Correct Indentation in Blank Lines

Category: indentation
Tags: editing, insert-mode, indentation

Use 'cc' or 'S' to start insert mode on a new line with correct indentation, avoiding manual indent issues

Example: Example (Vim)

```
" No direct vimscript mapping needed, just use built-in commands
```

Example: Example (Neovim)

```
-- Use built-in Vim commands 'cc' or 'S' to start insert mode
-- with correct indentation on a blank line
```

Source: vim.fandom.com

36.6 Disable Auto-Indentation for Current File

Category: indentation
Tags: configuration, editing

Quickly turn off auto-indentation for the current buffer

Example: Example (Vim)

```
:setlocal noautoindent nocindent nosmartindent indentexpr=
```

Example: Example (Neovim)

```
-- Disable auto-indentation for current buffer
vim.opt_local.autoindent = false
vim.opt_local.cindent = false
vim.opt_local.smartindent = false
vim.opt_local.indentexpr = ''
```

Source: vim.fandom.com

36.7 Disable Auto-Indentation for Specific File Types

Category: indentation
Tags: file-type, configuration, editing

Create a custom indent script to disable auto-indentation for specific file types like HTML

Example: Example (Vim)

```
let b:did_indent = 1
```

Example: Example (Neovim)

```
-- In ~/.config/nvim/after/indent/html.lua
vim.g.did_indent = 1
```

Source: vim.fandom.com

36.8 Easy Indentation with Cursor Position Preserved

Category: indentation
Tags: key-mapping, indent, cursor-preservation

Indent lines in normal and insert mode while keeping the cursor in the same relative position

Example: Example (Vim)

```
"Indent forward in normal mode
nmap <C-Tab> a<C-t><Esc>

"Indent backward in normal mode
nmap <C-S-Tab> a<C-d><Esc>

"Indent forward in insert mode
imap <C-Tab> <C-t>

"Indent backward in insert mode
imap <C-S-Tab> <C-d>
```

Example: Example (Neovim)

```
-- Indent forward in normal mode
vim.keymap.set('n', '<C-Tab>', 'a<C-t><Esc>', { desc = 'Indent line forward'
↪  })

-- Indent backward in normal mode
vim.keymap.set('n', '<C-S-Tab>', 'a<C-d><Esc>', { desc = 'Indent line
↪  backward' })

-- Indent forward in insert mode
vim.keymap.set('i', '<C-Tab>', '<C-t>', { desc = 'Indent line forward' })

-- Indent backward in insert mode
vim.keymap.set('i', '<C-S-Tab>', '<C-d>', { desc = 'Indent line backward' })
```

Source: vim.fandom.com

36.9 Flexible Indentation and Tab Settings

Category: indentation
Tags: tabs, formatting, indentation

Configure indentation to use 4 spaces instead of tabs, with flexible shift width settings

Example: Example (Vim)

```
" Indentation using 4 spaces
set shiftwidth=4
set softtabstop=4
set expandtab
```

Example: Example (Neovim)

```
-- Configure indentation
vim.opt.shiftwidth = 4
vim.opt.softtabstop = 4
vim.opt.expandtab = true
```

Source: vim.fandom.com

36.10 Improve PHP and HTML Indentation Support

Category: indentation
Tags: php, html, filetype, indent

Custom indent script to handle mixed PHP and HTML files, providing better indentation support for files with embedded HTML in PHP

Example: Example (Vim)

```
" Better indent support for PHP with HTML
function! GetPhpHtmlIndent(lnum)
  if exists('*HtmlIndent')
    let html_ind = HtmlIndent()
  else
    let html_ind = HtmlIndentGet(a:lnum)
  endif
  let php_ind = GetPhpIndent()

  if php_ind > -1
    return php_ind
  endif

  if html_ind > -1
    if getline(a:lnum) =~ "^<?" && (0< searchpair('<?', '', '?>', 'nWb')
          \ || 0 < searchpair('<?', '', '?>', 'nW'))
      return -1
```

```
    endif
    return html_ind
  endif

  return -1
endfunction

setlocal indentexpr=GetPhpHtmlIndent(v:lnum)
setlocal indentkeys+=<>>
```

Example: Example (Neovim)

```
local function get_php_html_indent(lnum)
  local html_ind = vim.fn['HtmlIndentGet'](lnum)
  local php_ind = vim.fn['GetPhpIndent']()

  if php_ind > -1 then
    return php_ind
  end

  if html_ind > -1 then
    local line = vim.fn.getline(lnum)
    if line:match('^<?') and (
      vim.fn.searchpair('<?', '', '?>', 'nWb') > 0 or
      vim.fn.searchpair('<?', '', '?>', 'nW') > 0
    ) then
      return -1
    end
    return html_ind
  end

  return -1
end

-- Set up for PHP files
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'php',
  callback = function()
    vim.bo.indentexpr = 'v:lua.get_php_html_indent(v:lnum)'
    vim.bo.indentkeys = vim.bo.indentkeys .. ',<>'
  end
})
```

Source: vim.fandom.com

36.11 Increment/Decrement Block Indentation

Category: indentation
Tags: code-formatting, editing, movement

Easily increase or decrease indentation for code blocks using motion commands

Example: Example (Vim)

```
" Increase indent of inner block
>i{

" Decrease indent of inner block
<i{
```

Example: Example (Neovim)

```
-- Increase indent of inner block
vim.cmd('normal >i{')

-- Decrease indent of inner block
vim.cmd('normal <i{')
```

Source: vim.fandom.com

36.12 Indent with Tabs, Align with Spaces

Category: indentation
Tags: whitespace, formatting, code-style

Configure Vim to use tabs for indentation and spaces for alignment, allowing flexible tab width while maintaining text alignment

Example: Example (Vim)

```
set noexpandtab
set copyindent
set preserveindent
set softtabstop=0
set shiftwidth=4
set tabstop=4
```

Example: Example (Neovim)

```
vim.opt.expandtab = false
vim.opt.copyindent = true
vim.opt.preserveindent = true
vim.opt.softtabstop = 0
vim.opt.shiftwidth = 4
vim.opt.tabstop = 4
```

Source: vim.fandom.com

36.13 Preserve Indent Across Blank Lines

Category: indentation
Tags: indent, autoindent, blank-lines

Automatically maintain indentation when creating new lines, even when traversing blank lines. Useful for maintaining consistent code structure and readability.

Example: Example (Vim)

```
function! IndentIgnoringBlanks(child)
  let lnum = v:lnum
  while v:lnum > 1 && getline(v:lnum-1) == ""
    normal k
    let v:lnum = v:lnum - 1
  endwhile
  if a:child == ""
    if ! &l:autoindent
      return 0
    elseif &l:cindent
      return cindent(v:lnum)
    endif
  else
    exec "let indent=".a:child
    if indent != -1
      return indent
    endif
  endif
  if v:lnum == lnum && lnum != 1
    return -1
  endif
  let next = nextnonblank(lnum)
  if next == lnum
    return -1
  endif
  if next != 0 && next-lnum <= lnum-v:lnum
    return indent(next)
  else
    return indent(v:lnum-1)
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.indent_ignoring_blanks(child)
  local lnum = vim.v.lnum
  while lnum > 1 and vim.fn.getline(lnum - 1) == "" do
    vim.cmd('normal! k')
    lnum = lnum - 1
  end

  if child == "" then
    if not vim.bo.autoindent then
```

```
      return 0
    elseif vim.bo.cindent then
      return vim.fn.cindent(lnum)
    end
  else
    local indent = vim.fn.eval(child)
    if indent ~= -1 then
      return indent
    end
  end

  if lnum == vim.v.lnum and vim.v.lnum ~= 1 then
    return -1
  end

  local next = vim.fn.nextnonblank(lnum)
  if next == lnum then
    return -1
  end

  if next ~= 0 and next - lnum <= lnum - vim.v.lnum then
    return vim.fn.indent(next)
  else
    return vim.fn.indent(vim.v.lnum - 1)
  end
end

-- Usage
vim.cmd('command! -bar IndentIgnoringBlanks lua
 ↪  _G.setup_indent_ignoring_blanks()')
```

Source: vim.fandom.com

36.14 Preserve Indentation When Typing Hash

Category: indentation
Tags: indent, configuration, insert-mode

Prevent Vim from removing indentation when typing # at the start of a line, especially useful for comments in various programming languages

Example: Example (Vim)

```
set cindent
set cinkeys-=0#
set indentkeys-=0#

" Alternative mapping for smartindent
:inoremap # X<BS>#
```

Example: Example (Neovim)

```
-- Preserve indentation for lines starting with #
vim.opt.cindent = true
vim.cmd('set cinkeys-=0#')
vim.cmd('set indentkeys-=0#')

-- Alternative mapping
vim.keymap.set('i', '#', function()
  local line = vim.api.nvim_get_current_line()
  local cursor = vim.api.nvim_win_get_cursor(0)
  vim.api.nvim_buf_set_text(0, cursor[1]-1, cursor[2], cursor[1]-1,
  ↪  cursor[2], {'#'})
end)
```

Source: vim.fandom.com

36.15 Quick Block Indentation Commands

Category: indentation
Tags: code-formatting, text-objects, block-editing

Efficient ways to reindent code blocks in Vim, using text object navigation

Example: Example (Vim)

```
" Reindent inner block
=i{

" Reindent block including braces
=a{

" Increase/decrease block indent
>i{
<i{
```

Example: Example (Neovim)

```
-- Lua equivalents (these work the same in Neovim)
-- Can be used in visual mode or with text objects
-- Use = for auto-reindent
-- Use > or < for manual indent changes
```

Source: vim.fandom.com

36.16 Quickly Reindent Code Blocks

Category: indentation
Tags: code-formatting, editing, movement

Powerful commands to reindent code blocks using text objects and motion commands

Example: Example (Vim)

```
" Reindent inner block
=i{

" Reindent block including braces
=a{

" Reindent with cursor on brace
=%
```

Example: Example (Neovim)

```
-- Reindent inner block
vim.cmd('normal =i{')

-- Reindent block including braces
vim.cmd('normal =a{')

-- Reindent with cursor on brace
vim.cmd('normal =%%')
```

Source: vim.fandom.com

36.17 Quickly Reindent Entire File or Selected Text

Category: indentation
Tags: formatting, code-style, text-editing

Easily reformat indentation for entire file or selected sections using built-in = operator

Example: Example (Vim)

```
" Reindent entire file
gg=G

" Reindent selected visual block
=

" Mapping to reindent file and return to original position
map <F7> gg=G<C-o><C-o>
```

Example: Example (Neovim)

```
-- Reindent entire file
vim.cmd('normal! gg=G')

-- Reindent selected visual block
```

```
vim.cmd('normal! gv=')

-- Mapping to reindent file and return to original position
vim.keymap.set('n', '<F7>', function()
  vim.cmd('normal! gg=G')
  vim.cmd('normal! 2\<C-o>')
end, { desc = 'Reindent entire file' })
```

Source: vim.fandom.com

36.18 Reindent Code Blocks Efficiently

Category: indentation
Tags: code-formatting, text-editing, navigation

Quickly reindent code blocks using text objects and indent commands in Vim/Neovim

Example: Example (Vim)

```
" Reindent inner block
=i{

" Reindent block including braces
=a{

" Increase/decrease indent of block
>i{
<i{
```

Example: Example (Neovim)

```
-- Lua equivalents use same keystrokes
-- These work out of the box in Neovim
-- =i{ reindents inner block
-- =a{ reindents block with braces
-- >i{ increases indent
-- <i{ decreases indent
```

Source: vim.fandom.com

36.19 Select and Manipulate Code Blocks by Indentation

Category: indentation
Tags: text-objects, selection, editing

Create text objects that select code blocks based on indentation levels, allowing easy manipulation of entire indented sections

Example: Example (Vim)

```
"Place in vimrc
onoremap <silent>ai :<C-U>cal <SID>IndTxtObj(0)<CR>
onoremap <silent>ii :<C-U>cal <SID>IndTxtObj(1)<CR>
vnoremap <silent>ai :<C-U>cal <SID>IndTxtObj(0)<CR><Esc>gv
vnoremap <silent>ii :<C-U>cal <SID>IndTxtObj(1)<CR><Esc>gv

function! s:IndTxtObj(inner)
  let curline = line(".")
  let lastline = line("$")
  let i = indent(line(".")) - &shiftwidth * (v:count1 - 1)
  let i = i < 0 ? 0 : i
  if getline(".") !~ "^\\s*$"
    "... rest of the function implementation ...
  endif
endfunction
```

Example: Example (Neovim)

```
local function ind_txt_obj(inner)
  local curline = vim.fn.line('.')
  local lastline = vim.fn.line('$')
  local indent_level = vim.fn.indent(curline) - vim.o.shiftwidth *
  ↪  (vim.v.count1 - 1)
  indent_level = indent_level < 0 and 0 or indent_level

  if vim.fn.getline('.') ~= '' then
    -- Implement selection logic similar to Vimscript function
    -- This would require translating the complex line selection logic
  end
end

-- Set up keymappings
vim.keymap.set({'o', 'v'}, 'ai', function() ind_txt_obj(false) end, { silent
↪  = true })
vim.keymap.set({'o', 'v'}, 'ii', function() ind_txt_obj(true) end, { silent
↪  = true })
```

Source: vim.fandom.com

36.20 Smart Dedent to Previous Indent Level

Category: indentation
Tags: indent-management, python-integration, insert-mode

Provides a smart way to dedent to the previous meaningful indent level, instead of just moving by fixed shiftwidth increments

Example: Example (Vim)

```
function! DedentToPrevious()
python << EOF
import vim
tabsize = int(vim.eval("&ts"))
l = vim.current.line
rest = l.lstrip()
indent = l[:-len(rest)]
if indent != "":
    cur_size = len(indent.replace("\t", " "*tabsize))
    idx = vim.current.window.cursor[0]-2
    while idx >= 0:
        ll = vim.current.buffer[idx]
        indent = ll[:-len(ll.lstrip())]
        if len(indent.replace("\t", " "*tabsize)) < cur_size:
            vim.current.line = indent+rest
            break
        idx -= 1
EOF
endfunction

" Replace Ctrl-D in insert mode with smart dedent
imap <C-d> <C-o>:call DedentToPrevious()<CR>
```

Example: Example (Neovim)

```
function _G.smart_dedent()
    local tabsize = vim.o.tabstop
    local current_line = vim.api.nvim_get_current_line()
    local rest = current_line:match("^%s*(.*)$")
    local current_indent = current_line:match("^(%s*)")

    if current_indent ~= "" then
        local cur_size = #(current_indent:gsub("\t", string.rep(" ",
        ↪  tabsize)))
        local current_line_nr = vim.api.nvim_win_get_cursor(0)[1]

        for idx = current_line_nr - 2, 0, -1 do
            local prev_line = vim.api.nvim_buf_get_lines(0, idx, idx+1,
            ↪  false)[1]
            local prev_indent = prev_line:match("^(%s*)")
            local prev_indent_size = #(prev_indent:gsub("\t", string.rep("
            ↪  ", tabsize)))

            if prev_indent_size < cur_size then
                vim.api.nvim_set_current_line(prev_indent .. rest)
                break
            end
        end
    end
end

-- Map Ctrl-D to smart dedent in insert mode
```

```
vim.keymap.set('i', '<C-d>', '<Cmd>lua _G.smart_dedent()<CR>', { noremap =
↪  true, silent = true })
```

Source: vim.fandom.com

36.21 Smart Tabs for Consistent Indentation

Category: indentation
Tags: tabs, spaces, code-formatting

Use tabs only for indentation levels while using spaces for text alignment, maintaining consistent code appearance across different tab widths

Example: Example (Vim)

```
" Recommended to use Smart Tabs plugin
:set noet sts=0 sw=4 ts=4
:set cindent
:set cinoptions=(0,u0,U0
```

Example: Example (Neovim)

```
-- Recommended to use Smart Tabs plugin
vim.opt.expandtab = false
vim.opt.softtabstop = 0
vim.opt.shiftwidth = 4
vim.opt.tabstop = 4
vim.opt.cindent = true
vim.opt.cinoptions = '(0,u0,U0'
```

Source: vim.fandom.com

36.22 Use Filetype-Based Indentation

Category: indentation
Tags: filetype, configuration, best-practices

Prefer filetype-based indentation over generic smartindent, which provides more accurate and language-specific indentation rules

Example: Example (Vim)

```
filetype indent on
set autoindent
```

Example: Example (Neovim)

```
-- Enable filetype-specific indentation
vim.cmd('filetype indent on')
vim.opt.autoindent = true
```

Source: vim.fandom.com

CHAPTER 37

Insert

37.1 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: Example (Vim)

```
" In insert mode:
Ctrl+t  " increase indent
Ctrl+d  " decrease indent
```

37.2 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: Example (Vim)

```
" In insert mode:
Ctrl+g u  " start new undo block
Ctrl+g U  " don't break undo with next movement
```

37.3 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: Example (Vim)

```
" In insert mode:
Ctrl+y  " copy character from line above
Ctrl+e  " copy character from line below
```

37.4 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: Example (Vim)

```
" In insert mode:
Ctrl+c  " quit insert mode (no abbreviation check)
Ctrl+[  " same as Escape key
```

37.5 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: Example (Vim)

```
gO  " insert line above without moving cursor
```

37.6 Insert calculation result

Category: Insert
Tags: calculation, expression, register

Use `Ctrl+r =` to insert the result of an expression calculation in insert mode.

Example: Example (Vim)

```
" In insert mode:
Ctrl+r =2+3<Enter>  " inserts 5
Ctrl+r =strftime("%Y-%m-%d")<Enter>  " inserts current date
```

37.7 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: Example (Vim)

```
" In insert mode:
Ctrl+v 65    " insert 'A' (ASCII 65)
Ctrl+v 169   " insert '©' (copyright symbol)
Ctrl+v 8364  " insert '€' (euro symbol)
```

37.8 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: Example (Vim)

```
" In insert mode:
Ctrl+k a:    " insert ä
Ctrl+k <<    " insert «
:digraphs    " show all digraphs
```

37.9 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: Example (Vim)

```
" In insert mode:
Ctrl+n        " next completion
Ctrl+p        " previous completion
Ctrl+x Ctrl+f " filename completion
```

37.10 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: Example (Vim)

```
" 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
```

37.11 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: Example (Vim)

```
" In insert mode:
Ctrl+g j  " move down to insertion start column
Ctrl+g k  " move up to insertion start column
```

37.12 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: Example (Vim)

```
" In insert mode:
Ctrl+j  " new line
Ctrl+m  " new line (alternative)
```

37.13 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: Example (Vim)

```
" In insert mode:
Ctrl+i  " insert tab character (same as Tab key)
```

37.14 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: Example (Vim)

```
o  " new line below
O  " new line above
```

37.15 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: Example (Vim)

```
" In insert mode:
Ctrl+r "     " paste from default register
Ctrl+r a     " paste from register 'a'
```

37.16 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: Example (Vim)

```
" In insert mode:
Ctrl+r 0  " paste from yank register
Ctrl+r "  " paste from default register
Ctrl+r +  " paste from system clipboard
```

37.17 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: Example (Vim)

```
" In insert mode:
Ctrl+a  " insert previously typed text
Ctrl+@  " insert previous text and exit insert mode
```

37.18 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: Example (Vim)

```
R   " enter replace mode
gR  " enter virtual replace mode
```

37.19 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: Example (Vim)

```
" In insert mode:
Ctrl+x Ctrl+e  " scroll window down
```

```
Ctrl+x Ctrl+y  " scroll window up
```

37.20 Trigger abbreviation manually

Category: Insert
Tags: abbreviation, trigger, expand

Use `Ctrl+]` to manually trigger abbreviation expansion in insert mode.

Example: Example (Vim)

```
" In insert mode (after setting abbreviation):
:iab teh the
teh<Ctrl+]>  " expands to 'the'
```

CHAPTER 38

Insert mode (advanced)

38.1 Advanced completion modes

Category: Insert Mode Advanced
Tags: completion, advanced, keyword, line

Use specialized completion commands for different contexts.

Example: Example (Vim)

```
" 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
```

38.2 Completion menu navigation

Category: Insert Mode Advanced
Tags: completion, menu, navigate, popup

Navigate and control completion popup menus effectively.

Example: Example (Vim)

```
" 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
```

38.3 Delete operations in insert mode

Category: Insert Mode Advanced
Tags: delete, backspace, word, line

Use various delete operations without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.4 Insert mode abbreviation control

Category: Insert Mode Advanced
Tags: abbreviation, control, expand, prevent

Control abbreviation expansion in insert mode. After setting `:iabbrev teh the`, typing ”teh ” expands to ”the ”.

Example: Example (Vim)

```
" In insert mode:
Ctrl+v <Space>  " insert space without expanding abbreviation
Ctrl+]          " manually expand abbreviation
```

38.5 Insert mode buffer operations

Category: Insert Mode Advanced
Tags: buffer, file, operation, switch

Perform buffer operations without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.6 Insert mode case conversion

Category: Insert Mode Advanced
Tags: case, conversion, upper, lower, toggle

Convert case of text without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.7 Insert mode folding control

Category: Insert Mode Advanced
Tags: fold, unfold, toggle, code

Control code folding without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.8 Insert mode formatting and alignment

Category: Insert Mode Advanced
Tags: format, align, text, paragraph

Format and align text without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.9 Insert mode macro operations

Category: Insert Mode Advanced
Tags: macro, record, replay, register

Work with macros without leaving insert mode. Note: Recording typically requires exiting insert mode first.

Example: Example (Vim)

```
" In insert mode:
Ctrl+o @a       " replay macro 'a'
Ctrl+o @@       " replay last macro
Ctrl+o qa       " start recording macro (then exit insert)
```

38.10 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: Example (Vim)

```
" 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
```

38.11 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: Example (Vim)

```
" In insert mode:
<Left>/<Right>  " move by character
<Up>/<Down>     " move by line
Ctrl+h          " move left (backspace)
Ctrl+l          " move right
```

38.12 Insert mode register shortcuts

Category: Insert Mode Advanced
Tags: register, shortcut, special, paste

Use special register shortcuts for common insert operations.

Example: Example (Vim)

```
" 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
```

38.13 Insert mode search operations

Category: Insert Mode Advanced
Tags: search, find, pattern, navigate

Perform searches without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.14 Insert mode terminal integration

Category: Insert Mode Advanced
Tags: terminal, command, external, shell

Execute external commands and insert their output.

Example: Example (Vim)

```
" 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
```

38.15 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: Example (Vim)

```
" 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)
```

38.16 Insert mode window operations

Category: Insert Mode Advanced
Tags: window, scroll, operation, view

Use window operations without leaving insert mode.

Example: Example (Vim)

```
" 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
```

38.17 Literal character insertion

Category: Insert Mode Advanced
Tags: literal, character, special, escape

Use `Ctrl+v` to insert special characters literally in insert mode.

Example: Example (Vim)

```
" 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
```

38.18 Smart indentation in insert mode

Category: Insert Mode Advanced
Tags: indent, smart, automatic, programming

Control automatic indentation behavior in insert mode.

Example: Example (Vim)

```
" 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
```

38.19 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: Example (Vim)

```
" 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
```

38.20 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: Example (Vim)

```
" 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 39

Integration tips

39.1 API and webhook integration

Category: Integration
Tags: api, webhook, http, rest, automation

Integrate with APIs and webhooks for automation and data exchange.

Example: Example (Vim)

```
" 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")
```

Source: Community contributed

39.2 Browser and documentation integration

Category: Integration
Tags: browser, documentation, help, external, web

Open external documentation and web resources from Neovim.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.3 Build system integration

Category: Integration
Tags: build, make, cmake, gradle, maven, build

Integrate with various build systems and compilation tools.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.4 Cloud platform integration

Category: Integration
Tags: cloud, aws, gcp, azure, kubectl, terraform

Integrate with cloud platforms and infrastructure tools.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.5 Continuous Integration integration

Category: Integration
Tags: ci, cd, github, actions, jenkins, pipeline

Integrate with CI/CD pipelines and automation systems.

Example: Example (Vim)

```
" 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"
```

Source: Community contributed

39.6 Database integration and querying

Category: Integration
Tags: database, sql, query, connection, dadbod

Connect to and query databases directly from Neovim.

Example: Example (Vim)

```
" 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;
```

Source: Community contributed

39.7 Development server integration

Category: Integration
Tags: server, development, hot, reload, livereload

Integrate with development servers and hot reload systems.

Example: Example (Vim)

```
" 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'
```

Source: Community contributed

39.8 Docker and container integration

Category: Integration
Tags: docker, container, dockerfile, build, run

Integrate Docker operations with Neovim development workflow.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.9 Documentation generation integration

Category: Integration
Tags: documentation, generate, doxygen, sphinx, javadoc

Integrate documentation generation tools with editing workflow.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.10 Email and notification integration

Category: Integration
Tags: email, notification, alert, smtp, webhook

Send notifications and emails from Neovim for workflow automation.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.11 External editor integration

Category: Integration
Tags: editor, external, gui, comparison, merge

Integrate with external editors for specific tasks and workflows.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.12 Git integration and workflow

Category: Integration
Tags: git, version, control, fugitive, workflow

Integrate Git operations seamlessly with Neovim editing workflow.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.13 Git integration with gitsigns plugin

Category: Integration
Tags: git, gitsigns, diff, blame, plugin

Use Gitsigns plugin commands for advanced git integration directly in Neovim. **Note**: Requires gitsigns.nvim plugin to be installed.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.14 Issue tracking integration

Category: Integration
Tags: issue, tracking, jira, github, gitlab, bug

Integrate with issue tracking systems for development workflow.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.15 Launch Vim Inside Visual Studio

Category: integration
Tags: external-tools, ide-integration, windows

Use gVim's -P option to embed Vim as an MDI window in Visual Studio, allowing internal editing

Example: Example (Vim)

```
gvim.exe -P "Microsoft Visual C++" --servername MDI_VIM
```

Example: Example (Neovim)

```
-- Lua equivalent for command-line launch
vim.fn.system({'gvim', '-P', 'Microsoft Visual C++', '--servername',
↪  'MDI_VIM'})
```

Source: vim.fandom.com

39.16 Monitoring and logging integration

Category: Integration
Tags: monitoring, logging, logs, metrics, observability

Integrate with monitoring and logging systems for development insights.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.17 Open Files in gVim from Source Navigator

Category: integration
Tags: external-tool, editor-integration, development

Configure Source-Navigator to open files directly in gVim with cursor positioning

Example: Example (Vim)

```
gvim "+call cursor(%l,%c)" %f
```

Example: Example (Neovim)

```
vim.fn.system('gvim "+call cursor(' .. line .. ',' .. col .. ')" ' ..
↪  filename)
```

Source: vim.fandom.com

39.18 Package manager integration

Category: Integration
Tags: package, manager, npm, pip, cargo, gem

Integrate with various package managers for dependency management.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.19 REST API testing integration

Category: Integration
Tags: rest, api, http, curl, testing, client

Test REST APIs directly from Neovim using HTTP client functionality.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.20 SSH and remote development integration

Category: Integration
Tags: ssh, remote, development, server, connection

Integrate SSH operations for remote development workflows.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.21 System clipboard integration

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

Seamlessly integrate with system clipboard for cross-application workflows.

Example: Example (Neovim)

```
-- 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'))
```

Source: Community contributed

39.22 Terminal multiplexer integration

Category: Integration
Tags: tmux, screen, multiplexer, pane, session

Integrate with terminal multiplexers for enhanced workflow management.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.23 Testing framework integration

Category: Integration
Tags: testing, framework, jest, pytest, rspec, junit

Integrate with various testing frameworks for efficient testing workflow.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.24 Version control system integration

Category: Integration
Tags: vcs, svn, mercurial, bazaar, perforce

Integrate with various version control systems beyond Git.

Example: Example (Vim)

```
" 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
```

Source: Community contributed

39.25 Visual Studio .NET Vim Integration

Category: integration
Tags: external-tools, ide-integration

Enhanced Vim integration in Visual Studio .NET, with ability to launch selected files directly in Vim

Example: Example (Vim)

```
gvim.exe +$(CurLine) "$(ItemPath)" --remote-tab-silent
```

Example: Example (Neovim)

```
-- Lua equivalent for file launch
vim.fn.system({'gvim', '+' .. vim.fn.line('.'), vim.fn.expand('%:p'),
↪  '--remote-tab-silent'})
```

Source: vim.fandom.com

CHAPTER 40

Lsp

40.1 LSP code actions

Category: LSP
Tags: lsp, actions, refactor, fix

Use `:lua vim.lsp.buf.code_action()` to show available code actions.

Example: Example (Vim)

```
:lua vim.lsp.buf.code_action()
```

40.2 LSP format document

Category: LSP
Tags: lsp, format, style, beautify

Use `:lua vim.lsp.buf.format()` to format current buffer using LSP.

Example: Example (Vim)

```
:lua vim.lsp.buf.format()
```

40.3 LSP implementation

Category: LSP
Tags: lsp, implementation, goto

Use `gi` to jump to implementation of symbol under cursor.

Example: Example (Vim)

```
gi  " jump to implementation
```

40.4 LSP incoming calls

Category: LSP
Tags: lsp, calls, incoming, hierarchy

Use `:lua vim.lsp.buf.incoming_calls()` to show incoming call hierarchy.

Example: Example (Vim)

```
:lua vim.lsp.buf.incoming_calls()
```

40.5 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: Example (Vim)

```
:lua print(vim.inspect(vim.lsp.buf.list_workspace_folders()))
```

40.6 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: Example (Vim)

```
:lua vim.lsp.buf.remove_workspace_folder()
```

40.7 LSP rename

Category: LSP
Tags: lsp, rename, refactor

Use `:lua vim.lsp.buf.rename()` to rename symbol under cursor across the project.

Example: Example (Vim)

```
:lua vim.lsp.buf.rename()
```

40.8 LSP show signature help

Category: LSP
Tags: lsp, signature, parameters, help

Use `:lua vim.lsp.buf.signature_help()` to show function signature help.

Example: Example (Vim)

```
:lua vim.lsp.buf.signature_help()
```

40.9 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: Example (Vim)

```
" Check if inlay hints are available:
:lua print(vim.lsp.inlay_hint.is_enabled())
```

Example: Example (Neovim)

```
-- 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,
})
```

CHAPTER 41

Lua

41.1 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: Example (Neovim)

```
-- 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, {})
```

41.2 Access environment variables in Lua

Category: Lua
Tags: lua, environment, vim.env

Use `vim.env` to access and set environment variables.

Example: Example (Neovim)

```
-- Read environment variable
local home = vim.env.HOME
local editor = vim.env.EDITOR

-- Set environment variable
vim.env.MY_VAR = "value"
```

41.3 Call Lua functions from Vimscript

Category: Lua
Tags: lua, vimscript, interop

Use `luaeval()` or `v:lua` to call Lua functions from Vimscript.

Example: Example (Vim)

```
" 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')
```

41.4 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: Example (Neovim)

```
-- Call built-in function
local line_count = vim.fn.line('$')

-- Call user function
local result = vim.fn.MyCustomFunction('arg1', 'arg2')
```

41.5 Check if list is empty

Category: Lua
Tags: lua, tables, utilities

Use `vim.tbl_isempty()` to check if a table is empty.

Example: Example (Neovim)

```
local my_table = {}

if vim.tbl_isempty(my_table) then
  print("Table is empty")
end
```

41.6 Check if table contains key

Category: Lua
Tags: lua, tables, utilities

Use `vim.tbl_contains()` to check if a table contains a value.

Example: Example (Neovim)

```
local modes = {'n', 'v', 'i'}

if vim.tbl_contains(modes, 'n') then
  print("Normal mode included")
end
```

41.7 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: Example (Neovim)

```
-- init.lua example
vim.opt.number = true
vim.opt.relativenumber = true
```

41.8 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: Example (Neovim)

```
---@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
```

41.9 Create autocommand groups

Category: Lua
Tags: lua, autocommands, groups, api

Use `vim.api.nvim_create_augroup()` to organize autocommands into groups.

Example: Example (Neovim)

```
-- 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,
})
```

41.10 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: Example (Neovim)

```
-- 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,
})
```

41.11 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: Example (Neovim)

```
-- 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"
})
```

41.12 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: Example (Neovim)

```
-- 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"
```

41.13 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: Example (Neovim)

```
-- 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" })
```

41.14 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: Example (Neovim)

```
-- 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!
```

41.15 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: Example (Neovim)

```
-- 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)
```

41.16 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: Example (Neovim)

```
-- 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',
})
```

41.17 Debug Lua values

Category: Lua
Tags: lua, debug, inspect

Use `vim.inspect()` to debug and pretty-print Lua values.

Example: Example (Vim)

```
:lua print(vim.inspect(vim.fn.getbufinfo()))
```

41.18 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: Example (Neovim)

```
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 } }
```

41.19 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: Example (Neovim)

```
-- 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
})
```

41.20 Defer function execution

Category: Lua
Tags: lua, defer, utilities

Use `vim.defer_fn()` to execute a function after a delay.

Example: Example (Neovim)

```
-- 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)
```

41.21 Execute Lua from command line

Category: Lua
Tags: lua, command-line, inline

Use `:lua` with `=` to evaluate and print expressions quickly.

Example: Example (Vim)

```
:lua =vim.version()
:lua =vim.fn.expand('%:p')
:lua =package.path
```

41.22 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: Example (Neovim)

```
-- Single command
vim.cmd("colorscheme habamax")

-- Multiple commands
vim.cmd([[
  set number
  set relativenumber
  highlight Normal guibg=NONE
]])
```

41.23 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: Example (Neovim)

```
-- 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
)
```

41.24 Filter and map tables

Category: Lua
Tags: lua, tables, functional

Use `vim.tbl_filter()` and `vim.tbl_map()` for functional table operations.

Example: Example (Neovim)

```
-- 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}
```

41.25 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: Example (Neovim)

```
-- 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,
})
```

41.26 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: Example (Neovim)

```
-- Get clients for current buffer
local clients = vim.lsp.get_clients({ bufnr = 0 })

for _, client in ipairs(clients) do
  print("Client: " .. client.name)
end
```

41.27 Get current mode in Lua

Category: Lua
Tags: lua, mode, api

Use `vim.api.nvim_get_mode()` to get the current editor mode.

Example: Example (Neovim)

```
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
```

41.28 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: Example (Neovim)

```
---@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))
```

41.29 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: Example (Neovim)

```
-- 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"
```

41.30 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: Example (Neovim)

```
---@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"
```

41.31 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: Example (Neovim)

```
-- Load lua/mymodule.lua
local mymodule = require('mymodule')

-- Load lua/other_modules/anothermodule.lua
local another = require('other_modules.anothermodule')
```

41.32 Lua keymaps

Category: Lua
Tags: lua, keymap, mapping

Use `vim.keymap.set()` to create keymaps with inline Lua functions.

Example: Example (Vim)

```
:lua vim.keymap.set("n", "<leader>hi", function() print("Hello!") end)
```

41.33 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: Example (Neovim)

```
-- 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"
```

41.34 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: Example (Neovim)

```
-- 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
```

41.35 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: Example (Neovim)

```
-- 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))
```

41.36 Reload Lua modules during development

Category: Lua
Tags: lua, modules, development, reload

Clear `package.loaded` to force reload of a module during development.

Example: Example (Neovim)

```
-- 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')
```

41.37 Run current Lua file

Category: Lua
Tags: lua, file, execute

Use `:luafile %` to execute the current Lua file inside Neovim.

Example: Example (Vim)

```
:luafile %  " run current Lua file
```

41.38 Run inline Lua code

Category: Lua
Tags: lua, inline, execute

Use `:lua` to run Lua code directly in Neovim.

Example: Example (Vim)

```
:lua print("Hello from Lua!")
```

41.39 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: Example (Neovim)

```
local ok, module = pcall(require, 'optional_module')
if ok then
  module.setup()
else
  print("Module not found: " .. module)
end
```

41.40 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: Example (Neovim)

```
-- 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)
```

41.41 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: Example (Neovim)

```
-- Current buffer
vim.bo.filetype = 'lua'
vim.bo.expandtab = true

-- Specific buffer
vim.bo[5].tabstop = 2
```

41.42 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: Example (Neovim)

```
-- Current buffer
vim.b.current_syntax = 'lua'

-- Specific buffer
vim.b[5].custom_data = { foo = "bar" }
```

41.43 Set cursor position in Lua

Category: Lua
Tags: lua, cursor, navigation

Use `vim.api.nvim_win_set_cursor()` to set cursor position programmatically.

Example: Example (Neovim)

```
-- 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]
```

41.44 Set global variables in Lua

Category: Lua
Tags: lua, variables, vim.g

Use `vim.g` to set global variables with native Lua types.

Example: Example (Neovim)

```
-- Set global variable
vim.g.mapleader = ' '
vim.g.my_config = { key = "value", enabled = true }

-- Read global variable
local leader = vim.g.mapleader
```

41.45 Set options with vim.o

Category: Lua
Tags: lua, options, vim.o

Use `vim.o` for direct variable-like access to global options.

Example: Example (Neovim)

```
-- Set global option
vim.o.ignorecase = true
vim.o.smartcase = true

-- More concise than vim.opt for simple assignments
vim.o.updatetime = 250
```

41.46 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: Example (Neovim)

```
-- 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')
```

41.47 Set tabpage-local variables

Category: Lua
Tags: lua, variables, tabpage, vim.t

Use `vim.t` to set tabpage-local variables.

Example: Example (Neovim)

```
-- Current tabpage
vim.t.tab_name = "Main"

-- Specific tabpage
vim.t[2].custom_data = {}
```

41.48 Set window-local options

Category: Lua
Tags: lua, options, window, vim.wo

Use `vim.wo` to set window-local options.

Example: Example (Neovim)

```
-- Current window
vim.wo.number = true
vim.wo.wrap = false

-- Specific window
vim.wo[1001].cursorline = true
```

41.49 Set window-local variables

Category: Lua
Tags: lua, variables, window, vim.w

Use `vim.w` to set window-local variables.

Example: Example (Neovim)

```
-- Current window
vim.w.quickfix_title = "My Results"

-- Specific window
vim.w[1001].custom_setting = true
```

41.50 Split and join strings

Category: Lua
Tags: lua, strings, utilities

Use `vim.split()` to split strings into tables.

Example: Example (Neovim)

```
-- 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 })
```

41.51 Trim whitespace from strings

Category: Lua
Tags: lua, strings, utilities

Use `vim.trim()` to remove leading and trailing whitespace.

Example: Example (Neovim)

```
local cleaned = vim.trim("  hello world  ")
-- Result: "hello world"
```

41.52 Use Lua heredoc in Vimscript

Category: Lua
Tags: lua, vimscript, heredoc

Use Lua heredoc syntax to write Lua code blocks in `init.vim`.

Example: Example (Vim)

```
lua << EOF
  local function hello()
    print("Hello from Lua in Vimscript!")
  end
  hello()
EOF
```

41.53 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: Example (Neovim)

```
-- 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)
```

41.54 Use vim.notify for notifications

Category: Lua
Tags: lua, notifications, ui

Use `vim.notify()` to show notifications with different log levels.

Example: Example (Neovim)

```
-- 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)
```

41.55 Use vim.ui.input for user input

Category: Lua
Tags: lua, ui, input

Use `vim.ui.input()` to prompt user for text input.

Example: Example (Neovim)

```
vim.ui.input({
  prompt = 'Enter your name: ',
  default = 'Anonymous',
}, function(input)
  if input then
    print('Hello, ' .. input)
  end
end)
```

41.56 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: Example (Neovim)

```
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
)
```

41.57 Validate function arguments

Category: Lua
Tags: lua, validation, utilities

Use `vim.validate()` to validate function arguments with clear error messages.

Example: Example (Neovim)

```
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
```

41.58 View loaded Lua modules

Category: Lua
Tags: modules, loaded, debug

Use `package.loaded` to view all loaded Lua modules.

Example: Example (Vim)

```
:lua print(vim.inspect(package.loaded))
```

41.59 Work with buffer text in Lua

Category: Lua
Tags: lua, buffer, text, api

Use buffer API functions to read and modify buffer content.

Example: Example (Neovim)

```
-- 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")
```

41.60 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: Example (Neovim)

```
-- 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 42

Macros

42.1 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: Example (Vim)

```
:let @a='<Ctrl+R><Ctrl+R>a'  " edit macro 'a' inline
```

42.2 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: Example (Vim)

```
@a  " execute macro 'a'
@@  " repeat last macro
```

42.3 Macro for data transformation

Category: Macros
Tags: macro, transform, data, format

Use macros to transform structured data formats efficiently.

Example: Example (Vim)

```
" 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
```

42.4 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: Example (Vim)

```
" 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
```

42.5 Quick macro shortcuts

Category: Macros
Tags: macro, shortcut, mapping, space

Set up convenient mappings for macro execution and recording.

Example: Example (Vim)

```
" 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>
```

42.6 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: Example (Vim)

```
qqq          " clear register q
qq           " start recording macro q
" ... your editing commands ...
@q           " recursive call to self
q            " stop recording
@q           " execute recursive macro
```

42.7 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: Example (Vim)

```
" 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
```

42.8 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: Example (Vim)

```
:args *.txt         " load all txt files
:argdo normal @q    " run macro q on all files
:argdo update       " save all changed files
```

42.9 Run macro over visual selection

Category: Macros
Tags: macro, visual, selection

Use `:'<,'>normal @q` to run macro q over visual selection.

Example: Example (Vim)

```
:'<,'>normal @q  " run macro q on selection
```

42.10 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: Example (Vim)

```
let @a='ddp'  " save line swap macro permanently
```

42.11 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: Example (Vim)

```
:reg     " view all registers
:reg a   " view macro in register 'a'
```

CHAPTER 43

Marks

43.1 Cut or Copy Lines Without Line Counting

Category: marks
Tags: navigation, copying, editing

Easily copy or cut lines using marks without manually counting line numbers

Example: Example (Vim)

```
mk      " Mark current line with 'k'
y'k     " Yank from mark to current position
d'k     " Delete from mark to current position
```

Example: Example (Neovim)

```
-- Mark current line with 'k'
vim.fn.setmark('k')
-- Yank from mark to current position
vim.cmd('normal y`k')
-- Delete from mark to current position
vim.cmd('normal d`k')
```

Source: vim.fandom.com

43.2 Master Vim Marks for Quick Navigation

Category: marks
Tags: navigation, bookmarks, jumping

Use lowercase marks within a file and uppercase marks across files to quickly jump between locations. Uppercase marks persist between Vim sessions when viminfo is enabled.

Example: Example (Vim)

```
" Set a mark in current file
ma

" Jump to mark
```

```
'a

" Jump to specific file mark
'A
```

Example: Example (Neovim)

```
-- Set a mark in current file
vim.cmd('ma')

-- Jump to mark within current file
vim.cmd("'a")

-- Jump to global file mark
vim.cmd("'A")
```

Source: vim.fandom.com

43.3 Navigate Between Marks Efficiently

Category: marks
Tags: navigation, marks, movement

Quickly jump between marks using built-in navigation commands

Example: Example (Vim)

```
" Jump to next/previous marked lines
]'  " Next line with a mark
['  " Previous line with a mark

" Jump to next/previous marks
]`  " Next mark
[`  " Previous mark
```

Example: Example (Neovim)

```
-- These are Ex commands, so use vim.cmd
-- Jump to next/previous marked lines
vim.cmd(']\'')
vim.cmd('[\'')

-- Jump to next/previous marks
vim.cmd(']`')
vim.cmd('[`')
```

Source: vim.fandom.com

43.4 Navigate Between Marks Quickly

Category: marks
Tags: navigation, movement

Use special commands to jump between marks efficiently, with support for count-based jumps.

Example: Example (Vim)

```
" Jump to next/previous lowercase mark
]'
['
```

Example: Example (Neovim)

```
-- Note: These are Vim commands that can be run in Lua
vim.cmd(']\'')
vim.cmd('[\'')
```

Source: vim.fandom.com

43.5 Quick File Bookmarking with Context

Category: marks
Tags: bookmarking, file-tracking, context-capture

Create a custom function to bookmark current file location with timestamp, current directory, and line details

Example: Example (Vim)

```
function! MoshBookmark()
  redir >> ~/.vims
  echo
  echo strftime("%Y-%b-%d %a %H:%M")
  echo "cd ". $PWD
  echo "vim ". expand("%:p").':'.line('.')
  echo ' word='.expand("<cword>")
  echo ' cline='.getline('.')
  redir END
endfunction
:command! MoshBookmark :call MoshBookmark()
```

Example: Example (Neovim)

```
function _G.mosh_bookmark()
  local file = io.open(vim.fn.expand('~/.vims'), 'a')
  if file then
```

```
    file:write('\n' .. os.date('%Y-%b-%d %a %H:%M') .. '\n')
    file:write('cd ' .. vim.fn.getcwd() .. '\n')
    file:write('vim ' .. vim.fn.expand('%:p') .. ':' .. vim.fn.line('.') ..
    ↪  '\n')
    file:write(' word=' .. vim.fn.expand('<cword>') .. '\n')
    file:write(' cline=' .. vim.fn.getline('.') .. '\n')
    file:close()
  end
end

vim.api.nvim_create_user_command('MoshBookmark', _G.mosh_bookmark, {})
```

Source: vim.fandom.com

43.6 Remember Last Position in Help Files

Category: marks
Tags: help-navigation, persistent-marks, workflow

Automatically mark the last position in Vim help files and easily return to it across sessions

Example: Example (Vim)

```
au BufLeave * if &ft == "help" | mark H | endif
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufLeave', {
  callback = function()
    if vim.bo.filetype == 'help' then
      vim.cmd('mark H')
    end
  end
})
```

Source: vim.fandom.com

43.7 Use Marks for Quick Navigation

Category: marks
Tags: navigation, bookmarks, file-jumping

Set and use marks to quickly jump between locations in files and across different files

Example: Example (Vim)

```
" Set a local mark in current file
ma

" Set a global mark across files
mA

" Jump to marks
'a  " Jump to line of mark a
`a  " Jump to exact position of mark a
`A  " Jump to global mark A in its file
```

Example: Example (Neovim)

```
-- Set local mark
vim.cmd('ma')

-- Set global mark
vim.cmd('mA')

-- Jump to marks
vim.cmd("'a")  -- Jump to line of mark a
vim.cmd("`a")  -- Jump to exact position of mark a
vim.cmd("`A")  -- Jump to global mark A in its file
```

Source: vim.fandom.com

43.8 Use Special Marks for Quick Reference

Category: marks
Tags: navigation, editing-history

Utilize Vim's built-in special marks to quickly jump to recent edit or change locations

Example: Example (Vim)

```
" Jump to last edit location
`.

" Jump to last exit location
`"
```

Example: Example (Neovim)

```
-- These can be run as Vim commands in Lua
vim.cmd('`.')
vim.cmd('`"')
```

Source: vim.fandom.com

43.9 Useful Special Marks for Quick Navigation

Category: marks
Tags: navigation, special-marks, editing

Use special marks to quickly jump to recent editing locations

Example: Example (Vim)

```
`` .  " Last change location
`` "  " Last exit location
`` 0  " Last file edited when exiting Vim
`` ''  " Previous jump location
`` `   " Previous exact location
```

Example: Example (Neovim)

```
-- Jump to special marks
vim.cmd('`.')
vim.cmd('`"')
vim.cmd('`0')
vim.cmd('``')  -- Previous jump location
vim.cmd('`^')  -- Last insertion point
```

Source: vim.fandom.com

CHAPTER 44

Modern neovim api

44.1 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: Example (Neovim)

```
-- 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
})
```

44.2 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: Example (Neovim)

```
-- 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
})
```

44.3 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: Example (Neovim)

```
-- 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 })
```

44.4 Custom diagnostic configuration

Category: Diagnostics
Tags: diagnostic, lsp, configuration, signs, virtual

Use `vim.diagnostic.config()` to customize how diagnostics are displayed and behave.

Example: Example (Neovim)

```
-- 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
```

44.5 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: Example (Neovim)

```
-- 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 45

Movement

45.1 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: Example (Vim)

```
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)
```

45.2 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: Example (Vim)

```
h  " move left
j  " move down
k  " move up
l  " move right
```

45.3 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: Example (Vim)

```
zz  " center line
zt  " line to top
zb  " line to bottom
```

45.4 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: Example (Vim)

```
g;       " previous change
g,       " next change
:changes " show change list
```

45.5 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: Example (Vim)

```
fa  " find next 'a'
Fa  " find previous 'a'
ta  " move to before next 'a'
Ta  " move to after previous 'a'
```

45.6 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: Example (Vim)

```
gg  " first line
G   " last line
```

45.7 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: Example (Vim)

```
Ctrl+o  " previous location
Ctrl+i  " next location
:jumps  " show jump list
```

45.8 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: Example (Vim)

```
5↑    " jump 5 lines up
5↓    " jump 5 lines down
10↑   " jump 10 lines up
3→    " move 3 characters right
```

45.9 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: Example (Vim)

```
gd  " go to definition
gD  " go to declaration
```

45.10 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: Example (Vim)

```
42G   " jump to line 42
:42   " jump to line 42
```

45.11 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: Example (Vim)

```
g_     " go to last non-blank character
yg_    " yank to last non-blank character (no trailing spaces)
dg_    " delete to last non-blank character
```

45.12 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: Example (Vim)

```
0  " line start
^  " first non-blank
$  " line end
```

45.13 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: Example (Vim)

```
42gg  " go to line 42
100G  " go to line 100
1G    " go to first line (same as gg)
```

45.14 Matching brackets

Category: Movement
Tags: brackets, matching, navigation

Use `%` to jump to the matching bracket, parenthesis, or brace.

Example: Example (Vim)

```
%  " jump to matching bracket
```

45.15 Middle of screen

Category: Movement
Tags: middle, screen, position

Use `M` to move cursor to the middle line of the screen.

Example: Example (Vim)

```
M  " move to middle of screen
```

45.16 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: Example (Vim)

```
Ctrl+f  " forward full page
Ctrl+b  " backward full page
Ctrl+d  " down half page
Ctrl+u  " up half page
```

45.17 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: Example (Vim)

```
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)
```

45.18 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: Example (Vim)

```
{  " previous paragraph
}  " next paragraph
```

45.19 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: Example (Vim)

```
;  " repeat search forward
,  " repeat search backward
```

45.20 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: Example (Vim)

```
H  " move to top of screen
L  " move to bottom of screen
```

45.21 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: Example (Vim)

```
Ctrl+e  " scroll down
Ctrl+y  " scroll up
```

45.22 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: Example (Vim)

```
(  " previous sentence
)  " next sentence
```

45.23 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: 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: Example (Vim)

```
Ctrl+z  " suspend to shell
jobs    " list suspended jobs
fg #5   " resume job #5
```

45.24 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: Example (Vim)

```
w  " next word start
e  " end of word
b  " previous word start
```

45.25 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: Example (Vim)

```
W  " next WORD (whitespace-separated)
E  " end of WORD
B  " previous WORD
```

45.26 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: Example (Vim)

```
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
```

45.27 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: Example (Vim)

```
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)
```

45.28 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: Example (Vim)

```
z20<Enter>  " make window 20 lines high
z+          " cursor to line below window
z^          " cursor to line above window
```

CHAPTER 46

Navigation

46.1 Advanced Cursor Jumping Techniques

Category: navigation
Tags: marks, jump-list, cursor-movement

Use marks and jump list to quickly navigate between different locations in a file

Example: Example (Vim)

```
" Set mark with mx
" Jump to mark with 'x
" Return to previous jump with ''
```

Example: Example (Neovim)

```
-- Set a mark
vim.cmd('normal! mx')

-- Jump to a specific mark
vim.cmd('normal! `x')

-- Return to previous jump
vim.cmd('normal! \'\'') -- Two single quotes
```

Source: vim.fandom.com

46.2 Advanced File and Buffer Navigation

Category: navigation
Tags: buffer-management, file-navigation, marks

Powerful navigation techniques using marks and movement history

Example: Example (Vim)

```
'.       " jump to last modification line
`.       " jump to exact spot in last modification line
<C-O>    " retrace movements backward
```

```
<C-I>    " retrace movements forward
:ju(mps) " list of your movements
```

Example: Example (Neovim)

```
-- Lua equivalents for advanced navigation
-- These are primarily Vim native commands, so use vim.cmd
vim.keymap.set('n', "'.", "'.", { desc = 'Jump to last modification line' })
vim.keymap.set('n', '`.', '`.', { desc = 'Jump to exact modification spot'
↪  })
vim.keymap.set('n', '<C-o>', '<C-o>', { desc = 'Navigate backward in jump
↪  history' })
vim.keymap.set('n', '<C-i>', '<C-i>', { desc = 'Navigate forward in jump
↪  history' })
```

Source: vim.fandom.com

46.3 Advanced Find and Till Commands

Category: navigation
Tags: search, motion, text-editing

Use 'find' and 'till' commands to quickly move and manipulate text within a line

Example: Example (Vim)

```
# Find and till command examples
fx   # Find next 'x' in line
tC   # Move till just before next 'C'
ctx  # Change till next 'x'
```

Example: Example (Neovim)

```
-- Vim's built-in find commands work the same in Neovim
-- You can create custom keymaps to enhance these motions
vim.keymap.set('n', '<leader>fx', 'fx', { desc = 'Find next x in line' })
vim.keymap.set('n', '<leader>tc', 'tC', { desc = 'Move till just before next
↪  C' })
```

Source: vim.fandom.com

46.4 Advanced Movement and Modification Markers

Category: navigation
Tags: marks, navigation, cursor-movement

Jump to last modification and retrace movement history

Example: Example (Vim)

```
'.      " jump to last modification line
`.      " jump to exact spot in last modification line
<C-O>   " retrace movements backward
<C-I>   " retrace movements forward
:ju(mps)  " list of your movements
```

Example: Example (Neovim)

```
-- Most of these are built-in Vim commands
-- For more advanced mark management in Neovim
vim.keymap.set('n', '<leader>m', function()
  vim.cmd('marks')  -- Show mark list
end, { desc = 'Show marks' })
```

Source: vim.fandom.com

46.5 Advanced Search and Movement

Category: navigation
Tags: search, movement, cursor

Enhanced word searching and cursor movement techniques

Example: Example (Vim)

```
* # g* g#      " find word under cursor (forwards/backwards)
%              " match brackets {}[]()
'. `. " jump to last modification line
```

Example: Example (Neovim)

```
-- Lua alternatives for advanced search and movement
-- Most of these are vim commands that work in Neovim
-- vim.cmd('normal! *')   -- search forward for word under cursor
-- vim.cmd('normal! #')   -- search backward for word under cursor
-- vim.cmd('normal! %')   -- match corresponding bracket
-- vim.cmd('normal! `.'   -- jump to last modification
```

Source: vim.fandom.com)

46.6 Advanced Tab Navigation Shortcuts

Category: navigation
Tags: tabs, key-mapping, window-management

Powerful tab navigation mappings to quickly move between and manage tabs

Example: Example (Vim)

```
" Tab navigation mappings
nnoremap <C-Left> :tabprevious<CR>
nnoremap <C-Right> :tabnext<CR>
nnoremap <silent> <A-Left> :tabm -1<CR>
nnoremap <silent> <A-Right> :tabm +1<CR>
```

Example: Example (Neovim)

```
-- Tab navigation mappings
vim.keymap.set('n', '<C-Left>', ':tabprevious<CR>', { noremap = true, silent
↪  = true })
vim.keymap.set('n', '<C-Right>', ':tabnext<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<A-Left>', ':tabm -1<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<A-Right>', ':tabm +1<CR>', { noremap = true, silent =
↪  true })
```

Source: vim.fandom.com_file_in_new_tab)

46.7 Advanced Tag Search and Browsing

Category: navigation
Tags: tags, code-search, fuzzy-finding

Use pattern matching and flexible tag browsing techniques to explore code efficiently

Example: Example (Vim)

```
" Search tags with pattern
:tag /<pattern>

" Browse tag list
:tfirst    " Go to first tag
:tnext     " Go to next tag
:tprevious " Go to previous tag
```

Example: Example (Neovim)

```
-- Search tags with pattern
-- Note: Requires setting up a tags file first
vim.keymap.set('n', '<leader>tp', function()
  vim.cmd('tag /' .. vim.fn.input('Tag pattern: '))
end, { desc = 'Search tags by pattern' })

-- Tag list navigation helpers
vim.keymap.set('n', '<leader>tf', '<cmd>tfirst<CR>', { desc = 'First tag' })
vim.keymap.set('n', '<leader>tn', '<cmd>tnext<CR>', { desc = 'Next tag' })
```

```
vim.keymap.set('n', '<leader>tp', '<cmd>tprevious<CR>', { desc = 'Previous
↪  tag' })
```

Source: vim.fandom.com

46.8 Alternative Help Navigation Mapping

Category: navigation
Tags: help, key-mapping, navigation

Additional key mappings for navigating help documentation using Alt keys

Example: Example (Neovim)

```
-- Alternative help navigation using Alt keys
vim.keymap.set('n', '<M-left>', '<C-o>', { desc = 'Go back in help
↪  navigation' })
vim.keymap.set('n', '<M-down>', '<C-]>', { desc = 'Follow help link' })
```

Source: vim.fandom.com

46.9 Alternative Tag Navigation

Category: navigation
Tags: tags, window-management

Use built-in Vim command to open tag in a new window

Example: Example (Neovim)

```
-- Use CTRL-W ] to open tag in a new split window
-- This is a built-in Vim/Neovim feature
```

Source: vim.fandom.com

46.10 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: Example (Vim)

```
:ls          " list all buffers
:b#          " switch to previous buffer
Ctrl+^       " alternate between current and previous buffer
:b partial   " switch to buffer matching partial name
```

Source: Community contributed

46.11 Center Cursor on Movement

Category: navigation
Tags: key-mapping, movement

Automatically center the screen whenever moving up or down with j/k keys

Example: Example (Vim)

```
" Center cursor on movement
nnoremap j jzz
nnoremap k kzz
```

Example: Example (Neovim)

```
-- Center cursor on movement
vim.keymap.set('n', 'j', 'jzz', { noremap = true })
vim.keymap.set('n', 'k', 'kzz', { noremap = true })
```

Source: vim.fandom.com

46.12 Center Screen on Search Navigation

Category: navigation
Tags: search, screen-positioning, key-mapping

Automatically center the screen when navigating search results, improving readability and focus

Example: Example (Vim)

```
" Map search navigation keys to always center screen
nnoremap n nzz
nnoremap N Nzz
nnoremap * *zz
nnoremap # #zz
nnoremap g* g*zz
nnoremap g# g#zz
```

Example: Example (Neovim)

```
-- Center screen on search navigation
vim.keymap.set('n', 'n', 'nzz', { desc = 'Next search result, center screen'
↪  })
vim.keymap.set('n', 'N', 'Nzz', { desc = 'Previous search result, center
↪  screen' })
vim.keymap.set('n', '*', '*zz', { desc = 'Search word forward, center
↪  screen' })
vim.keymap.set('n', '#', '#zz', { desc = 'Search word backward, center
↪  screen' })
```

Source: vim.fandom.com

46.13 Configure Matchit for ColdFusion Tag Jumping

Category: navigation
Tags: matchit, language-support, tag-navigation

Enable % key to jump between matching ColdFusion tags like <cfif> and </cfif>

Example: Example (Vim)

```
" In ~/.vim/ftplugin/cf.vim
if exists("loaded_matchit")
    let b:match_words = '<cfif\>:\<cfelseif\>:\<cfelse\>:\</cfif>,'
                      \ . '<cfloop\>:\</cfloop>,'
                      \ . '<cfoutput\>:\</cfoutput>'
endif
```

Example: Example (Neovim)

```
-- In ~/.config/nvim/ftplugin/cf.lua
if vim.g.loaded_matchit then
  vim.b.match_words = table.concat({
    '<cfif\>:<cfelseif\>:<cfelse\>:</cfif>',
    '<cfloop\>:</cfloop>',
    '<cfoutput\>:</cfoutput>'
  }, ',')
end
```

Source: vim.fandom.com

46.14 Context-Sensitive H/L Movement

Category: navigation
Tags: key-mapping, window-navigation, scroll

Enhance H and L keys to act as context-sensitive page up/down when at window edges,

providing more efficient window navigation

Example: Example (Vim)

```
function! Hcontext()
  if (winline() == 1 && line('.') != 1)
    exe "normal! \<pageup>H"
  else
    exe "normal! H"
  endif
endfunc

function! Lcontext()
  if (winline() == winheight(0) && line('.') != line('$'))
    exe "normal! \<pagedown>L"
  else
    exe "normal! L"
  endif
endfunc

noremap H :call Hcontext()<CR>
noremap L :call Lcontext()<CR>
```

Example: Example (Neovim)

```
function _G.hcontext()
  if (vim.fn.winline() == 1 and vim.fn.line('.') ~= 1) then
    vim.cmd('normal! \<PageUp>H')
  else
    vim.cmd('normal! H')
  end
end

function _G.lcontext()
  if (vim.fn.winline() == vim.fn.winheight(0) and vim.fn.line('.') ~=
  ↪  vim.fn.line('$')) then
    vim.cmd('normal! \<PageDown>L')
  else
    vim.cmd('normal! L')
  end
end

vim.keymap.set('n', 'H', _G.hcontext)
vim.keymap.set('n', 'L', _G.lcontext)
```

Source: vim.fandom.com

46.15 Create Custom Help-Style Tags in Source Code

Category: navigation
Tags: ctags, custom-tags, code-navigation

Use Ctags to create custom help-style tag navigation in non-help files by defining a new tag type

Example: Example (Vim)

```
" Add to ~/.ctags
--regex-java=/\*([A-Za-z0-9_\-]*)\*/\1/h,htag/

" Add to ~/.vimrc for taglist support
let tlist_java_settings = 'java;p:package;c:class;i:interface;' .
                              \ 'f:field;m:method;h:htag'
```

Example: Example (Neovim)

```
-- Equivalent configuration would typically be done in ctags config
-- For Neovim, you can use telescope or LSP for advanced navigation

-- Example of setting up custom tag navigation in Lua
require('telescope').setup({
  extensions = {
    -- Configure custom tag navigation
  }
})
```

Source: vim.fandom.com

46.16 Create Custom Tags with Line and Column Precision

Category: navigation
Tags: ctags, tag-navigation, custom-indexing

Generate a custom tags file that allows precise jumping to specific line and column positions in files, useful for complex project navigation

Example: Example (Vim)

```
function! MakeTags(filespec, pattern)
  let tags = []
  for file in glob(a:filespec, 0, 1)
    for hit in SearchFile(file, a:pattern)
      call add(tags, printf("%s\t%s\t/\%%%dl\%%%dc/", hit[2], file, hit[0],
      ↪  hit[1]))
    endfor
  endfor
  return sort(tags)
endfunction
```

Example: Example (Neovim)

```
function MakeTags(filespec, pattern)
  local tags = {}
  local files = vim.fn.glob(filespec, false, true)
  for _, file in ipairs(files) do
    local hits = SearchFile(file, pattern)
    for _, hit in ipairs(hits) do
      table.insert(tags, string.format('%s\t%s\t/\%%%dl\%%%dc/', hit[3],
      ↪  file, hit[1], hit[2]))
    end
  end
  table.sort(tags)
  return tags
end
```

Source: vim.fandom.com

46.17 Dynamically Build Tags from Path

Category: navigation
Tags: tags, development, path-management

Automatically build tag files from all directories in your path, making it easier to navigate large codebases with multiple include directories

Example: Example (Vim)

```
function! BuildTagsFromPath()
python << EOF
import sys
import vim
paths = vim.eval("&path")
pathsSplit = paths.split(",")

tags = vim.eval("&tags")
for path in pathsSplit:
  tags += "%s/tags," % (path)
cmdSetTags = "set tags=%s" % tags
vim.command(cmdSetTags)
EOF
endfunction

call BuildTagsFromPath()
```

Example: Example (Neovim)

```
local function build_tags_from_path()
  local paths = vim.o.path
  local path_list = vim.split(paths, ",")

  local tags = vim.o.tags
```

```
  for _, path in ipairs(path_list) do
    tags = tags .. path .. "/tags,"
  end

  vim.o.tags = tags
end

vim.api.nvim_create_user_command('BuildTagsFromPath', build_tags_from_path,
↪  {})

-- Call the function when setting up
build_tags_from_path()
```

Source: vim.fandom.com

46.18 Easy Help Navigation on AZERTY Keyboards

Category: navigation
Tags: help, keyboard-mapping, key-navigation

Alternative key mapping for following help links on AZERTY keyboards, where Ctrl-] doesn't work

Example: Example (Vim)

```
" Remap help navigation for AZERTY keyboards
nmap <F9> <C-]>
map! <F9> <C-]>
```

Example: Example (Neovim)

```
-- Remap help navigation for AZERTY keyboards
vim.keymap.set('n', '<F9>', '<C-]>', { desc = 'Follow help link' })
vim.keymap.set({'i', 'c'}, '<F9>', '<C-]>', { desc = 'Follow help link' })
```

Source: vim.fandom.com

46.19 Easy Help Tag Navigation for German Keyboards

Category: navigation
Tags: key-mapping, help-navigation

Provide alternative key mappings to jump to help tags on German keyboards where ']' is difficult to access

Example: Example (Vim)

```
" Option 1: Map to 't' key
nnoremap t <C-]>

" Option 2: Map to umlaut key
nnoremap ü <C-]>
nnoremap Ü <C-O>
```

Example: Example (Neovim)

```
-- Option 1: Map to 't' key
vim.keymap.set('n', 't', '<C-]>', { desc = 'Jump to help tag' })

-- Option 2: Map to umlaut key
vim.keymap.set('n', 'ü', '<C-]>', { desc = 'Jump to help tag' })
vim.keymap.set('n', 'Ü', '<C-O>', { desc = 'Jump back from help tag' })
```

Source: vim.fandom.com_with_German_keyboard)

46.20 Efficient Cscope Navigation in Vim/Neovim

Category: navigation
Tags: code-navigation, c-development, project-management

Configure Vim/Neovim to use Cscope for advanced code navigation, allowing quick symbol, definition, and reference searches across large codebases

Example: Example (Vim)

```
if has('cscope')
  set cscopetag cscopeverbose

  if has('quickfix')
    set cscopequickfix=s-,c-,d-,i-,t-,e-
  endif

  cnoreabbrev csa cs add
  cnoreabbrev csf cs find
  cnoreabbrev csk cs kill
  cnoreabbrev csr cs reset
  cnoreabbrev css cs show
  cnoreabbrev csh cs help
endif
```

Example: Example (Neovim)

```
if vim.fn.has('cscope') == 1 then
  vim.opt.cscopetag = true
  vim.opt.cscopeverbose = true
```

```
  if vim.fn.has('quickfix') == 1 then
    vim.opt.cscopequickfix = 's-,c-,d-,i-,t-,e-'
  end

  -- Create commands/abbreviations using Neovim command creation
  vim.cmd[[
    cnoreabbrev csa cs add
    cnoreabbrev csf cs find
    cnoreabbrev csk cs kill
    cnoreabbrev csr cs reset
    cnoreabbrev css cs show
    cnoreabbrev csh cs help
  ]]
end
```

Source: vim.fandom.com

46.21 Efficient Tab Navigation Shortcuts

Category: navigation
Tags: key-mapping, tab-management, productivity

Create convenient keymappings for navigating and managing tabs using easy-to-remember shortcuts

Example: Example (Vim)

```
" Navigate tabs
nnoremap <C-S-tab> :tabprevious<CR>
nnoremap <C-tab>   :tabnext<CR>
nnoremap <C-t>     :tabnew<CR>

" Use H and L to cycle tabs
nnoremap H gT
nnoremap L gt
```

Example: Example (Neovim)

```
-- Tab navigation shortcuts
vim.keymap.set('n', '<C-S-Tab>', ':tabprevious<CR>', { noremap = true,
↪  silent = true })
vim.keymap.set('n', '<C-Tab>', ':tabnext<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<C-t>', ':tabnew<CR>', { noremap = true, silent = true
↪  })

-- Cycle tabs with H and L
vim.keymap.set('n', 'H', 'gT', { noremap = true })
vim.keymap.set('n', 'L', 'gt', { noremap = true })
```

Source: vim.fandom.com

46.22 Efficient Tab Navigation and Management

Category: navigation
Tags: tabs, key-mapping, window-management

Enhanced tab navigation and management with custom key mappings for moving between tabs and repositioning tabs

Example: Example (Vim)

```
" Navigate tabs
nnoremap <C-Left> :tabprevious<CR>
nnoremap <C-Right> :tabnext<CR>

" Move current tab
nnoremap <silent> <A-Left> :tabm -1<CR>
nnoremap <silent> <A-Right> :tabm +1<CR>
```

Example: Example (Neovim)

```
-- Navigate tabs
vim.keymap.set('n', '<C-Left>', ':tabprevious<CR>', { silent = true })
vim.keymap.set('n', '<C-Right>', ':tabnext<CR>', { silent = true })

-- Move current tab
vim.keymap.set('n', '<A-Left>', ':tabm -1<CR>', { silent = true })
vim.keymap.set('n', '<A-Right>', ':tabm +1<CR>', { silent = true })
```

Source: vim.fandom.com

46.23 Efficient Tab Page Navigation

Category: navigation
Tags: tab-management, key-mapping, window-management

Enhanced tab navigation with keyboard shortcuts for quickly moving between and managing tab pages

Example: Example (Vim)

```
" Tab navigation shortcuts
nnoremap <C-Left> :tabprevious<CR>
nnoremap <C-Right> :tabnext<CR>
nnoremap <silent> <A-Left> :tabm -1<CR>
nnoremap <silent> <A-Right> :tabm +1<CR>
```

Example: Example (Neovim)

```
-- Tab navigation shortcuts
vim.keymap.set('n', '<C-Left>', ':tabprevious<CR>', { noremap = true, silent
↪  = true })
vim.keymap.set('n', '<C-Right>', ':tabnext<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<A-Left>', ':tabm -1<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<A-Right>', ':tabm +1<CR>', { noremap = true, silent =
↪  true })
```

Source: vim.fandom.com

46.24 Efficient Tag Navigation in Vim/Neovim

Category: navigation
Tags: tags, code-navigation, jump-to-definition

Quickly navigate between code definitions using tags with multiple flexible methods

Example: Example (Vim)

```
" Jump to tag definition
Ctrl-]

" List multiple tag matches
g]

" Preview tag without leaving current window
Ctrl-W }
```

Example: Example (Neovim)

```
-- Jump to tag definition
vim.keymap.set('n', '<C-]>', '<cmd>tag<CR>', { desc = 'Jump to tag
↪  definition' })

-- List multiple tag matches
vim.keymap.set('n', 'g]', '<cmd>tselect<CR>', { desc = 'List tag matches' })

-- Preview tag in preview window
vim.keymap.set('n', '<C-W>}', '<cmd>ptag<CR>', { desc = 'Preview tag' })
```

Source: vim.fandom.com

46.25 Efficient Text Navigation Commands

Category: navigation
Tags: movement, text-objects, cursor-navigation

Powerful movement commands to quickly navigate through text without using arrow keys

Example: Example (Vim)

```
" Movement commands
" e - end of word
" w - start of next word
" b - back to start of word
" ) - next sentence
" } - next paragraph
```

Example: Example (Neovim)

```
-- Neovim movement commands are built-in
-- Use native Vim movement keys
-- Example: Add custom mapping for safer navigation
vim.keymap.set('n', '<leader>w', 'w', { desc = 'Move to next word start' })
vim.keymap.set('n', '<leader>e', 'e', { desc = 'Move to word end' })
```

Source: vim.fandom.com

46.26 Efficient Text Navigation and Manipulation

Category: navigation
Tags: movement, text-objects, editing

Powerful cursor movement and text editing commands that work across words, sentences, and paragraphs

Example: Example (Neovim)

```
-- Word movement
vim.keymap.set('n', 'w', 'w')  -- Move forward by word
vim.keymap.set('n', 'b', 'b')  -- Move backward by word
vim.keymap.set('n', 'e', 'e')  -- Move to end of word

-- Sentence and paragraph movement
vim.keymap.set('n', '(', '(')  -- Move to previous sentence
vim.keymap.set('n', ')', ')')  -- Move to next sentence
vim.keymap.set('n', '{', '{')  -- Move to previous paragraph
vim.keymap.set('n', '}', '}')  -- Move to next paragraph
```

Source: vim.fandom.com

46.27 Efficient Vim Navigation and Editing Basics

Category: navigation
Tags: beginner, key-mapping, text-editing

Key tips for new Vim users to get started with basic navigation and editing

Example: Example (Vim)

```
# Basic Vim navigation and editing
# Enter insert mode
i
# Exit and save
:w myfile.txt
# Exit without saving
:q
```

Example: Example (Neovim)

```
-- Lua equivalents (conceptual)
-- Enter insert mode
vim.cmd('startinsert')
-- Save file
vim.cmd('write myfile.txt')
-- Exit without saving
vim.cmd('quit')
```

Source: vim.fandom.com

46.28 Enable Caret Browsing for Vim-Like Text Navigation

Category: navigation
Tags: browser, keyboard-navigation

Firefox's caret browsing mode allows keyboard-based text cursor movement similar to Vim's navigation

Example: Example (Neovim)

```
-- In Firefox, press F7 to toggle caret browsing
-- Enables moving through text with keyboard like in Vim
```

Source: vim.fandom.com

46.29 Enhanced Bracket and Brace Navigation

Category: navigation
Tags: movement, text-objects, matching

Use % key to jump between matching braces, brackets, and more advanced language constructs

Example: Example (Vim)

```
" Enable extended matching with matchit.vim
runtime macros/matchit.vim
```

Example: Example (Neovim)

```
-- For Neovim, you can use built-in matchit or a Lua equivalent plugin
-- Add matchit.vim or use a Lua-based matching plugin
```

Source: vim.fandom.com

46.30 Enhanced Matching with % Key

Category: navigation
Tags: motion, text-objects, programming

Expand the % key functionality to jump between matching elements beyond just brackets, including code blocks, comments, and preprocessor conditionals

Example: Example (Vim)

```
" Enable matchit plugin
runtime macros/matchit.vim
```

Example: Example (Neovim)

```
-- In Neovim, can use built-in matchit or treesitter for enhanced matching
-- Recommend installing nvim-treesitter for advanced language-aware matching
require('nvim-treesitter.configs').setup {
  matchup = {
    enable = true,
  }
}
```

Source: vim.fandom.com

46.31 Enhanced Word Search with Smart Matching

Category: navigation
Tags: search, word-navigation, mapping

Custom mappings for word search that respect smartcase and whole word boundaries

Example: Example (Vim)

```
nnoremap * /\<<C-R>=expand('<cword>')\>\><CR>
nnoremap # ?\<<C-R>=expand('<cword>')\>\><CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '*', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.search('\<' .. word .. '\>')
end)

vim.keymap.set('n', '#', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.search('\<' .. word .. '\>', 'b')
end)
```

Source: vim.fandom.com

46.32 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: Example (Neovim)

```
-- 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
```

Source: Community contributed

46.33 Fast Jump to PHP Variables on a Line

Category: navigation
Tags: php, movement, key-mapping

Quickly jump between PHP variables on the same line using custom mappings

Example: Example (Vim)

```
noremap L f$
noremap H F$
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'L', 'f$', { desc = 'Jump to next PHP variable' })
vim.keymap.set('n', 'H', 'F$', { desc = 'Jump to previous PHP variable' })
```

Source: vim.fandom.com

46.34 Fast Window Scrolling with Alt Key Mappings

Category: navigation
Tags: key-mapping, scrolling, window-management

Quick mappings to scroll windows horizontally and vertically using Alt key combinations

Example: Example (Vim)

```
" Horizontal and vertical window scrolling
nmap <A-l> 4zl
nmap <A-h> 4zh
nmap <A-k> <C-y>
nmap <A-j> <C-e>
```

Example: Example (Neovim)

```
-- Scroll window horizontally and vertically
vim.keymap.set('n', '<A-l>', '4zl', { desc = 'Scroll window right' })
vim.keymap.set('n', '<A-h>', '4zh', { desc = 'Scroll window left' })
vim.keymap.set('n', '<A-k>', '<C-y>', { desc = 'Scroll window up' })
vim.keymap.set('n', '<A-j>', '<C-e>', { desc = 'Scroll window down' })
```

Source: vim.fandom.com

46.35 Fast buffer access

Category: Navigation
Tags: buffer, fast, access, number

Create mappings to quickly access first nine buffers using leader key combinations.

Example: Example (Vim)

```
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
```

Source: Community contributed

46.36 Find Current Verilog Module Name

Category: navigation
Tags: search, module-navigation, verilog

Quickly identify the current Verilog module by searching backwards and echoing the module name

Example: Example (Vim)

```
map ` ma?module<CR>Wyiw'a:echo "module -->" @0<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '`', function()
  -- Save current position
  vim.cmd('normal! ma')

  -- Search backwards for module definition
  vim.cmd('?module')

  -- Yank module name and return to mark
  vim.cmd('normal! Wyiw`a')

  -- Echo module name
  local module_name = vim.fn.getreg('0')
  print("module --> " .. module_name)
end, { desc = 'Find current Verilog module' })
```

Source: vim.fandom.com

46.37 Flexible File Navigation

Category: navigation
Tags: file-navigation, goto-file, path-searching

Enhance 'goto file' command with configurable file name and path detection

Example: Example (Vim)

```
set path+=**
set isfname+=32
set suffixesadd+=.js,.py
```

Example: Example (Neovim)

```
vim.opt.path:append('**')
vim.opt.isfname:append(32)
vim.opt.suffixesadd:append({'.js', '.py'})
```

Source: vim.fandom.com

46.38 Flexible Python Test File Toggle

Category: navigation
Tags: python, file-navigation, buffer-management

Toggle between source and test files with a smart mechanism that checks existing buffers and handles different file naming patterns

Example: Example (Vim)

```
function! UTestToggle()
python << EOF
import vim
from os import path

UTEST_PREFIX='test/utest_'
curfile = vim.current.buffer.name
if curfile:
  if UTEST_PREFIX in curfile:
    vim.command('e %s' % curfile.replace(UTEST_PREFIX, ''))
  else:
    filepath, filename = path.split(curfile)
    vim.command('e %s' % path.join(filepath, UTEST_PREFIX+filename))
EOF
endfunction

nnoremap <M-t> :call UTestToggle()<CR>
```

Example: Example (Neovim)

```
function UTestToggle()
  local UTEST_PREFIX = 'test/utest_'
  local curfile = vim.fn.expand('%:p')

  if curfile:match(UTEST_PREFIX) then
    local source_file = curfile:gsub(UTEST_PREFIX, '')
    vim.cmd('edit ' .. source_file)
  else
    local filepath = vim.fn.expand('%:p:h')
    local filename = vim.fn.expand('%:t')
    local test_file = filepath .. '/' .. UTEST_PREFIX .. filename
    vim.cmd('edit ' .. test_file)
  end
end

vim.keymap.set('n', '<M-t>', UTestToggle, { desc = 'Toggle between source
↪  and test files' })
```

Source: vim.fandom.com

46.39 Flexible Tag Search and Navigation

Category: navigation
Tags: tag-search, code-exploration, fuzzy-finding

Advanced tag searching with pattern matching and browsing multiple matches

Example: Example (Vim)

```
" Search tags with a pattern
:tag /<pattern>

" Jump to tag only if single match, otherwise list
:tjump <tagname>
```

Example: Example (Neovim)

```
-- Search tags with a pattern
vim.cmd.tag('/' .. pattern)

-- Conditional tag jumping
vim.keymap.set('n', '<leader>tj', function()
  vim.cmd('tjump ' .. vim.fn.expand('<cword>'))
end, { desc = 'Conditional tag jump' })
```

Source: vim.fandom.com

46.40 Function Definition Search

Category: navigation
Tags: code-search, programming, function-navigation

Custom function to search for function definitions more precisely

Example: Example (Vim)

```
function! GotoDefinition()
  let n = search("\<".expand("<cword>")."\>[^(]*([^)]*)\s*\n*\s*{")
endfunction
map <F4> :call GotoDefinition()<CR>
imap <F4> <c-o>:call GotoDefinition()<CR>
```

Example: Example (Neovim)

```
function _G.goto_definition()
  vim.fn.search(vim.fn.expand('<cword>') .. '[^(]*([^)]*)\s*\n*\s*{')
end

vim.keymap.set('n', '<F4>', _G.goto_definition, { desc = 'Go to function
↪  definition' })
vim.keymap.set('i', '<F4>', '<c-o>:lua _G.goto_definition()<CR>', { desc =
↪  'Go to function definition' })
```

Source: vim.fandom.com

46.41 Fuzzy Search Multiple Vim Contexts

Category: navigation
Tags: fuzzy-search, fzf, productivity

Create powerful fuzzy search mappings for various Vim contexts like buffers, commands, mappings, and more using FZF

Example: Example (Vim)

```
" Fuzzy search mappings
nnoremap <silent> <C-a>b :call FZFOpen(':Buffers')<CR>
nnoremap <silent> <C-a>c :call fzf#run({'source': GetCommands(), 'sink':
↪  function('HandleCommand'), 'options': '-m'})<CR>
nnoremap <silent> <C-a>m :call fzf#run({'source': GetMappings(), 'options':
↪  '-m -n 2'})<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent requires fzf.vim plugin setup
local fzf = require('fzf-lua')

vim.keymap.set('n', '<C-a>b', function()
  fzf.buffers()
end, { desc = 'Fuzzy search buffers' })

vim.keymap.set('n', '<C-a>c', function()
  fzf.command_history()
end, { desc = 'Fuzzy search command history' })

vim.keymap.set('n', '<C-a>m', function()
  fzf.keymaps()
end, { desc = 'Fuzzy search key mappings' })
```

Source: vim.fandom.com

46.42 Fuzzy Search Tags with Regular Expressions

Category: navigation
Tags: tags, search, regular-expressions

Use regular expressions to flexibly search and find tags when you're not sure of the exact function name

Example: Example (Vim)

```
:tj /handle.*event<C-D>
```

Example: Example (Neovim)

```
-- Lua equivalent for tag searching
-- Note: Requires having tags file generated
vim.cmd('tj /handle.*event')
-- Use <C-D> for command-line completion
```

Source: vim.fandom.com

46.43 Generate Tags for Current File

Category: navigation
Tags: ctags, code-navigation, file-indexing

Use ctags to generate and display tags for the current file, helping quickly navigate function and variable definitions

Example: Example (Vim)

```
map <M-l> <Esc>:!/usr/local/bin/vimlocals.sh % <CR>:
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-l>', function()
  local current_file = vim.fn.expand('%')
  vim.cmd('!ctags -x -n ' .. current_file)
end, { desc = 'Show tags for current file' })
```

Source: vim.fandom.com

46.44 Generate and Use Custom Tag Files

Category: navigation
Tags: file-navigation, tags, project-management

Generate a custom tags file for quick navigation between files, especially useful for large projects with many files of the same type

Example: Example (Vim)

```
# Generate tags file for .jsp files
# Terminal command:
find -name '*.jsp' -printf '%f\t%P\t1\n' |sort > jsp.tags

# In Vim:
:set tags+=jsp.tags
:tag file.jsp  # Quick file navigation
```

Example: Example (Neovim)

```
-- Lua doesn't change the core functionality, but you can use vim.fn to run
↪  shell commands
-- Generate tags file
vim.fn.system('find -name "*.jsp" -printf "%f\t%P\t1\n" | sort > jsp.tags')

-- Add tags file
vim.opt.tags:append('jsp.tags')

-- Use :tag as before
```

Source: vim.fandom.com

46.45 Go to declaration

Category: Navigation
Tags: lsp, declaration, goto

Use `gD` to go to declaration of symbol under cursor.

Example: Example (Vim)

```
gD  " go to declaration
```

Source: Community contributed

46.46 Highlight and Navigate CSV Columns Easily

Category: navigation
Tags: csv, data-parsing, text-manipulation

Provides functions to highlight specific columns in CSV files and navigate between columns with easy commands

Example: Example (Vim)

```
function! CSVH(colnr)
  if a:colnr > 1
    let n = a:colnr - 1
    execute 'match Keyword /^([^,]*,)\{'.n.'}\zs[^,]*/'
    execute 'normal! 0'.n.'f,'
  elseif a:colnr == 1
    match Keyword /^[^,]*/
    normal! 0
  else
    match
  endif
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
```

Example: Example (Neovim)

```
local function csv_highlight(colnr)
  if colnr > 1 then
    local n = colnr - 1
    vim.cmd('match Keyword /^([^,]*,){' .. n .. '}\\zs[^,]*/')
    vim.cmd('normal! 0' .. n .. 'f,')
  elseif colnr == 1 then
    vim.cmd('match Keyword /^[^,]*/')
    vim.cmd('normal! 0')
  else
    vim.cmd('match')
  end
end

vim.api.nvim_create_user_command('Csv', function(opts)
  csv_highlight(tonumber(opts.args))
end, { nargs = 1 })
```

Source: vim.fandom.com

46.47 Highlight and Navigate CSV Columns Efficiently

Category: navigation
Tags: csv, text-processing, highlighting

Provides advanced navigation and highlighting for CSV files, allowing easy column selection, searching, and manipulation

Example: Example (Vim)

```
function! CSVH(colnr)
  if a:colnr > 1
    let n = a:colnr - 1
    execute 'match Keyword /^([^,]*,)\{'.n.'}\zs[^,]*/')
    execute 'normal! 0'.n.'f,'
  elseif a:colnr == 1
    match Keyword /^[^,]*/
    normal! 0
  else
    match
  endif
endfunction
command! -nargs=1 Csv :call CSVH(<args>)
```

Example: Example (Neovim)

```
function _G.highlight_csv_column(colnr)
  if colnr > 1 then
    local n = colnr - 1
    vim.cmd(string.format('match Keyword /^([^,]*,)\\{%d}\\zs[^,]*/', n))
    vim.cmd(string.format('normal! 0%df,', n))
  elseif colnr == 1 then
    vim.cmd('match Keyword /^[^,]*/')
    vim.cmd('normal! 0')
  else
    vim.cmd('match')
  end
end

vim.api.nvim_create_user_command('Csv', function(opts)
  _G.highlight_csv_column(tonumber(opts.args))
end, { nargs = 1 })
```

Source: vim.fandom.com

46.48 Improve Last Position Jump for Easy Vim

Category: navigation
Tags: cursor-position, vim-improvement, last-jump

Enhance the last position jump functionality in Easy Vim to restore cursor position, even

beyond the end of line (EOL)

Example: Example (Neovim)

```
-- For Neovim, this can be achieved with an autocmd
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    local last_pos = vim.fn.line("'\"") -- Get last known position
    if last_pos > 0 and last_pos <= vim.fn.line('$') then
      vim.cmd('normal! g`"')
    end
  end
})
```

Source: vim.fandom.com

46.49 Inline Java Documentation Browsing

Category: navigation
Tags: documentation, java, productivity

Quickly view Java class and method documentation without leaving Vim, using generated help files

Example: Example (Vim)

```
" Use :help to access Java docs
:help String
:help java.util.List
```

Example: Example (Neovim)

```
-- Similar navigation in Neovim
-- After setting up doclet help files
vim.cmd('help String')
vim.cmd('help java.util.List')
```

Source: vim.fandom.com

46.50 Interactive Jump List Selection

Category: navigation
Tags: jump-list, custom-function, interactive

Create a custom function to interactively select and navigate jump list locations

Example: Example (Vim)

```
function! GotoJump()
  jumps
  let j = input('Please select your jump: ')
  if j != ''
    let pattern = '\v\c^\+'
    if j =~ pattern
      let j = substitute(j, pattern, '', 'g')
      execute "normal " . j . "\<c-i>"
    else
      execute "normal " . j . "\<c-o>"
    endif
  endif
endfunction

" Optional mapping
nmap <Leader>j :call GotoJump()<CR>
```

Example: Example (Neovim)

```
function _G.goto_jump()
  vim.cmd('jumps')
  local jump = vim.fn.input('Please select your jump: ')
  if jump ~= '' then
    local pattern = '^+'
    if jump:match(pattern) then
      jump = jump:gsub(pattern, '')
      vim.api.nvim_feedkeys(jump .. '\<C-i>', 'n', true)
    else
      vim.api.nvim_feedkeys(jump .. '\<C-o>', 'n', true)
    end
  end
end

-- Optional mapping
vim.keymap.set('n', '<Leader>j', _G.goto_jump, { desc = 'Interactive jump
↪  list navigation' })
```

Source: vim.fandom.com

46.51 Jump Between C++ Methods Quickly

Category: navigation
Tags: c++, search, key-mapping

Easily navigate between methods in C++ files using Alt-Shift-J/K key mappings with a regex-based search

Example: Example (Vim)

```
nnoremap <M-J> /\v^(\w+\s+)?\w+::\w+\(.*\)
nnoremap <M-K> ?\v^(\w+\s+)?\w+::\w+\(.*\)
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-J>', function()
  vim.fn.search('\v^(\w+\s+)?\w+::\w+\(.*\)')
end, { desc = 'Jump to next C++ method' })

vim.keymap.set('n', '<M-K>', function()
  vim.fn.search('\v^(\w+\s+)?\w+::\w+\(.*\)', 'b')
end, { desc = 'Jump to previous C++ method' })
```

Source: vim.fandom.com

46.52 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: Example (Vim)

```
]m  " next function start
[m  " previous function start
```

Source: Community contributed

46.53 Jump to File from Diff View

Category: navigation
Tags: diff, file-navigation, key-mapping

Adds a mapping to jump from a diff view to the original file at the corresponding line, making code review and comparison easier

Example: Example (Vim)

```
function! DiffJumpToFile()
  let a=line(".")
  let b=search("^\(---\|\*\*\*\) ", "b")
  let c=getline(b)
  let d=strpart(c, 4, match(c, ",")-4)
  let f=search("^\(---\|\*\*\*\) .*\t", "b")
  let g=getline(f)
  let h=match(g, "\t", 4)
```

```
  let i=strpart(g, 4, h-4)
  execute ":b " . i
  execute "normal " . (d+a-b-1) . "G"
endfunction
nmap <buffer> <CR> :call DiffJumpToFile()<CR>
```

Example: Example (Neovim)

```
function _G.diff_jump_to_file()
  local current_line = vim.fn.line('.')
  local header_line = vim.fn.search('^(---|\*\*\*) ', 'b')
  local header_text = vim.fn.getline(header_line)
  local first_line_num = tonumber(header_text:match('(%d+),'))
  local file_header_line = vim.fn.search('^(---|\*\*\*) .*        ', 'b')
  local file_header_text = vim.fn.getline(file_header_line)
  local filename = file_header_text:match('(.-)        ')

  vim.cmd('buffer ' .. filename)
  vim.cmd(tostring(first_line_num + current_line - header_line - 1) .. 'G')
end

vim.api.nvim_create_autocmd('FileType', {
  pattern = 'diff',
  callback = function()
    vim.keymap.set('n', '<CR>', _G.diff_jump_to_file, { buffer = true })
  end
})
```

Source: vim.fandom.com

46.54 Jump to Lines with Same/Lower Indentation

Category: navigation
Tags: indentation, movement, programming

Quickly navigate between lines with the same or lower indentation level, which is especially useful in languages like Python, Haskell, or when editing XML

Example: Example (Vim)

```
function! NextIndent(exclusive, fwd, lowerlevel, skipblanks)
  let line = line('.')
  let column = col('.')
  let lastline = line('$')
  let indent = indent(line)
  let stepvalue = a:fwd ? 1 : -1
  while (line > 0 && line <= lastline)
    let line = line + stepvalue
    if ( ! a:lowerlevel && indent(line) == indent ||
          \ a:lowerlevel && indent(line) < indent)
      if (! a:skipblanks || strlen(getline(line)) > 0)
```

```
        if (a:exclusive)
          let line = line - stepvalue
        endif
        exe line
        exe "normal " column . "|"
        return
      endif
    endif
  endwhile
endfunction

" Normal mode mappings
nnoremap <silent> [l :call NextIndent(0, 0, 0, 1)<CR>
nnoremap <silent> ]l :call NextIndent(0, 1, 0, 1)<CR>
nnoremap <silent> [L :call NextIndent(0, 0, 1, 1)<CR>
nnoremap <silent> ]L :call NextIndent(0, 1, 1, 1)<CR>
```

Example: Example (Neovim)

```
function _G.next_indent(exclusive, fwd, lowerlevel, skipblanks)
  local line = vim.fn.line('.')
  local column = vim.fn.col('.')
  local lastline = vim.fn.line('$')
  local indent = vim.fn.indent(line)
  local stepvalue = fwd and 1 or -1

  while line > 0 and line <= lastline do
    line = line + stepvalue
    local current_indent = vim.fn.indent(line)
    local line_content = vim.fn.getline(line)

    if (not lowerlevel and current_indent == indent) or
       (lowerlevel and current_indent < indent) then
      if not skipblanks or #line_content > 0 then
        if exclusive then
          line = line - stepvalue
        end
        vim.fn.cursor(line, column)
        return
      end
    end
  end
end

-- Set up keymappings
vim.keymap.set('n', '[l', function() _G.next_indent(false, false, false,
 ↪  true) end, { silent = true })
vim.keymap.set('n', ']l', function() _G.next_indent(false, true, false,
 ↪  true) end, { silent = true })
vim.keymap.set('n', '[L', function() _G.next_indent(false, false, true,
 ↪  true) end, { silent = true })
vim.keymap.set('n', ']L', function() _G.next_indent(false, true, true, true)
 ↪  end, { silent = true })
```

Source: vim.fandom.com

46.55 Jump to Local and Global Definitions Easily

Category: navigation
Tags: definition-jumping, code-navigation, key-mapping

Combines local and global variable/function definition jumping into a single key mapping, first trying local search, then falling back to global tags

Example: Example (Vim)

```
function! GoDefinition()
  let l:pos = getpos('.')
  normal! gd
  if getpos('.') == l:pos
    execute 'tag' expand('<cword>')
  endif
endfunction
nnoremap <C-]> :<C-U>call GoDefinition()<CR>
```

Example: Example (Neovim)

```
function _G.go_definition()
  local original_pos = vim.fn.getpos('.')
  vim.cmd('normal! gd')
  local new_pos = vim.fn.getpos('.')

  if vim.fn.list2str(original_pos) == vim.fn.list2str(new_pos) then
    local word = vim.fn.expand('<cword>')
    vim.cmd('tag ' .. word)
  end
end

vim.keymap.set('n', '<C-]>', _G.go_definition, { desc = 'Jump to local or
↪  global definition' })
```

Source: vim.fandom.com

46.56 Jump to Matching Braces and Code Blocks

Category: navigation
Tags: movement, code-navigation, editing

Use the % key to quickly jump between matching braces, brackets, and other code block delimiters across multiple programming languages

Example: Example (Vim)

```
" Default behavior
" % jumps to matching brace/bracket/parenthesis
set matchtime=3
set showmatch
```

Example: Example (Neovim)

```
-- Configure matching brace highlighting
vim.opt.matchtime = 3
vim.opt.showmatch = true
```

Source: vim.fandom.com

46.57 Jump to Matching Braces and Syntax Elements

Category: navigation
Tags: movement, syntax-navigation, text-objects

Use % key to jump between matching syntax elements, including braces, comments, and language-specific constructs

Example: Example (Vim)

```
" Built-in % navigation
" Works with (), [], {}, comments, preprocessor blocks
```

Example: Example (Neovim)

```
-- Recommend using built-in % key
-- Can be enhanced with matchit plugin
-- No specific Lua configuration needed
```

Source: vim.fandom.com

46.58 Jump to PDF Page from LaTeX File

Category: navigation
Tags: latex, pdf, workflow, file-navigation

Automatically open a PDF viewer to the corresponding page of the current LaTeX file, making document navigation easier

Example: Example (Vim)

```
function! EvinceNearestLabel()
  let line = search("\\label{", "bnW")
  if line > 0
    let m = matchstr(getline(line), '\\label{\zs[^}]*\ze}')
    if !empty(m)
      call LoadEvinceByLabel(m)
    endif
  endif
endfunction

nnoremap <buffer> <LocalLeader>e :call EvinceNearestLabel()<CR>
```

Example: Example (Neovim)

```
function _G.nearest_label_pdf()
  local line_nr = vim.fn.search('\\label{', 'bnW')
  if line_nr > 0 then
    local line = vim.fn.getline(line_nr)
    local label = line:match('\\label{([^}]+)}')
    if label and label ~= '' then
      vim.fn.LoadEvinceByLabel(label)
    end
  end
end

vim.keymap.set('n', '<LocalLeader>e', _G.nearest_label_pdf, { buffer = true
↪  })
```

Source: vim.fandom.com

46.59 Jump to Random Line in Buffer

Category: navigation
Tags: random, line-jumping, utility

Create a command to jump to a random line in the current buffer, useful for testing or exploring large files

Example: Example (Vim)

```
function! RandomLine() range
  let first_line = a:firstline
  let last_line = a:lastline
  execute 'normal! '.(system('od -vAn -N3 -tu4 /dev/urandom') % (last_line -
  ↪  first_line + 1) + first_line).'G'
endfunction
command! -range=% RandomLine <line1>,<line2>call RandomLine()
```

Example: Example (Neovim)

```
function _G.random_line(first_line, last_line)
  local total_lines = last_line - first_line + 1
  local random_line = first_line + math.random(total_lines - 1)
  vim.api.nvim_win_set_cursor(0, {random_line, 0})
end

vim.api.nvim_create_user_command('RandomLine', function(opts)
  local first_line = opts.line1
  local last_line = opts.line2
  _G.random_line(first_line, last_line)
end, {range = true})
```

Source: vim.fandom.com

46.60 Jump to Variable Declaration

Category: navigation
Tags: code-navigation, programming, search

Quickly jump to local or global variable declarations in source code

Example: Example (Vim)

```
" Jump to local variable declaration
gd  " Normal mode command

" Jump to global variable declaration
gD  " Normal mode command
```

Example: Example (Neovim)

```
-- These Vim commands work natively in Neovim
-- No specific Lua conversion needed
-- Use gd and gD in normal mode
```

Source: vim.fandom.com

46.61 Jump to a Random Line in Buffer

Category: navigation
Tags: random, jumping, buffer-navigation

Provides multiple methods to jump to a random line in the current buffer using different random number generation techniques

Example: Example (Vim)

```
command! RandomLine execute 'normal! '.(system('/bin/bash -c "echo -n
↪  $RANDOM"') % line('$')).'G'
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('RandomLine', function()
  local total_lines = vim.fn.line('$')
  local random_line = math.random(total_lines)
  vim.cmd(string.format('normal! %dG', random_line))
end, {})
```

Source: vim.fandom.com

46.62 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: Example (Vim)

```
[{  " jump to block start
]}  " jump to block end
```

Source: Community contributed

46.63 Keyboard-Driven Web Browsers with Vim Keybindings

Category: navigation
Tags: browser, keyboard-navigation, productivity

Some specialized web browsers natively support Vim-like keyboard navigation

Example: Example (Neovim)

```
-- Recommended Vim-like browsers:
-- Qutebrowser (keyboard-driven)
-- Vimb (WebKit + GTK+)
-- Vieb (Electron-based)
-- Luakit (Lua + WebKit)
```

Source: vim.fandom.com

46.64 LSP go to references

Category: Navigation
Tags: lsp, references, goto

Use `gr` to go to references of symbol under cursor (requires LSP server).

Example: Example (Vim)

```
gr  " go to references
```

Source: Community contributed

46.65 Master Vim Movement Shortcuts

Category: navigation
Tags: movement, cursor-navigation, efficiency

Comprehensive collection of efficient cursor movement commands in Vim/Neovim, enabling faster text navigation

Example: Example (Neovim)

```
-- Movement shortcuts
-- Basic navigation
vim.keymap.set('n', 'w', 'w', { desc = 'Move to next word' })
vim.keymap.set('n', 'b', 'b', { desc = 'Move to previous word' })
vim.keymap.set('n', '0', '0', { desc = 'Move to start of line' })
vim.keymap.set('n', '$', '$', { desc = 'Move to end of line' })

-- Screen-based navigation
vim.keymap.set('n', 'H', 'H', { desc = 'Move to top of screen' })
vim.keymap.set('n', 'M', 'M', { desc = 'Move to middle of screen' })
vim.keymap.set('n', 'L', 'L', { desc = 'Move to bottom of screen' })
```

Source: vim.fandom.com

46.66 Move Between Words in Insert Mode

Category: navigation
Tags: insert-mode, word-movement, key-mapping

Quick navigation between words while in insert mode using keyboard shortcuts

Example: Example (Vim)

```
" Built-in shortcuts
" Shift + Right Arrow: Move to next word
" Shift + Left Arrow: Move to previous word
```

```
" Ctrl + Right/Left Arrow also work across different vim behaviors
```

Example: Example (Neovim)

```
-- These are built-in Neovim behaviors
-- No specific Lua configuration needed
-- Equivalent to Vim's native word movement in insert mode
```

Source: vim.fandom.com

46.67 Move Cursor by Display Lines in Wrapped Text

Category: navigation
Tags: cursor-movement, text-wrapping, key-mapping

Remap navigation keys to move by displayed lines instead of physical lines when text wrapping is enabled, improving readability and navigation in wrapped text

Example: Example (Vim)

```
" Remap j, k, and arrow keys to move by display lines
noremap <silent> k gk
noremap <silent> j gj
noremap <silent> 0 g0
noremap <silent> $ g$

" Support for operator-pending mode
onoremap <silent> j gj
onoremap <silent> k gk
```

Example: Example (Neovim)

```
-- Remap movement keys to navigate by display lines
vim.keymap.set('n', 'k', 'gk', { silent = true })
vim.keymap.set('n', 'j', 'gj', { silent = true })
vim.keymap.set('n', '0', 'g0', { silent = true })
vim.keymap.set('n', '$', 'g$', { silent = true })

-- Support for operator-pending mode
vim.keymap.set('o', 'j', 'gj', { silent = true })
vim.keymap.set('o', 'k', 'gk', { silent = true })
```

Source: vim.fandom.com

46.68 Move Word Under Cursor with Alt-Arrows

Category: navigation
Tags: text-movement, word-editing, key-mapping

Move the word under the cursor in different directions without explicitly selecting it first

Example: Example (Vim)

```
nnoremap <A-Right> viwm`w``xwzvP`[1v<Space>
nnoremap <A-Left> viwm`bb``xbzvP`[1v<Space>
nnoremap <A-Down> viwjkxjzvP`[1v<Space>
nnoremap <A-Up> viwkjxkzvP`[1v<Space>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<A-Right>', 'viwm`w``xwzvP`[1v<Space>', { noremap =
↪  true })
vim.keymap.set('n', '<A-Left>', 'viwm`bb``xbzvP`[1v<Space>', { noremap =
↪  true })
vim.keymap.set('n', '<A-Down>', 'viwjkxjzvP`[1v<Space>', { noremap = true })
vim.keymap.set('n', '<A-Up>', 'viwkjxkzvP`[1v<Space>', { noremap = true })
```

Source: vim.fandom.com

46.69 Multiple Ways to Navigate to a Specific Line

Category: navigation
Tags: line-navigation, cursor-movement

Multiple methods to quickly jump to a specific line number in Vim/Neovim

Example: Example (Vim)

```
42G
42gg
:42<CR>
exe 42
```

Example: Example (Neovim)

```
-- Jump to line 42
vim.cmd('42')

-- Alternative methods
vim.cmd('normal! 42G')
vim.cmd('normal! 42gg')
```

Source: vim.fandom.com

46.70 Navigate File Change History

Category: navigation
Tags: change-list, navigation, editing-history

Easily jump between previous change locations in a file using built-in change list navigation

Example: Example (Vim)

```
" Jump to previous change
g;

" Jump to next change
g,

" View change list
:changes
```

Example: Example (Neovim)

```
-- Jump to previous change
vim.keymap.set('n', 'g;', 'g;', { desc = 'Go to previous change' })

-- Jump to next change
vim.keymap.set('n', 'g,', 'g,', { desc = 'Go to next change' })

-- View change list
vim.api.nvim_command('changes')
```

Source: vim.fandom.com

46.71 Navigate Help Labels with Tab Keys

Category: navigation
Tags: help-buffer, key-mapping, search

Add Tab/Shift-Tab mappings to quickly jump between |labels| in help documentation

Example: Example (Vim)

```
" Navigate between help labels
nnoremap <buffer> <Tab> /<Bar>\zs\k*\ze<Bar><CR>
nnoremap <buffer> <S-Tab> ?<Bar>\zs\k*\ze<Bar><CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for navigating help labels
vim.api.nvim_create_autocmd('FileType', {
```

```
    pattern = 'help',
    callback = function()
      vim.keymap.set('n', '<Tab>', '/<Bar>\zs\k*\ze<Bar><CR>', { buffer = true
      ↪  })
      vim.keymap.set('n', '<S-Tab>', '?<Bar>\zs\k*\ze<Bar><CR>', { buffer =
      ↪  true })
    end
  })
```

Source: vim.fandom.com

46.72 Navigate Jump List Efficiently

Category: navigation
Tags: jump-list, navigation, cursor-movement

Quickly move between previous and next jump locations in Vim/Neovim, similar to browser navigation

Example: Example (Vim)

```
" Jump navigation
noremap <C-o> Jump back to previous location
noremap <C-i> Jump forward to next location
:jumps  " Display jump list
```

Example: Example (Neovim)

```
-- Jump navigation
vim.keymap.set('n', '<C-o>', '<C-o>', { desc = 'Jump back to previous
↪  location' })
vim.keymap.set('n', '<C-i>', '<C-i>', { desc = 'Jump forward to next
↪  location' })
-- Display jump list
vim.api.nvim_command('jumps')
```

Source: vim.fandom.com

46.73 Navigate Jump List with Keyboard Shortcuts

Category: navigation
Tags: jump-list, navigation, motion

Quickly move between previously visited locations in Vim/Neovim using built-in jump list navigation shortcuts

Example: Example (Vim)

```
" Navigate jump list
Ctrl-O  " Jump back to previous location
Ctrl-I  " Jump forward to newer location
:jumps  " Display jump list
```

Example: Example (Neovim)

```
-- Jump list navigation can be used directly in Neovim
-- Ctrl-O and Ctrl-I work the same way
-- Use :jumps to view jump history
vim.keymap.set('n', '<C-o>', '<C-o>', { desc = 'Jump back in jump list' })
vim.keymap.set('n', '<C-i>', '<C-i>', { desc = 'Jump forward in jump list'
↪  })
```

Source: vim.fandom.com

46.74 Navigate LaTeX Sections Efficiently

Category: navigation
Tags: latex, section-jumping, text-objects

Custom mapping to jump between LaTeX sections (part, chapter, section) with support for forward/backward movement and count repetitions

Example: Example (Vim)

```
" section jumping
noremap <buffer> <silent> ]] :<c-u>call TexJump2Section( v:count1, '' )<CR>
noremap <buffer> <silent> [[ :<c-u>call TexJump2Section( v:count1, 'b' )<CR>

function! TexJump2Section( cnt, dir )
  let i = 0
  let pat =
  ↪  '^\s*\\(part\|chapter\|\(sub\)*section\|paragraph\)\>\|\%$\|\%^'
  let flags = 'W' . a:dir
  while i < a:cnt && search( pat, flags ) > 0
    let i = i+1
  endwhile
  let @/ = pat
endfunction
```

Example: Example (Neovim)

```
local function tex_jump_to_section(count, direction)
  local pattern =
  ↪  '^\s*\\(part\|chapter\|(sub)*section\|paragraph)\>\|\%$\|\%^'
  local flags = 'W' .. (direction or '')
```

```
  for _ = 1, count do
    vim.fn.search(pattern, flags)
  end

  vim.fn.setreg('/', pattern)
end

vim.keymap.set('n', ']]', function()
  tex_jump_to_section(vim.v.count1, '')
end, { buffer = true, silent = true })

vim.keymap.set('n', '[[', function()
  tex_jump_to_section(vim.v.count1, 'b')
end, { buffer = true, silent = true })
```

Source: vim.fandom.com

46.75 Navigate Lines with Same Indentation

Category: navigation
Tags: indentation, movement, code-navigation

Quickly jump between lines with the same or lower indentation level, useful in Python, XML, and other indentation-sensitive languages

Example: Example (Vim)

```
" Jump between lines with same or lower indentation
nnoremap <silent> [l :call NextIndent(0, 0, 0, 1)<CR>
nnoremap <silent> ]l :call NextIndent(0, 1, 0, 1)<CR>
nnoremap <silent> [L :call NextIndent(0, 0, 1, 1)<CR>
nnoremap <silent> ]L :call NextIndent(0, 1, 1, 1)<CR>
```

Example: Example (Neovim)

```
-- Create a Lua function to replicate NextIndent behavior
local function next_indent(exclusive, fwd, lower_level, skip_blanks)
  local line = vim.fn.line('.')
  local column = vim.fn.col('.')
  local last_line = vim.fn.line('$')
  local indent = vim.fn.indent(line)
  local step = fwd and 1 or -1

  while line > 0 and line <= last_line do
    line = line + step
    local current_indent = vim.fn.indent(line)
    local line_not_blank = vim.fn.strlen(vim.fn.getline(line)) > 0

    if (not lower_level and current_indent == indent) or
       (lower_level and current_indent < indent) then
      if not skip_blanks or line_not_blank then
```

```
        if exclusive then
          line = line - step
        end
        vim.cmd(tostring(line))
        vim.cmd('normal! ' .. column .. '|')
        return
      end
    end
  end
end

-- Set up keymappings in Neovim
vim.keymap.set('n', '[l', function() next_indent(false, false, false, true)
 ↪  end, { silent = true })
vim.keymap.set('n', ']l', function() next_indent(false, true, false, true)
 ↪  end, { silent = true })
vim.keymap.set('n', '[L', function() next_indent(false, false, true, true)
 ↪  end, { silent = true })
vim.keymap.set('n', ']L', function() next_indent(false, true, true, true)
 ↪  end, { silent = true })
```

Source: vim.fandom.com

46.76 Navigate Vim Help on AZERTY Keyboards

Category: navigation
Tags: help, keyboard-mapping, azerty

Alternative key mapping for navigating Vim help on AZERTY keyboards where Ctrl+] doesn't work

Example: Example (Vim)

```
" Map F9 to follow help tags and links
nmap <F9> <C-]>
map! <F9> <C-]>
```

Example: Example (Neovim)

```
-- Map F9 to follow help tags and links in Neovim
vim.keymap.set('n', '<F9>', '<C-]>', { desc = 'Follow help tag' })
vim.keymap.set({'i', 'c'}, '<F9>', '<C-]>', { desc = 'Follow help tag' })
```

Source: vim.fandom.com

46.77 Navigate Wrapped Lines Smoothly

Category: navigation
Tags: movement, wrapped-lines, visual-mode

Use visual line movement commands to navigate wrapped lines more intuitively, treating wrapped lines as visual lines instead of logical lines

Example: Example (Vim)

```
" Movement commands for wrapped lines
nnoremap j gj
nnoremap k gk
nnoremap 0 g0
nnoremap $ g$
nnoremap ^ g^
```

Example: Example (Neovim)

```
-- Movement commands for wrapped lines
vim.keymap.set('n', 'j', 'gj', { noremap = true })
vim.keymap.set('n', 'k', 'gk', { noremap = true })
vim.keymap.set('n', '0', 'g0', { noremap = true })
vim.keymap.set('n', '$', 'g$', { noremap = true })
vim.keymap.set('n', '^', 'g^', { noremap = true })
```

Source: vim.fandom.com

46.78 Navigate to Function Start and Back

Category: navigation
Tags: function-navigation, marks

Use built-in Vim commands to navigate to function start and return to original position

Example: Example (Vim)

```
Press [[ to go to function start
Press '' to return to previous position
```

Example: Example (Neovim)

```
-- Built-in Vim commands, no specific Lua implementation needed
-- [[  : Go to previous function start
-- ''  : Return to previous mark
```

Source: vim.fandom.com

46.79 Navigate to Next/Previous Open Fold

Category: navigation
Tags: folding, key-mapping, navigation

Custom function to move between open folds more intuitively, using]z and [z mappings

Example: Example (Vim)

```
function! GoToOpenFold(direction)
  let start = line('.')
  if (a:direction == "next")
    while (foldclosed(start) != -1)
      let start = start + 1
    endwhile
  else
    while (foldclosed(start) != -1)
      let start = start - 1
    endwhile
  endif
  call cursor(start, 0)
endfunction
nmap ]z :cal GoToOpenFold("next")
nmap [z :cal GoToOpenFold("prev")
```

Example: Example (Neovim)

```
function _G.go_to_open_fold(direction)
  local start = vim.fn.line('.')
  if direction == 'next' then
    while vim.fn.foldclosed(start) ~= -1 do
      start = start + 1
    end
  else
    while vim.fn.foldclosed(start) ~= -1 do
      start = start - 1
    end
  end
  vim.fn.cursor(start, 0)
end

vim.keymap.set('n', ']z', function() _G.go_to_open_fold('next') end)
vim.keymap.set('n', '[z', function() _G.go_to_open_fold('prev') end)
```

Source: vim.fandom.com

46.80 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: Example (Vim)

```
:A      " alternate file
Ctrl+^  " previous buffer
```

Source: Community contributed

46.81 Open Files Under Cursor Quickly

Category: navigation
Tags: file-operations, navigation, productivity

Provides multiple ways to open files referenced under the cursor, including opening in same window, split, or tab

Example: Example (Vim)

```
" Open file in current window
gf

" Open file in new window
<c-w>f

" Open file in new tab
<c-w>gf
```

Example: Example (Neovim)

```
-- These are built-in Vim/Neovim commands, so no direct Lua translation
↪  needed
-- Can be used directly in Neovim
-- Recommend setting up additional keymaps if desired
vim.keymap.set('n', '<leader>f', 'gf', { desc = 'Open file under cursor' })
vim.keymap.set('n', '<leader>sf', '<C-w>f', { desc = 'Open file in split' })
vim.keymap.set('n', '<leader>tf', '<C-w>gf', { desc = 'Open file in new tab'
↪  })
```

Source: vim.fandom.com

46.82 Open Files Under Cursor in Different Windows

Category: navigation
Tags: file-navigation, window-management, productivity

Quickly open files referenced under the cursor in different window modes using built-in Vim commands

Example: Example (Vim)

```
" Open file in same window
gf

" Open file in new horizontal split
<c-w>f
```

```
" Open file in new tab
<c-w>gf
```

Example: Example (Neovim)

```
-- Lua doesn't need explicit mapping, these are built-in Vim commands
-- Can be used directly in Neovim
```

Source: vim.fandom.com

46.83 Open Files Under Cursor with Multiple Methods

Category: navigation
Tags: file-navigation, cursor-movement, buffer-management

Multiple ways to open files referenced under the cursor, including in same window, new window, or new tab

Example: Example (Vim)

```
" Open file under cursor in different ways
gf              " Open in current window
<c-w>f          " Open in new horizontal split
<c-w>gf         " Open in new tab
```

Example: Example (Neovim)

```
-- Open file under cursor
-- Vim's default gf, <C-w>f, and <C-w>gf work in Neovim
-- For explicit mapping if needed:
vim.keymap.set('n', '<leader>f', 'gf', { desc = 'Open file under cursor' })
vim.keymap.set('n', '<leader>sf', '<C-w>f', { desc = 'Open file in
 ↪  horizontal split' })
vim.keymap.set('n', '<leader>tf', '<C-w>gf', { desc = 'Open file in new tab'
 ↪  })
```

Source: vim.fandom.com

46.84 Open Files Using Filename Under Cursor

Category: navigation
Tags: file-operations, navigation, productivity

Quickly open files by placing cursor on filename and using built-in commands

Example: Example (Vim)

```
" Open file in current window
gf

" Open file in new window
<c-w>f

" Open file in new tab
<c-w>gf
```

Example: Example (Neovim)

```
-- These are built-in Vim/Neovim commands, no specific Lua translation
↪  needed
-- Use as-is in normal mode
-- gf: Open file under cursor in current window
-- <C-w>f: Open file under cursor in new horizontal split
-- <C-w>gf: Open file under cursor in new tab
```

Source: vim.fandom.com

46.85 Open Files with Line Numbers

Category: navigation
Tags: file-navigation, cursor-movement

Open files directly to a specific line number using uppercase commands

Example: Example (Vim)

```
" Open file and go to line number
gF  " Same window
<c-w>F  " New window
<c-w>gF  " New tab
```

Example: Example (Neovim)

```
-- Open file and go to line number
vim.keymap.set('n', 'gF', 'gF', { desc = 'Open file at line number' })
vim.keymap.set('n', '<c-w>F', '<c-w>F', { desc = 'Open file at line number
↪  in split' })
vim.keymap.set('n', '<c-w>gF', '<c-w>gF', { desc = 'Open file at line number
↪  in new tab' })
```

Source: vim.fandom.com

46.86 Open Man Pages for Word Under Cursor

Category: navigation
Tags: man-pages, documentation, key-mapping

Quickly open man pages for the word under the cursor using the K key in Vim/Neovim

Example: Example (Vim)

```
fun! ReadMan()
  let s:man_word = expand('<cword>')
  :exe ":wincmd n"
  :exe ":r!man " . s:man_word . " | col -b"
  :exe ":goto"
  :exe ":delete"
  :exe ":set filetype=man"
endfun
map K :call ReadMan()<CR>
```

Example: Example (Neovim)

```
function _G.read_man()
  local man_word = vim.fn.expand('<cword>')
  vim.cmd('new')
  vim.cmd('read !man ' .. man_word .. ' | col -b')
  vim.cmd('norm! gg')
  vim.cmd('delete')
  vim.bo.filetype = 'man'
end

vim.keymap.set('n', 'K', _G.read_man, { desc = 'Open man page for word under
↪  cursor' })
```

Source: vim.fandom.com

46.87 Page Up/Down While Keeping Cursor Position

Category: navigation
Tags: scrolling, cursor-position, key-mapping

Scroll page up/down while maintaining the cursor's relative screen position, similar to Borland-style scrolling

Example: Example (Vim)

```
function! MyPageUp()
  let visible_lines = GetNumberOfVisibleLines()
  execute "normal " . visible_lines . "\<C-U>:set scroll=0\r"
endfunction
```

```
function! MyPageDown()
  let visible_lines = GetNumberOfVisibleLines()
  execute "normal " . visible_lines . "\<C-D>:set scroll=0\r"
endfunction

noremap <PageUp> :call MyPageUp()<CR>
noremap <PageDown> :call MyPageDown()<CR>
```

Example: Example (Neovim)

```
local function get_visible_lines()
  local cur_line = vim.fn.line('.')
  local cur_col = vim.fn.virtcol('.')
  vim.cmd('normal H')
  local top_line = vim.fn.line('.')
  vim.cmd('normal L')
  local bot_line = vim.fn.line('.')
  vim.cmd('normal ' .. cur_line .. 'G')
  vim.cmd('normal ' .. cur_col .. '|')
  return bot_line - top_line
end

vim.keymap.set('n', '<PageUp>', function()
  local visible_lines = get_visible_lines()
  vim.cmd('normal ' .. visible_lines .. '\<C-U>')
  vim.o.scroll = 0
end)

vim.keymap.set('n', '<PageDown>', function()
  local visible_lines = get_visible_lines()
  vim.cmd('normal ' .. visible_lines .. '\<C-D>')
  vim.o.scroll = 0
end)
```

Source: vim.fandom.com

46.88 Position Line at Screen Top/Middle/Bottom

Category: navigation
Tags: screen-positioning, cursor-placement

Methods to bring a specific line to the top, middle, or bottom of the screen

Example: Example (Vim)

```
42G zt   " Line 42 at top
42G z.   " Line 42 at middle
42G zb   " Line 42 at bottom

" Screen positioning
H   " Move to top of screen
```

```
M   " Move to middle of screen
L   " Move to bottom of screen
```

Example: Example (Neovim)

```
-- Screen positioning methods
vim.cmd('42 normal! zt')  -- Line 42 at top
vim.cmd('42 normal! z.')  -- Line 42 at middle
vim.cmd('42 normal! zb')  -- Line 42 at bottom

-- Move cursor to screen regions
vim.cmd('normal! H')  -- Top of screen
vim.cmd('normal! M')  -- Middle of screen
vim.cmd('normal! L')  -- Bottom of screen
```

Source: vim.fandom.com

46.89 Powerful Tag Navigation in Vim/Neovim

Category: navigation
Tags: tags, code-browsing, jump-to-definition

Efficiently navigate code by jumping to tag definitions using multiple methods

Example: Example (Vim)

```
" Jump to tag
Ctrl-]

" List multiple tag matches
g]

" Preview tag without leaving current window
Ctrl-W }
```

Example: Example (Neovim)

```
-- Jump to tag definition
vim.keymap.set('n', '<C-]>', '<cmd>tag<CR>', { desc = 'Jump to tag' })

-- List multiple tag matches
vim.keymap.set('n', 'g]', '<cmd>tselect<CR>', { desc = 'List tag matches' })

-- Preview tag in preview window
vim.keymap.set('n', '<C-w>}', '<cmd>ptag<CR>', { desc = 'Preview tag' })
```

Source: vim.fandom.com

46.90 Powerful Text Editing with Vim Motions

Category: navigation
Tags: text-objects, editing, motion-commands

Learn to combine movement commands with actions for efficient text manipulation

Example: Example (Vim)

```
# Examples of powerful text editing motions
# Delete 2 words forward
d2w
# Change 3 words backward
c3b
# Yank everything to next parenthesis
y/)
```

Example: Example (Neovim)

```
-- While Lua doesn't directly implement these, you can use vim.api to create
↪  similar mappings
-- Example of creating a wrapper for text object manipulation
vim.keymap.set('n', '<leader>d2w', 'd2w', { desc = 'Delete 2 words forward'
↪  })
vim.keymap.set('n', '<leader>c3b', 'c3b', { desc = 'Change 3 words backward'
↪  })
```

Source: vim.fandom.com

46.91 Quick Change to Current File's Directory

Category: navigation
Tags: file-navigation, directory-management

Quickly change Vim's current directory to the directory of the active file

Example: Example (Vim)

```
com Fcd cd %:p:h
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Fcd', 'cd %:p:h', {})
```

Source: vim.fandom.com

46.92 Quick Entity Definition Jump in DTDs

Category: navigation
Tags: search, mapping, dtd

Quickly jump to entity definitions in DTD files by searching backwards for the entity name

Example: Example (Vim)

```
:map <Leader>e yiw ?<!ENTITY % <C-R>"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>e', function()
  -- Yank word under cursor
  vim.cmd('yiw')
  -- Search backwards for entity definition
  vim.cmd('?<!ENTITY % ' .. vim.fn.getreg('"') .. '\n')
end, { desc = 'Jump to Entity Definition' })
```

Source: vim.fandom.com

46.93 Quick Indentation Level Navigation

Category: navigation
Tags: indentation, code-navigation, movement

Simple Alt-key mappings to quickly move between lines with the same indentation

Example: Example (Vim)

```
nnoremap <M-,> :call search('^'. matchstr(getline('.'), '(^\s*)').'\%<' .
↪  line('.') . 'l\S', 'be')<CR>
nnoremap <M-.> :call search('^'. matchstr(getline('.'), '(^\s*)').'\%>' .
↪  line('.') . 'l\S', 'e')<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-,>', function()
  local current_line = vim.fn.getline('.')
  local indent = current_line:match('^%s*')
  vim.cmd('call search("^' .. indent .. '\\%<' .. vim.fn.line('.') ..
  ↪  'l\\S", "be")')
end)

vim.keymap.set('n', '<M-.>', function()
  local current_line = vim.fn.getline('.')
  local indent = current_line:match('^%s*')
```

```
  vim.cmd('call search("^' .. indent .. '\\%>' .. vim.fn.line('.') ..
  ↪  'l\\S", "e")')
end)
```

Source: vim.fandom.com

46.94 Quick Jump to Function Definition

Category: navigation
Tags: search, code-navigation, key-mapping

Quickly jump to a function definition by placing cursor on function name and using a custom mapping

Example: Example (Vim)

```
nmap gx yiw/^\(sub\<Bar>function\)\s\+<C-R>"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'gx', function()
  -- Yank current word
  vim.cmd('normal! yiw')

  -- Search for function definition
  local search_term = vim.fn.getreg('"')
  vim.cmd('/' .. '^(sub\\|function)\\s\\+' .. search_term)
end, { desc = 'Jump to function definition' })
```

Source: vim.fandom.com

46.95 Quick Jump to Keyword Occurrences

Category: navigation
Tags: search, jump, keyword, navigation

Easily list and jump to occurrences of a keyword in the current file and included files

Example: Example (Vim)

```
" List occurrences of keyword under cursor, and
" jump to selected occurrence.
function! s:JumpOccurrence()
  let v:errmsg = ""
  exe "normal [I"
  if strlen(v:errmsg) == 0
    let nr = input("Which one: ")
    if nr =~ '\d\+'
```

```
      exe "normal! " . nr . "[\t"
    endif
  endif
endfunction

" List occurrences of keyword entered at prompt, and
" jump to selected occurrence.
function! s:JumpPrompt()
  let keyword = input("Keyword to find: ")
  if strlen(keyword) > 0
    let v:errmsg = ""
    exe "ilist! " . keyword
    if strlen(v:errmsg) == 0
      let nr = input("Which one: ")
      if nr =~ '\d\+'
        exe "ijump! " . nr . keyword
      endif
    endif
  endif
endfunction

nnoremap <Leader>j :call <SID>JumpOccurrence()<CR>
nnoremap <Leader>p :call <SID>JumpPrompt()<CR>
```

Example: Example (Neovim)

```
local function jump_occurrence()
  vim.cmd('normal! [I')
  local nr = vim.fn.input('Which one: ')
  if nr ~= '' and tonumber(nr) then
    vim.cmd('normal! ' .. nr .. '[\t')
  end
end

local function jump_prompt()
  local keyword = vim.fn.input('Keyword to find: ')
  if keyword ~= '' then
    vim.cmd('ilist! ' .. keyword)
    local nr = vim.fn.input('Which one: ')
    if nr ~= '' and tonumber(nr) then
      vim.cmd('ijump! ' .. nr .. keyword)
    end
  end
end

vim.keymap.set('n', '<Leader>j', jump_occurrence, { desc = 'Jump to keyword
↪  occurrence' })
vim.keymap.set('n', '<Leader>p', jump_prompt, { desc = 'Prompt and jump to
↪  keyword' })
```

Source: vim.fandom.com

46.96 Quick Man Page Lookup for Word Under Cursor

Category: navigation
Tags: man-pages, documentation, quick-lookup

Press K in normal mode to instantly view the man page for the word under the cursor, providing quick documentation access

Example: Example (Vim)

```
" Optional: Configure man page lookup
set keywordprg=man\ -a
```

Example: Example (Neovim)

```
-- Configure man page lookup
vim.opt.keywordprg = 'man -a'
```

Source: vim.fandom.com

46.97 Quick Navigation Between C++ Methods

Category: navigation
Tags: key-mapping, c++, search

Easily jump to next or previous method in C++ files using Alt+J/K key mappings with a regex pattern

Example: Example (Vim)

```
nnoremap <M-J> /\v^(\w+\s+)?\w+::\w+\(.*\)
nnoremap <M-K> ?\v^(\w+\s+)?\w+::\w+\(.*\)
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-J>', '/\v^(\w+\s+)?\w+::\w+\(.*\)<CR>', { desc =
↪  'Jump to next C++ method' })
vim.keymap.set('n', '<M-K>', '?\v^(\w+\s+)?\w+::\w+\(.*\)<CR>', { desc =
↪  'Jump to previous C++ method' })
```

Source: vim.fandom.com

46.98 Quick Navigation in Help Buffers

Category: navigation
Tags: key-mapping, help-buffer, custom-navigation

Remap Enter/Backspace to quickly navigate help documentation, making it more intuitive to follow links and go back

Example: Example (Vim)

```
" Remap help buffer navigation
if exists("b:did_ftplugin")
  finish
endif

" Map Enter to follow help links
nnoremap <buffer><CR> <c-]>

" Map Backspace to go back
nnoremap <buffer><BS> <c-T>
```

Example: Example (Neovim)

```
-- Lua equivalent for help buffer navigation
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'help',
  callback = function()
    vim.keymap.set('n', '<CR>', '<C-]>', { buffer = true })
    vim.keymap.set('n', '<BS>', '<C-T>', { buffer = true })
  end
})
```

Source: vim.fandom.com

46.99 Quick Navigation to Current Day in Calendar

Category: navigation
Tags: calendar, search, quick-navigation

Function to quickly jump to the current month and day in a pre-generated calendar file

Example: Example (Vim)

```
function! Today()
  let calfile = g:cal
  if (bufnr(calfile) > 0)
    exe ":bu " . bufnr(calfile)
  else
    exe "edit " . calfile
  endif
  let a = search("- " . strftime("%b %Y") . " -", "w")
  let a = search("^.. " . strftime("%d"), "w")
endfunction

map <S-F4> :call Today()<CR>
```

Example: Example (Neovim)

```
function _G.goto_today()
  local calfile = vim.g.cal
  local buf_exists = vim.fn.bufnr(calfile) > 0

  if buf_exists then
    vim.cmd('buffer ' .. vim.fn.bufnr(calfile))
  else
    vim.cmd('edit ' .. calfile)
  end

  local current_month = os.date('%b %Y')
  local current_day = os.date('%d')

  vim.fn.search('- ' .. current_month .. ' -', 'w')
  vim.fn.search('^.. ' .. current_day, 'w')
end

vim.keymap.set('n', '<S-F4>', _G.goto_today)
```

Source: vim.fandom.com

46.100 Quick Navigation with Word Labels

Category: navigation
Tags: motion, jump, labeling

Dynamically label words on screen for rapid, precise navigation without using a mouse

Example: Example (Vim)

```
let LABEL = ["a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p
↪  ","q","r","s",
"t","u","v","w","x","y","z","A","B","C","D","E","F","G","H","I",
"J","K","L","M","N","O","P","Q","R","S","T","U","V","W","X","Y",
"Z","1","2","3","4","5","6","7","8","9","0"]

function! GoTo(range)
    normal! Hmt
    for i in range(0,a:range)
        exe 'normal! Wr' . g:LABEL[i%len(g:LABEL)]
    endfor
    normal! 'tzt
    echo "Index?"
    redraw
    let label=nr2char(getchar())
    normal! u'tzt
    for i in range(0,a:range)
        exe 'normal! Wr' . (1+i/len(g:LABEL))
    endfor
    normal! 'tzt
    echo "Number?"
```

```
    redraw
    let offset=getchar()
    let offset=(49 <= offset && offset <= 57) ? offset-48 : 1
    normal! u'tzt
    let index=index(g:LABEL,label)
    exe 'normal! ' . ((offset-1)*len(g:LABEL)+index+1) . 'W'
endfu
nnoremap <TAB> :call GoTo(248)<CR>
```

Example: Example (Neovim)

```
-- Lua implementation of word labeling navigation
local function goto_labeled_word(range)
    range = range or 248
    -- Implement similar logic using Neovim Lua APIs
    -- Note: This is a complex translation and would require more detailed
    ↪  implementation
    vim.keymap.set('n', '<Tab>', function()
        -- Placeholder for dynamic word labeling navigation
        vim.notify('Word labeling navigation', vim.log.levels.INFO)
    end)
end

goto_labeled_word()
```

Source: vim.fandom.com

46.101 Quick Open Perl Module Source

Category: navigation
Tags: perl, file-operations, key-mapping

Quickly open the source file for a Perl module by placing cursor on module name and using a custom mapping

Example: Example (Vim)

```
nnoremap <Leader>pm :call LoadPerlModule()<CR>

function! LoadPerlModule()
  execute 'e `perldoc -l ' . expand("<cWORD>") . '`'
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>pm', function()
  local module = vim.fn.expand('<cWORD>')
  local cmd = string.format('e `perldoc -l %s`', module)
  vim.cmd(cmd)
end, { desc = 'Open Perl Module Source' })
```

Source: vim.fandom.com

46.102 Quick PHP Variable Navigation

Category: navigation
Tags: php, key-mapping, text-objects

Easily jump between PHP variables on the same line using custom key mappings

Example: Example (Vim)

```
noremap L f$
noremap H F$
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'L', 'f$', { desc = 'Jump to next PHP variable' })
vim.keymap.set('n', 'H', 'F$', { desc = 'Jump to previous PHP variable' })
```

Source: vim.fandom.com

46.103 Quick Python Module-Test File Switching

Category: navigation
Tags: python, file-navigation, productivity

Quickly switch between Python source files and their corresponding test files using a custom function that looks for a special comment with the test file path

Example: Example (Vim)

```
fun! JumpToTestFile()
  let line = getline("$")
  if line =~ "^### testfile: "
    let filename = strpart(line, 14)
    execute ":e " . filename
  else
    echo "TEST PATTERN ### testfile: NOT FOUND!"
  endif
endfun

nmap <buffer> <F5> :call JumpToTestFile()<CR>
```

Example: Example (Neovim)

```
function JumpToTestFile()
  local last_line = vim.fn.getline('$')
```

```
    if last_line:match('^### testfile: ') then
      local filename = last_line:sub(15)
      vim.cmd('edit ' .. filename)
    else
      print('TEST PATTERN ### testfile: NOT FOUND!')
    end
  end

  vim.api.nvim_create_autocmd('FileType', {
    pattern = 'python',
    callback = function()
      vim.keymap.set('n', '<F5>', JumpToTestFile, { buffer = true })
    end
  })
```

Source: vim.fandom.com

46.104 Quick Recovery When Spell Checking

Category: navigation
Tags: spell-checking, text-editing, navigation

When accidentally marking a misspelled word as correct during spell checking, use Ctrl-O to jump back and then zug to mark the word as not found.

Example: Example (Vim)

```
" Navigate back in jump list
Ctrl-O
" Mark word as not found
zug
```

Example: Example (Neovim)

```
-- Vim's built-in navigation and spell checking works similarly in Neovim
-- Ctrl-O to jump back
-- Use vim.cmd to execute Vim commands
vim.cmd('normal! \<C-O>')
vim.cmd('normal! zug')
```

Source: vim.fandom.com

46.105 Quick Reference Documents in Tabs

Category: navigation
Tags: help, documentation, tabs

Open help pages and reference documents in separate tabs for easy access without disrupting current workspace

Example: Example (Vim)

```
:tab help toc
:tab help <topic>
```

Example: Example (Neovim)

```
-- Open help in a new tab
vim.cmd('tab help toc')
vim.cmd('tab help ' .. topic)
```

Source: vim.fandom.com

46.106 Quick Tab Navigation and Management

Category: navigation
Tags: tab-management, key-mapping, buffer-navigation

Efficient ways to navigate and manage tabs in Vim/Neovim, including quick key mappings for switching and moving tabs

Example: Example (Vim)

```
" Navigate tabs
nnoremap <C-Left> :tabprevious<CR>
nnoremap <C-Right> :tabnext<CR>

" Move tabs
nnoremap <silent> <A-Left> :tabm -1<CR>
nnoremap <silent> <A-Right> :tabm +1<CR>

" Toggle buffer display in tabs
let notabs = 0
nnoremap <silent> <F8> :let notabs=!notabs<Bar>:if
↪  notabs<Bar>:tabo<Bar>:else<Bar>:tab ball<Bar>:tabn<Bar>:endif<CR>
```

Example: Example (Neovim)

```
-- Navigate tabs
vim.keymap.set('n', '<C-Left>', ':tabprevious<CR>', { noremap = true, silent
↪  = true })
vim.keymap.set('n', '<C-Right>', ':tabnext<CR>', { noremap = true, silent =
↪  true })

-- Move tabs
vim.keymap.set('n', '<A-Left>', ':tabm -1<CR>', { noremap = true, silent =
↪  true })
vim.keymap.set('n', '<A-Right>', ':tabm +1<CR>', { noremap = true, silent =
↪  true })
```

```
-- Toggle buffer display in tabs
vim.api.nvim_set_var('notabs', 0)
vim.keymap.set('n', '<F8>', function()
  local notabs = vim.g.notabs
  vim.g.notabs = notabs == 0 and 1 or 0
  if notabs == 0 then
    vim.cmd('tabo')
  else
    vim.cmd('tab ball')
    vim.cmd('tabn')
  end
end, { silent = true })
```

Source: vim.fandom.com

46.107 Quickfix Window Navigation Mappings

Category: navigation
Tags: quickfix, key-mapping, navigation

Add convenient mappings to navigate through quickfix list with easy key bindings

Example: Example (Vim)

```
nnoremap <kPlus>     :cnext<CR> :norm! zz<CR>
nnoremap <kMinus>    :cprev<CR> :norm! zz<CR>
nnoremap <kMultiply> :cc<CR> :norm! zz<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<kPlus>', ':cnext<CR>zz', { silent = true })
vim.keymap.set('n', '<kMinus>', ':cprev<CR>zz', { silent = true })
vim.keymap.set('n', '<kMultiply>', ':cc<CR>zz', { silent = true })
```

Source: vim.fandom.com

46.108 Quickly Get Current Function Name

Category: navigation
Tags: function-navigation, code-browsing, mapping

Easily retrieve the name of the current function you're in using a custom mapping that searches backwards to the function definition

Example: Example (Vim)

```
map _F ma[[k"xy$`a:echo @x<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '_F', function()
  -- Move to previous '{' in first column
  vim.cmd('normal! ma[[k"xy$`a')
  -- Echo the function name
  print(vim.fn.getreg('x'))
end, { desc = 'Show current function name' })
```

Source: vim.fandom.com

46.109 Quickly Open Perl Module Source

Category: navigation
Tags: perl, file-operations, productivity

Quickly open the source file for a Perl module by cursor position or command

Example: Example (Vim)

```
nnoremap <Leader>pm :call LoadPerlModule()<CR>

function! LoadPerlModule()
  execute 'e `perldoc -l ' . expand("<cWORD>") . '`'
endfunction
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>pm', function()
  local module = vim.fn.expand('<cWORD>')
  local cmd = string.format('e `perldoc -l %s`', module)
  vim.cmd(cmd)
end, { desc = 'Open Perl Module Source' })
```

Source: vim.fandom.com

46.110 Quickly Search Vim Help Topics

Category: navigation
Tags: help, search, documentation

Multiple efficient methods to search and navigate Vim help topics, including using wildcard completion and built-in search commands

Example: Example (Vim)

```
" Search help topics with partial match
:he shell<c-d>
```

```
" Use helpgrep to search help files
:helpgrep shell
:copen

" Create a custom help search command
:command! -nargs=+ -complete=tag Helpt :Tselect /.*<args>.*
```

Example: Example (Neovim)

```
-- Built-in help search
-- Use vim.cmd to run help commands in Lua
vim.cmd('helpgrep shell')
vim.cmd('copen')

-- Create a custom help search function
vim.api.nvim_create_user_command('Helpt', function(opts)
  vim.cmd('Tselect /.*' .. opts.args .. '.*')
end, { nargs = '+', complete = 'tag' })
```

Source: vim.fandom.com

46.111 Random Line Selection with Range Support

Category: navigation
Tags: random, range-selection, ruby

Advanced random line selection that supports specifying a range for random line jumping

Example: Example (Vim)

```
function! RandomLine() range
  ruby first_line = (VIM::evaluate 'a:firstline').to_i
  ruby last_line = (VIM::evaluate 'a:lastline').to_i
  ruby VIM::command(((rand last_line - first_line + 1) + first_line).to_s)
endfunction
command! -range=% RandomLine <line1>,<line2>call RandomLine()
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('RandomLine', function(opts)
  local first_line = opts.line1
  local last_line = opts.line2
  local random_line = math.random(first_line, last_line)
  vim.cmd(string.format('%d', random_line))
end, { range = true })
```

Source: vim.fandom.com

46.112 Read Javadoc Directly in Vim

Category: navigation
Tags: documentation, java, help

Use a custom doclet to generate Vim help files for Java documentation, allowing quick access to Javadoc within Vim

Example: Example (Vim)

```
" Set keywordprg to blank to use 'K' for quick Javadoc lookup
set keywordprg=
" Use :help to access Javadoc
:help String
:help java.util.List
```

Example: Example (Neovim)

```
-- Configure keywordprg for quick Javadoc lookup
vim.o.keywordprg = ''
-- Access Javadoc using :help
-- Requires installing Vim Doclet from vimdoclet.sourceforge.net
```

Source: vim.fandom.com

46.113 Recursive File Finding with Path Option

Category: navigation
Tags: file-search, project-navigation, recursive-search

Easily search and open files across multiple directories using Vim's path option with recursive search

Example: Example (Vim)

```
:set path+=./**
:find filename.txt  # Find file in current and subdirectories
:sfind filename.txt  # Open in split window
```

Example: Example (Neovim)

```
-- Set recursive path
vim.opt.path:append('./**')

-- Lua doesn't change :find syntax, but you can use vim.cmd
vim.cmd('find filename.txt')
vim.cmd('sfind filename.txt')
```

Source: vim.fandom.com

46.114 Recursive File Search with Fuzzy Matching

Category: navigation
Tags: file-search, fuzzy-finder, productivity

Quickly search and open files recursively in the current directory tree using fuzzy matching

Example: Example (Vim)

```
nmap <F5> :FuzzyFinderFile **/<CR>
map <Leader>t :FufFile **/<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', function()
  require('fzf-lua').files({ cwd = vim.fn.getcwd() })
end, { desc = 'Recursive file search' })

vim.keymap.set('n', '<leader>t', function()
  require('fzf-lua').files({ cwd = vim.fn.getcwd() })
end, { desc = 'Recursive file search' })
```

Source: vim.fandom.com

46.115 Recursive Project File Finding

Category: navigation
Tags: project-management, file-search, productivity

Quickly find files in a project by setting recursive path search, enabling fast file navigation across deep directory structures

Example: Example (Vim)

```
set path=$PWD/**
```

Example: Example (Neovim)

```
vim.opt.path = vim.fn.getcwd() .. '/**'
```

Source: vim.fandom.com

46.116 Screen-Based Movement Commands

Category: navigation
Tags: screen-movement, viewport, scrolling

Quick ways to move through the viewport without using mouse or scrollbar

Example: Example (Vim)

```
" H - top of screen
" M - middle of screen
" L - bottom of screen
" 10<PageUp> - move 10 pages up
```

Example: Example (Neovim)

```
-- Screen movement is native in Neovim
-- Can wrap with custom mappings for enhanced control
vim.keymap.set('n', '<leader>H', 'H', { desc = 'Move to top of screen' })
vim.keymap.set('n', '<leader>M', 'M', { desc = 'Move to middle of screen' })
vim.keymap.set('n', '<leader>L', 'L', { desc = 'Move to bottom of screen' })
```

Source: vim.fandom.com

46.117 Scroll Without Moving Cursor

Category: navigation
Tags: scroll, cursor-position, screen-movement

Scroll the screen up or down while keeping the cursor on the same screen line, providing a smooth reading experience

Example: Example (Vim)

```
function! s:Saving_scroll(cmd)
  let save_scroll = &scroll
  execute 'normal! ' . a:cmd
  let &scroll = save_scroll
endfunction

nnoremap <C-J> :call <SID>Saving_scroll("1<C-V><C-D>")<CR>
vnoremap <C-J> <Esc>:call <SID>Saving_scroll("gv1<C-V><C-D>")<CR>
nnoremap <C-K> :call <SID>Saving_scroll("1<C-V><C-U>")<CR>
vnoremap <C-K> <Esc>:call <SID>Saving_scroll("gv1<C-V><C-U>")<CR>
```

Example: Example (Neovim)

```
local function saving_scroll(cmd)
  local save_scroll = vim.o.scroll
  vim.cmd('normal! ' .. cmd)
  vim.o.scroll = save_scroll
end

vim.keymap.set('n', '<C-J>', function() saving_scroll('1\<C-D>') end)
vim.keymap.set('v', '<C-J>', function() saving_scroll('gv1\<C-D>') end)
```

```
vim.keymap.set('n', '<C-K>', function() saving_scroll('1\<C-U>') end)
vim.keymap.set('v', '<C-K>', function() saving_scroll('gv1\<C-U>') end)
```

Source: vim.fandom.com

46.118 Smart Middle of Line Movement

Category: navigation
Tags: cursor-movement, text-editing

Move cursor to the middle of the physical line, ignoring leading/trailing whitespace

Example: Example (Vim)

```
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>
```

Example: Example (Neovim)

```
local function gm()
  vim.cmd('normal! ^')
  local first_col = vim.fn.virtcol('.')
  vim.cmd('normal! g_')
  local last_col = vim.fn.virtcol('.')
  vim.cmd('normal! ' .. math.floor((first_col + last_col) / 2) .. '|')
end

vim.keymap.set('n', 'gm', gm, { silent = true })
vim.keymap.set('o', 'gm', gm, { silent = true })
```

Source: vim.fandom.com

46.119 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: Example (Vim)

```
[/  " jump to previous start of C comment
]/  " jump to next end of C comment
[*  " same as [/ (alternative)
]*  " same as ]/ (alternative)
```

Source: Community contributed

46.120 Square bracket navigation - changes and diffs

Category: Navigation
Tags: change, diff, navigation

Use `[c` and `]c` to jump between changes in diff mode.

Example: Example (Vim)

```
[c  " jump to previous change
]c  " jump to next change
```

Source: Community contributed

46.121 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: Example (Vim)

```
[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
```

Source: Community contributed

46.122 Square bracket navigation - folds

Category: Navigation
Tags: fold, navigation, code

Use `[z` and `]z` to jump to start/end of open fold.

Example: Example (Vim)

```
[z  " jump to start of open fold
]z  " jump to end of open fold
```

Source: Community contributed

46.123 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: Example (Vim)

```
[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
```

Source: Community contributed

46.124 Square bracket navigation - member functions

Category: Navigation
Tags: function, member, class, navigation

Use `[m` and `]m` to jump between member function starts.

Example: Example (Vim)

```
[m  " jump to previous start of member function
]m  " jump to next start of member function
```

Source: Community contributed

46.125 Square bracket navigation - preprocessing

Category: Navigation
Tags: preprocessing, define, include

Use `[#` and `]#` to jump between #if/#else/#endif blocks.

Example: Example (Vim)

```
[#  " jump to previous #if, #else, or #ifdef
]#  " jump to next #endif or #else
```

Source: Community contributed

46.126 Square bracket navigation - sections

Category: Navigation
Tags: section, navigation, document

Use `[[` and `]]` to jump between sections, `[]` and `][` to jump between SECTIONS (different formatting).

Example: Example (Vim)

```
[[  " jump to previous section
]]  " jump to next section
[]  " jump to previous SECTION
][  " jump to next SECTION
```

Source: Community contributed

46.127 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: Example (Vim)

```
[d  " show first #define matching word
]d  " show first #define matching word
[i  " show first line containing word
]i  " show first line containing word
```

Source: Community contributed

46.128 Square bracket navigation - spelling

Category: Navigation
Tags: spelling, error, navigation

Use `[s` and `]s` to jump between misspelled words.

Example: Example (Vim)

```
[s  " jump to previous misspelled word
]s  " jump to next misspelled word
```

Source: Community contributed

46.129 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: Example (Vim)

```
[(  " jump to previous unmatched (
])  " jump to next unmatched )
[{  " jump to previous unmatched {
]}  " jump to next unmatched }
```

Source: Community contributed

46.130 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: Example (Vim)

```
:Lexplore    " toggle left explorer
:Vexplore    " open explorer in vertical split
:Sexplore    " open explorer in horizontal split
:Explore     " open explorer in current window
```

Source: Community contributed

46.131 Track and Search Recently Visited Directories

Category: navigation
Tags: directory-tracking, autocmd, productivity

Automatically track and cache visited directories for quick fuzzy navigation

Example: Example (Vim)

```
" Track visited directories
autocmd DirChanged * call SaveLastDir()

function! SaveLastDir()
  if !exists('g:dirs')
    call LoadDir()
    let g:lastdir = ''
  endif

  let reg = getcwd()
  if reg == g:lastdir
    return
  endif
  call add(g:dirs, reg)
endfunction
```

Example: Example (Neovim)

```
-- Lua implementation of directory tracking
vim.api.nvim_create_autocmd('DirChanged', {
  callback = function()
    local current_dir = vim.fn.getcwd()
    if not vim.g.dirs then
      vim.g.dirs = {}
    end

    if current_dir ~= vim.g.last_dir then
      table.insert(vim.g.dirs, current_dir)
      vim.g.last_dir = current_dir
    end
  end
})
```

Source: vim.fandom.com

46.132 Use Alt/Meta Keys for Quick Navigation

Category: navigation
Tags: terminal, key-mapping, navigation

In terminal Vim, use Alt/Meta + key combinations to quickly switch modes and navigate without pressing Escape

Example: Example (Vim)

```
" Use Alt keys for quick navigation in terminal Vim
" Alt+h, Alt+j, Alt+k, Alt+l for movement
" Alt+o to open line below
" Alt+A to append to end of line
```

Example: Example (Neovim)

```
-- Terminal Vim navigation (configurable in init.lua)
-- Requires terminal to send appropriate escape sequences
vim.g.terminal_alt_key_navigation = true
```

Source: vim.fandom.com

46.133 Use Alt/Meta Keys in Terminal Vim

Category: navigation
Tags: terminal, key-mapping, productivity

Leverage Alt/Meta keys to quickly switch modes and perform actions without Escape

Example: Example (Vim)

```
" No direct Vimscript implementation - depends on terminal configuration
```

Example: Example (Neovim)

```
-- Configure terminal to send Alt/Meta key events
-- Specific implementation depends on terminal settings
```

Source: vim.fandom.com

46.134 Use Environment Variables for Dynamic File Paths

Category: navigation
Tags: file-navigation, environment

Define environment variables to simplify file path references across different systems

Example: Example (Vim)

```
let $mydir = 'C:/Documents and Settings/My Name/My Documents'
```

Example: Example (Neovim)

```
-- Set environment variable for file paths
vim.env.mydir = 'C:/Documents and Settings/My Name/My Documents'
```

Source: vim.fandom.com

46.135 Use K to View PHP Function Help

Category: navigation
Tags: php, documentation, key-mapping

Configure Vim to show PHP function documentation when pressing K on a function name

Example: Example (Vim)

```
" In ftplugin/php.vim
set keywordprg=:help
```

Example: Example (Neovim)

```
-- In lua/ftplugin/php.lua
vim.opt.keywordprg = ':help'
```

Source: vim.fandom.com

46.136 Use g; and g, for Edit Location Navigation

Category: navigation
Tags: edit-locations, movement

Move backward and forward through edit locations within a file

Example: Example (Vim)

```
" g; - go to previous edit location
" g, - go to next edit location
```

Example: Example (Neovim)

```
-- Built-in Vim/Neovim commands work as-is
-- Recommended mapping for clarity
vim.keymap.set('n', 'g;', 'g;', { desc = 'Go to previous edit location' })
vim.keymap.set('n', 'g,', 'g,', { desc = 'Go to next edit location' })
```

Source: vim.fandom.com

46.137 Use g_ to Navigate End of Line in Langmap

Category: navigation
Tags: langmap, cursor-movement, keyboard-layout

When using a non-English keyboard layout that lacks the $ symbol, use g_ to move to

the last non-blank character in the line

Example: Example (Vim)

```
" No specific code needed, just use g_ motion
```

Example: Example (Neovim)

```
-- Use g_ motion to navigate to end of line
-- Works with any keyboard layout
```

Source: vim.fandom.com

46.138 View Man Pages Directly in Vim/Neovim

Category: navigation
Tags: man-pages, documentation, help

Quickly view man pages within Vim/Neovim without leaving the editor

Example: Example (Vim)

```
runtime ftplugin/man.vim
:Man <command>
set keywordprg=:Man
```

Example: Example (Neovim)

```
-- Enable man page viewer
vim.cmd.runtime('ftplugin/man.vim')

-- Open man page
vim.cmd.Man('<command>')

-- Set K to use :Man by default
vim.o.keywordprg = ':Man'
```

Source: vim.fandom.com

46.139 View Multiple Man Page Chapters

Category: navigation
Tags: man-pages, documentation, advanced-lookup

Prefix the chapter number before pressing K to view specific sections of a man page

Example: Example (Vim)

```
" Example: 3 K would look up chapter 3 man pages
```

Example: Example (Neovim)

```
-- No direct Lua equivalent, relies on Vim's built-in man page lookup
```

Source: vim.fandom.com

46.140 View jump list

Category: Navigation
Tags: jump, list, view

Use `:jumps` to show the jump list with all stored positions.

Example: Example (Vim)

```
:jumps  " show jump list
```

Source: Community contributed

46.141 Vim-Style Browser Navigation Extensions

Category: navigation
Tags: browser-navigation, key-mapping, productivity

Several browser extensions allow Vim-like keyboard navigation, reducing mouse usage and improving browsing efficiency

Example: Example (Neovim)

```
-- Recommended browser extensions for Vim-like navigation:
-- Vimium (Chrome/Chromium)
-- Tridactyl (Firefox)
-- Surfingkeys (Cross-browser)
-- Vim Vixen (Firefox)
```

Source: vim.fandom.com

46.142 Windows-like Ctrl+Left/Right Word Navigation

Category: navigation
Tags: key-mapping, windows-compatibility

Remap Ctrl+Left and Ctrl+Right to move by words, mimicking Windows text editing be-

havior

Example: Example (Vim)

```
"Edit mapping for word navigation
nnoremap <C-Left> b
nnoremap <C-Right> w

"For visual and insert modes
vnoremap <C-S-Left> b
vnoremap <C-S-Right> w
inoremap <C-S-Left> <C-\><C-O>b
inoremap <C-S-Right> <C-\><C-O>w
```

Example: Example (Neovim)

```
-- Word navigation like Windows
vim.keymap.set('n', '<C-Left>', 'b', { noremap = true })
vim.keymap.set('n', '<C-Right>', 'w', { noremap = true })

-- Additional mappings for visual and insert modes
vim.keymap.set('v', '<C-S-Left>', 'b', { noremap = true })
vim.keymap.set('v', '<C-S-Right>', 'w', { noremap = true })
vim.keymap.set('i', '<C-S-Left>', '<C-\><C-O>b', { noremap = true })
vim.keymap.set('i', '<C-S-Right>', '<C-\><C-O>w', { noremap = true })
```

Source: vim.fandom.com

46.143 Wrap Cursor Movement Between Lines

Category: navigation
Tags: cursor-movement, key-mapping, configuration

Allows left/right cursor keys and h/l to wrap to previous/next line when reaching line boundaries, improving navigation fluidity

Example: Example (Vim)

```
set whichwrap+=<,>,h,l,[,]
```

Example: Example (Neovim)

```
vim.opt.whichwrap:append '<,>h,l,[,]'
```

Source: vim.fandom.com

CHAPTER 47

Neovim features

47.1 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: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufWritePre', {
  pattern = '*.lua',
  callback = function()
    vim.lsp.buf.format()
  end,
  desc = 'Format Lua files on save'
})
```

47.2 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: Example (Neovim)

```
-- 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)
```

47.3 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: Example (Vim)

```
:term           " open terminal
Ctrl+\ Ctrl+n   " exit terminal mode
```

47.4 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: Example (Neovim)

```
-- 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()
```

47.5 Extended marks

Category: Neovim Features
Tags: marks, extmarks, api, highlighting

Use extmarks for advanced text annotations and virtual text that persists across edits.

Example: Example (Neovim)

```
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'
```

```
})
```

47.6 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: Example (Neovim)

```
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'
})
```

47.7 Health checks

Category: Neovim Features
Tags: health, check, diagnostics, system

Use `:checkhealth` to diagnose Neovim installation and plugin issues.

Example: Example (Vim)

```
:checkhealth           " check all health
:checkhealth nvim      " check Neovim core
:checkhealth telescope " check specific plugin
```

47.8 Lua configuration

Category: Neovim Features
Tags: lua, configuration, modern, scripting

Use Lua for configuration instead of Vimscript for better performance and modern syntax.

Example: Example (Neovim)

```
-- ~/.config/nvim/init.lua
vim.opt.number = true
vim.opt.relativenumber = true
vim.keymap.set('n', '<leader>ff', '<cmd>Telescope find_files<cr>')
```

47.9 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: Example (Vim)

```
/word   " search for 'word'
cgn     " change next match
.       " repeat change on next match
```

47.10 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: Example (Vim)

```
:cn     " next quickfix item
:cp     " previous quickfix item
:copen  " open quickfix window
```

47.11 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: Example (Neovim)

```
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
})
```

47.12 Statusline and tabline API

Category: Neovim Features
Tags: statusline, tabline, ui, customization

Customize statusline and tabline using Lua functions for dynamic content.

Example: Example (Neovim)

```
function _G.custom_statusline()
  return '%f %m %r%=%l,%c %p%%'
end

vim.opt.statusline = '%!v:lua.custom_statusline()'
```

47.13 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: Example (Vim)

```
vaf  " select around function
vif  " select inside function
vac  " select around class
```

47.14 User commands

Category: Neovim Features
Tags: commands, user, custom, lua

Create custom user commands with Lua for better functionality and completion.

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Hello', function(opts)
  print('Hello ' .. (opts.args or 'World'))
end, {
  nargs = '?',
  desc = 'Say hello to someone'
})
```

47.15 Virtual text

Category: Neovim Features
Tags: virtual, text, inline, diagnostics

Display virtual text inline for diagnostics, git blame, or other contextual information.

Example: Example (Neovim)

```
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 48

Neovim terminal

48.1 Hidden terminal processes

Category: Terminal
Tags: terminal, hidden, background, process

Use hidden terminals to run background processes while maintaining editor workflow.

Example: Example (Vim)

```
: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
})
```

48.2 Split terminal workflows

Category: Terminal
Tags: terminal, split, workflow, development

Use terminal splits for integrated development workflows without leaving Neovim.

Example: Example (Vim)

```
:split | terminal         " horizontal split terminal
:vsplit | terminal        " vertical split terminal
:tabnew | terminal        " terminal in new tab
" Create persistent terminal splits for common tasks
```

48.3 Terminal REPL workflows

Category: Terminal
Tags: terminal, repl, workflow, interactive

Use terminal for REPL-driven development with language-specific interactive environ-

ments.

Example: Example (Vim)

```
: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
```

48.4 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: Example (Vim)

```
: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
})
```

48.5 Terminal autocmd events

Category: Terminal
Tags: terminal, autocmd, events, TermOpen

Use terminal-specific autocommand events to customize terminal behavior and appearance.

Example: Example (Vim)

```
: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
```

48.6 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: Example (Vim)

```
: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)
```

48.7 Terminal buffer naming

Category: Terminal
Tags: terminal, buffer, naming, identification

Use buffer naming to identify and switch between multiple terminal instances easily.

Example: Example (Vim)

```
:terminal ++title=server " named terminal buffer
:terminal ++title=build
:ls                      " shows named terminal buffers
:buffer server           " switch to named terminal
```

48.8 Terminal color and appearance

Category: Terminal
Tags: terminal, color, appearance, highlight

Use terminal-specific highlighting and color configuration for better visual integration.

Example: Example (Vim)

```
: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
```

48.9 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: Example (Vim)

```
: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
```

48.10 Terminal environment variables

Category: Terminal
Tags: terminal, environment, variables, env

Use environment variable control for terminal processes launched from Neovim.

Example: Example (Vim)

```
:let $EDITOR = 'nvim'
:let $TERM = 'xterm-256color'
:terminal env              " show environment
:terminal ENV_VAR=value command  " set env var for command
```

48.11 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: Example (Vim)

```
: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
```

48.12 Terminal output processing

Category: Terminal
Tags: terminal, output, processing, callback

Use terminal output callbacks to process terminal output and integrate with editor workflows.

Example: Example (Vim)

```
: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
})
```

48.13 Terminal plugin integration

Category: Terminal
Tags: terminal, plugin, integration, compatibility

Use terminal integration patterns that work well with common Neovim plugins and workflows.

Example: Example (Vim)

```
" 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
```

48.14 Terminal process communication

Category: Terminal
Tags: terminal, process, communication, stdin

Use `chansend()` to send input to terminal processes programmatically.

Example: Example (Vim)

```
: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
```

48.15 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: Example (Vim)

```
: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
```

48.16 Terminal session persistence

Category: Terminal
Tags: terminal, session, persistence, restore

Use terminal session restoration to maintain terminal state across Neovim sessions.

Example: Example (Vim)

```
" 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
```

48.17 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: Example (Vim)

```
: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
```

48.18 Terminal window management

Category: Terminal
Tags: terminal, window, management, layout

Use advanced window management for terminal-focused layouts and workflows.

Example: Example (Vim)

```
:tabnew | terminal        " dedicated terminal tab
:only | split | terminal  " editor above, terminal below
:vsplit | terminal | vertical resize 80  " side terminal
" Create terminal-focused layout commands
```

48.19 Terminal with specific shell

Category: Terminal
Tags: terminal, shell, specific, custom

Use `:terminal` with specific shell or command for customized terminal environments.

Example: Example (Vim)

```
:terminal bash            " specific shell
:terminal python3         " Python REPL
:terminal node            " Node.js REPL
:terminal zsh -c 'cd ~/project && zsh'  " custom environment
```

48.20 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: Example (Vim)

```
: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 49

Normal mode (advanced)

49.1 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: Example (Vim)

```
Ctrl+^    " switch to alternate buffer
Ctrl+6    " same as Ctrl+^ (switch to alternate)
```

49.2 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: Example (Vim)

```
~         " toggle case of character under cursor
g~w       " toggle case of word
guw       " lowercase word
gUw       " uppercase word
g~~       " toggle case of entire line
```

49.3 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: Example (Vim)

```
g~$       " toggle case from cursor to end of line
guw       " lowercase word under cursor
gUiw      " uppercase inner word
g~ap      " toggle case of paragraph
```

49.4 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: Example (Vim)

```
cw        " change word
cc        " change entire line
C         " change from cursor to end of line
ciw       " change inner word
cip       " change inner paragraph
```

49.5 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: Example (Vim)

```
" In insert mode:
Ctrl+n    " next completion
Ctrl+p    " previous completion
Ctrl+x Ctrl+f  " filename completion
Ctrl+x Ctrl+l  " line completion
```

49.6 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: Example (Vim)

```
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
```

49.7 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: Example (Vim)

```
" 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)
```

49.8 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: Example (Vim)

```
:          " enter command-line mode
Q          " enter Ex mode (exit with :vi)
```

49.9 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: Example (Vim)

```
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
```

49.10 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: Example (Vim)

```
!}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
```

49.11 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: Example (Vim)

```
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
```

49.12 Format text

Category: Normal Mode
Tags: format, indent, text, alignment

Use `=` with motion to format/indent text, `==` to format current line.

Example: Example (Vim)

```
==        " format current line
=ap       " format around paragraph
=G        " format from cursor to end of file
gg=G      " format entire file
```

49.13 Go to column

Category: Normal Mode
Tags: column, position, horizontal, goto

Use `{number}|` to go to specific column number on current line.

Example: Example (Vim)

```
10|       " go to column 10
1|        " go to column 1 (beginning of line)
$         " go to end of line (last column)
```

49.14 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. Works on hex (0x1F), octal (017), binary (0b1010), and decimals.

Example: Example (Vim)

```
Ctrl+a    " increment number under cursor
Ctrl+x    " decrement number under cursor
5Ctrl+a   " increment by 5
```

49.15 Indent and outdent

Category: Normal Mode
Tags: indent, outdent, shift, tab

Use > to indent, < to outdent. Works with motions and counts.

Example: Example (Vim)

```
>>        " indent current line
<<        " outdent current line
>ap       " indent around paragraph
3>>       " indent next 3 lines
>G        " indent from cursor to end
```

49.16 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: Example (Vim)

```
I         " insert at beginning of line (first non-blank)
A         " append at end of line
gI        " insert at column 1 (absolute beginning)
```

49.17 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: Example (Vim)

```
J         " join lines with space
gJ        " join lines without space
3J        " join current line with next 2 lines
```

49.18 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: Example (Vim)

```
yyp       " duplicate current line below
yyP       " duplicate current line above
Yp        " same as yyp (Y yanks line, p pastes)
```

49.19 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: Example (Vim)

```
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
```

49.20 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: Example (Vim)

```
o         " open new line below and enter insert mode
O         " open new line above and enter insert mode
3o        " open 3 new lines below
```

49.21 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: Example (Vim)

```
p         " put after cursor/line
P         " put before cursor/line
"ap       " put from register 'a' after cursor
"0p       " put from yank register (register 0)
```

49.22 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: Example (Vim)

```
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
```

49.23 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: Example (Vim)

```
.         " repeat last change
dd.       " delete line, then repeat delete
cw foo<Esc>.  " change word to foo, then repeat on next word
```

49.24 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: Example (Vim)

```
ra        " replace character under cursor with 'a'
r<Space>  " replace with space
R         " enter replace mode
```

49.25 Search under cursor

Category: Normal Mode
Tags: search, word, cursor, find, highlight

Use `*` to search forward for word under cursor, `#` to search backward.

Example: Example (Vim)

```
*          " 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
```

49.26 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: Example (Vim)

```
]s         " next misspelled word
[s         " previous misspelled word
z=         " show spelling suggestions
zg         " add word to good word list
zw         " add word as misspelled
```

49.27 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: Example (Vim)

```
Ctrl+]     " jump to tag definition
Ctrl+t     " return from tag jump
g Ctrl+]   " show list of matching tags
```

49.28 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: Example (Vim)

```
u          " undo last change
Ctrl+r     " redo last undone change
U          " undo all changes on current line
```

49.29 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: Example (Vim)

```
v         " start character-wise visual selection
V         " start line-wise visual selection
Ctrl+v    " start block-wise visual selection
gv        " reselect last visual selection
```

49.30 Window navigation

Category: Normal Mode
Tags: window, switch, navigate, split

Use `Ctrl+w` followed by direction to move between windows.

Example: Example (Vim)

```
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
```

49.31 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: Example (Vim)

```
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 50
Performance

50.1 Disable unused features

Category: Performance
Tags: disable, features, optimization, settings

Disable unused built-in features to improve performance and reduce memory usage.

Example: Example (Neovim)

```
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
```

Source: Community contributed

50.2 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: Example (Neovim)

```
-- lazy.nvim example
{
  "telescope.nvim",
  cmd = "Telescope",  -- load only when command is used
  keys = "<leader>ff" -- load only when key is pressed
}
```

Source: Community contributed

50.3 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: Example (Neovim)

```
-- 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
```

Source: Community contributed

50.4 Memory usage monitoring

Category: Performance
Tags: memory, monitoring, usage, debug

Monitor Neovim memory usage to identify memory leaks or excessive usage.

Example: Example (Vim)

```
:lua print(collectgarbage("count") .. " KB")  " current memory usage
:lua collectgarbage()  " force garbage collection
```

Source: Community contributed

50.5 Optimize file type detection

Category: Performance
Tags: filetype, detection, performance

Use efficient filetype detection and disable unnecessary patterns.

Example: Example (Neovim)

```
vim.g.do_filetype_lua = 1  -- use Lua for filetype detection
vim.g.did_load_filetypes = 0  -- don't use Vim script detection
```

Source: Community contributed

50.6 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: Example (Neovim)

```
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'
})
```

Source: Community contributed

50.7 Optimize updatetime

Category: Performance
Tags: updatetime, performance, responsiveness

Set appropriate updatetime for better responsiveness (default 4000ms is often too slow).

Example: Example (Neovim)

```
vim.opt.updatetime = 250  -- faster completion and diagnostics
```

Source: Community contributed

50.8 Profile Lua code

Category: Performance
Tags: profile, lua, performance, debug

Use built-in Lua profiler to identify performance bottlenecks in your config.

Example: Example (Neovim)

```
-- 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))
```

Source: Community contributed

50.9 Profile startup time

Category: Performance
Tags: profile, startup, performance

Use `nvim --startuptime profile.log` to profile Neovim startup time.
Source: Community contributed

50.10 Reduce redraw frequency

Category: Performance
Tags: redraw, display, performance, optimization

Use lazyredraw to improve performance during macros and complex operations.

Example: Example (Neovim)

```
vim.opt.lazyredraw = true  -- don't redraw during macros
```

Source: Community contributed

50.11 Syntax highlighting limits

Category: Performance
Tags: syntax, highlighting, limits, large files

Set limits for syntax highlighting to maintain performance on large files.

Example: Example (Neovim)

```
vim.opt.synmaxcol = 200  -- don't highlight lines longer than 200 chars
vim.g.syntax_timeout = 1000  -- timeout after 1 second
```

Source: Community contributed

50.12 Use swap files efficiently

Category: Performance
Tags: swap, files, memory, performance

Configure swap files for better performance and crash recovery.

Example: Example (Neovim)

```
vim.opt.swapfile = true
vim.opt.directory = vim.fn.expand('~/.local/share/nvim/swap//')
vim.opt.updatecount = 100  -- write swap after 100 keystrokes
```

Source: Community contributed

CHAPTER 51

Performance (advanced)

51.1 Autocommand optimization

Category: Performance Optimization Advanced
Tags: autocommand, event, performance, grouping

Optimize autocommand usage to reduce event processing overhead.

Example: Example (Neovim)

```
-- 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
})
```

51.2 Buffer and window optimization

Category: Performance Optimization Advanced
Tags: buffer, window, memory, cleanup, optimization

Optimize buffer and window management for better memory usage.

Example: Example (Neovim)

```
-- 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
```

51.3 Completion system optimization

Category: Performance Optimization Advanced
Tags: completion, cmp, performance, async, cache

Optimize completion systems for faster and more responsive completions.

Example: Example (Neovim)

```
-- 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 },
  },
})
```

51.4 Concurrent operations optimization

Category: Performance Optimization Advanced
Tags: concurrent, async, parallel, threading, performance

Implement concurrent operations for better performance and responsiveness.

Example: Example (Neovim)

```
-- 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
```

51.5 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: Example (Neovim)

```
-- 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
})
```

51.6 Display and rendering optimization

Category: Performance Optimization Advanced
Tags: display, render, redraw, terminal, optimization

Optimize display rendering and terminal performance.

Example: Example (Neovim)

```
-- 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
```

51.7 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: Example (Neovim)

```
-- 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
```

51.8 LSP performance optimization

Category: Performance Optimization Advanced
Tags: lsp, language, server, performance, debounce

Optimize Language Server Protocol interactions for better responsiveness.

Example: Example (Neovim)

```
-- 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 })
```

51.9 Large file handling optimization

Category: Performance Optimization Advanced
Tags: large, file, handling, performance, memory

Implement specialized handling for large files to maintain performance.

Example: Example (Neovim)

```
-- 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
})
```

51.10 Memory management and garbage collection

Category: Performance Optimization Advanced
Tags: memory, garbage, collection, lua, cleanup

Implement efficient memory management and garbage collection strategies.

Example: Example (Neovim)

```
-- 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
})
```

51.11 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: Example (Neovim)

```
-- 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
```

51.12 Optimize plugin loading strategy

Category: Performance Optimization Advanced
Tags: plugin, loading, lazy, startup, optimization

Implement sophisticated plugin loading strategies for minimal startup time.

Example: Example (Neovim)

```
-- 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
```

51.13 Plugin configuration caching

Category: Performance Optimization Advanced
Tags: cache, config, plugin, startup, optimization

Implement configuration caching for faster plugin loading.

Example: Example (Neovim)

```
-- 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
```

51.14 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: Example (Neovim)

```
-- 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)
```

51.15 Startup time profiling and analysis

Category: Performance Optimization Advanced
Tags: profile, startup, analysis, benchmark, timing

Implement comprehensive startup profiling and performance analysis.

Example: Example (Neovim)

```
-- 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, {})
```

51.16 Syntax and highlighting optimization

Category: Performance Optimization Advanced
Tags: syntax, highlight, treesitter, performance

Optimize syntax highlighting for better performance on large files.

Example: Example (Neovim)

```
-- 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 52

Plugins

52.1 Add Version Info to Plugins

Category: plugins
Tags: plugin-development, version-tracking, scripting

Instead of using a boolean flag to prevent multiple plugin loads, set a version number that can be checked by other plugins

Example: Example (Vim)

```
if exists('g:loaded_dbext') || &cp
  finish
endif
let g:loaded_dbext = 503
```

Example: Example (Neovim)

```
if vim.g.loaded_dbext or vim.o.compatible then
  return
end
vim.g.loaded_dbext = 503
```

Source: vim.fandom.com

52.2 Advanced File and Buffer Selection Plugins

Category: plugins
Tags: file-navigation, buffer-management, productivity

Several powerful plugins provide advanced UI elements for selecting buffers, files, and recent documents with fuzzy finding capabilities

Example: Example (Vim)

```
" Recommended plugins for enhanced file/buffer selection
" Command-T, CtrlP, FuzzyFinder, unite
```

Example: Example (Neovim)

```
-- Recommended Neovim plugins for file/buffer selection
-- telescope.nvim (modern replacement for these plugins)
require('telescope.builtin').find_files()
require('telescope.builtin').buffers()
```

Source: vim.fandom.com

52.3 Advanced Matching with matchit Plugin

Category: plugins
Tags: syntax-navigation, language-support

Extend % key to match words, language-specific constructs, and more complex syntax elements

Example: Example (Vim)

```
" Enable matchit plugin
runtime macros/matchit.vim
```

Example: Example (Neovim)

```
-- For matchit, use vim.cmd to load plugin
vim.cmd('runtime macros/matchit.vim')
```

Source: vim.fandom.com

52.4 Advanced UI Selection Plugins for Vim/Neovim

Category: plugins
Tags: fuzzy-finder, ui-enhancement, file-navigation

Several powerful plugins provide advanced item selection interfaces for improved workflow in Vim and Neovim

Example: Example (Vim)

```
" Notable plugins for advanced UI selection
" - CtrlP
" - FuzzyFinder
" - unite.vim
```

Example: Example (Neovim)

```
-- Recommended modern Neovim alternatives
-- telescope.nvim
-- fzf-lua
-- which-key.nvim
```

Source: vim.fandom.com

52.5 Asynchronous Linting with ALE

Category: plugins
Tags: linting, async, code-quality

Use Asynchronous Lint Engine (ALE) for real-time, non-blocking linting across multiple languages

Example: Example (Vim)

```
" ALE configuration is typically done in .vimrc
let g:ale_linters = {
  \ 'python': ['pylint', 'flake8'],
  \ 'javascript': ['eslint']
  \ }
```

Example: Example (Neovim)

```
-- ALE configuration in Lua
vim.g.ale_linters = {
  python = {'pylint', 'flake8'},
  javascript = {'eslint'}
}
```

Source: vim.fandom.com

52.6 Auto-Generate Project Tags with Makefile

Category: plugins
Tags: tags, project-management, ctags

Automatically generate ctags for a project using a Makefile and Project.vim plugin, ensuring tags are always up-to-date

Example: Example (Vim)

```
" In .vimprojects file
example=d:/example {
  project=project CD=. in=maketags.vim filter="*.c *.h" {
    source/src1.c
```

```
    source/src2.c
  }
}
```

Example: Example (Neovim)

```
-- Equivalent project configuration in Lua
-- Would typically be managed through a plugin like project.nvim
-- Create a similar workflow using autocmds or a custom function
require('project_nvim').setup({
  detection_methods = { 'pattern' },
  patterns = { '.git', 'Makefile', 'tags' }
})
```

Source: vim.fandom.com

52.7 Auto-Source Plugins in Vim

Category: plugins
Tags: plugin-management, configuration, vim-compatibility

Automatically source all Vim script plugins from a specific directory, mimicking Vim 6.0's plugin feature for older versions

Example: Example (Vim)

```
exec "source " . substitute(glob($VIM."/plugins/*.vim"), "\n", "\nsource ",
↪  "g")
```

Example: Example (Neovim)

```
-- Load all .vim plugins from a specific directory
local plugin_dir = vim.fn.expand('$VIM/plugins/*.vim')
for _, plugin in ipairs(vim.fn.glob(plugin_dir, false, true)) do
  vim.cmd('source ' .. plugin)
end
```

Source: vim.fandom.com

52.8 Automatically Generate PHPDoc Comments

Category: plugins
Tags: documentation, php, automation

Use JCommenter.vim to automatically generate documentation comments for PHP functions, classes, and variables

Example: Example (Vim)

```
call JCommentWriter()
```

Example: Example (Neovim)

```
-- Note: This requires installing the JCommenter.vim plugin
-- Equivalent functionality would need to be implemented in a Lua plugin
-- Potential Neovim approach using treesitter or custom function
function GenerateDocComment()
  -- Analyze function/class structure
  -- Generate appropriate documentation template
end
```

Source: vim.fandom.com

52.9 C++ Code Completion with OmniCppComplete

Category: plugins
Tags: code-completion, c++, omnicomplete

Set up advanced C++ code completion using OmniCppComplete with ctags integration for improved intellisense

Example: Example (Vim)

```
" OmniCppComplete configuration
let OmniCpp_NamespaceSearch = 1
let OmniCpp_GlobalScopeSearch = 1
let OmniCpp_ShowAccess = 1
let OmniCpp_ShowPrototypeInAbbr = 1
let OmniCpp_MayCompleteDot = 1
let OmniCpp_MayCompleteArrow = 1
let OmniCpp_MayCompleteScope = 1
let OmniCpp_DefaultNamespaces = ["std", "_GLIBCXX_STD"]

" Automatically close popup menu
au CursorMovedI,InsertLeave * if pumvisible() == 0|silent! pclose|endif
set completeopt=menuone,menu,longest,preview
```

Example: Example (Neovim)

```
-- OmniCppComplete configuration in Neovim
vim.g.OmniCpp_NamespaceSearch = 1
vim.g.OmniCpp_GlobalScopeSearch = 1
vim.g.OmniCpp_ShowAccess = 1
vim.g.OmniCpp_ShowPrototypeInAbbr = 1
vim.g.OmniCpp_MayCompleteDot = 1
vim.g.OmniCpp_MayCompleteArrow = 1
vim.g.OmniCpp_MayCompleteScope = 1
vim.g.OmniCpp_DefaultNamespaces = {"std", "_GLIBCXX_STD"}
```

```
-- Automatically close popup menu
vim.api.nvim_create_autocmd({"CursorMovedI", "InsertLeave"}, {
  callback = function()
    if vim.fn.pumvisible() == 0 then
      vim.cmd('silent! pclose')
    end
  end
})

-- Set completion options
vim.opt.completeopt = {"menuone", "menu", "longest", "preview"}
```

Source: vim.fandom.com

52.10 Configure SVN Integration with VCSCommand

Category: plugins
Tags: version-control, svn, integration

Configure VCSCommand plugin to work with TortoiseSVN checkouts, especially for non-standard SSH configurations

Example: Example (Vim)

```
let VCSCommandSVNExec="C:\\Progra~1\\TortoiseSVN\\bin\\svn.exe"
let VCSCommandVCSTypeOverride= [['C:\\working\\*', 'SVN']]
```

Example: Example (Neovim)

```
vim.g.VCSCommandSVNExec = "C:\\Progra~1\\TortoiseSVN\\bin\\svn.exe"
vim.g.VCSCommandVCSTypeOverride = {{"C:\\working\\*", "SVN"}}
```

Source: vim.fandom.com

52.11 Configure TagList for ANT Build File Navigation

Category: plugins
Tags: taglist, navigation, xml

Enhance navigation in ANT build files by configuring TagList to recognize project, target, and property tags

Example: Example (Vim)

```
let g:tlist_ant_settings = 'ant;p:Project;t:Target;r:Property'
```

Example: Example (Neovim)

```
vim.g.tlist_ant_settings = 'ant;p:Project;t:Target;r:Property'
```

Source: vim.fandom.com

52.12 Create Custom Calendar with Special Dates

Category: plugins
Tags: calendar, custom-dates, diary

Enhance Vim calendar script to display special dates with custom signs and colors from a dedicated holidays file

Example: Example (Vim)

```
let calendar_sign = 'MyGetSpecialDay'
function! MyGetSpecialDay(day, month, year)
  let l:m100d = 10000 + (a:month * 100 ) + a:day
  let l:holidays = expand(g:calendar_diary) . "/holidays"
  exe "split " . l:holidays
  let l:found = search(l:m100d)
  if l:found
    let l:found = 'h'
  endif
  quit
  return l:found
endfunction
```

Example: Example (Neovim)

```
vim.g.calendar_sign = function(day, month, year)
  local m100d = 10000 + (month * 100) + day
  local holidays_path = vim.fn.expand(vim.g.calendar_diary or '~/diary') ..
  ↪  "/holidays"

  local file = io.open(holidays_path, 'r')
  if file then
    for line in file:lines() do
      if line:match(tostring(m100d)) then
        file:close()
        return 'h'
      end
    end
    file:close()
  end

  return nil
end
```

Source: vim.fandom.com

52.13 Create Custom Commands with Autocompletion

Category: plugins
Tags: custom-commands, autocompletion, scripting

Create user-defined commands with range support, parameters, and autocompletion to simplify complex operations

Example: Example (Vim)

```
" Define a custom command with range and file completion
command! -range=% -nargs=1 -complete=file MyCommand <line1>,<line2>call
 ↪  s:MyCommandFunction(<f-args>)

function! s:MyCommandFunction(...) range
  split
  execute "norm! " . a:firstline . "GV"
  execute "norm! " . a:lastline . 'G"ay'
  enew
  norm! "ap
  exe "sav! " . a:1
  q
endfunction
```

Example: Example (Neovim)

```
-- Lua equivalent for creating custom commands
vim.api.nvim_create_user_command('MyCommand', function(opts)
  local start_line = opts.line1
  local end_line = opts.line2
  local filename = opts.args

  -- Similar functionality implemented using Neovim Lua APIs
  vim.cmd(start_line .. ',' .. end_line .. 'yank')
  vim.cmd('new')
  vim.cmd('put')
  vim.cmd('write! ' .. filename)
  vim.cmd('bdelete')
end, {
  range = true,
  nargs = 1,
  complete = 'file'
})
```

Source: vim.fandom.com

52.14 Cross-Platform LaTeX Editing Plugins

Category: plugins
Tags: latex, development-workflow, productivity

Recommended plugins for seamless LaTeX editing with forward and backward search capabilities

Example: Example (Neovim)

```
-- Recommended Plugins:
-- 1. Automatic LaTeX Plugin (ATP)
-- 2. LaTeX Suite
-- Both support cross-platform forward and backward searching
```

Source: vim.fandom.com

52.15 Debug Mode for Plugin Development

Category: plugins
Tags: debugging, plugin-development, workflow

Add a global debug variable to control plugin loading during development

Example: Example (Vim)

```
if exists("plugin_name_loaded") && !exists("g:plugin_name_debug_mode")
  finish
endif
let plugin_name_loaded = 1
```

Example: Example (Neovim)

```
if vim.g.plugin_name_loaded and not vim.g.plugin_name_debug_mode then
  return
end
vim.g.plugin_name_loaded = true
```

Source: vim.fandom.com

52.16 Dynamic Plugin Version Selection

Category: plugins
Tags: plugin-management, version-control, runtime

Automatically select the correct plugin version at runtime using v:version check

Example: Example (Vim)

```
if v:version >= 700
  runtime! plugin/matchit.vim
endif
```

Example: Example (Neovim)

```
-- Lua version for dynamic plugin loading
if vim.version().major >= 7 then
  vim.cmd('runtime! plugin/matchit.vim')
end
```

Source: vim.fandom.com

52.17 Easily Create LaTeX Tables with Vim

Category: plugins
Tags: latex, text-processing, external-tool

Use Perl LaTeX::Table module to quickly convert text tables into formatted LaTeX table code

Example: Example (Vim)

```
# Select lines and run external filter
:'<,'>!ltpretty
```

Example: Example (Neovim)

```
-- Requires external Perl module
-- Select lines and use vim.fn.system() to run ltpretty filter
-- Example:
local selected_text = vim.fn.getline(vim.fn.line("'<"), vim.fn.line("'>"))
local latex_table = vim.fn.system('ltpretty', table.concat(selected_text,
 ↪  "\n"))
```

Source: vim.fandom.com

52.18 Easily Extend Vim Functionality with Plugins

Category: plugins
Tags: plugin-management, vim-extension, configuration

Plugins can extend Vim's functionality by loading script files automatically when Vim starts. Use a plugin manager to simplify installation and management.

Example: Example (Vim)

```
" Use a plugin manager like vim-plug
call plug#begin('~/.vim/plugged')
Plug 'plugin-name'
call plug#end()
```

Example: Example (Neovim)

```
-- Use a plugin manager like lazy.nvim
require('lazy').setup({
  'plugin-name'
})
```

Source: vim.fandom.com

52.19 Flexible Ctags Generation for Multi-Directory Projects

Category: plugins
Tags: ctags, project-configuration, build-tools

Create a flexible Makefile to generate ctags across multiple project directories

Example: Example (Vim)

```
" Makefile for tags generation
DIRS := source include common
SRC := $(foreach dir,$(DIRS),$(wildcard $(dir)/*.[ch]))

CTAGS_FLAGS=--c++-kinds=+p --fields=+imaS --extra=+q

tags: $(SRC)
        ctags $(CTAGS_FLAGS) $(SRC)
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.fn and vim.system
local function generate_ctags()
  local dirs = { 'source', 'include', 'common' }
  local files = {}

  for _, dir in ipairs(dirs) do
    local pattern = string.format('%s/*.[ch]', dir)
    vim.list_extend(files, vim.fn.glob(pattern, false, true))
  end

  vim.system({
    'ctags',
    '--c++-kinds=+p',
    '--fields=+imaS',
    '--extra=+q',
    unpack(files)
  })
end

-- Can be bound to a command or autocmd
vim.api.nvim_create_user_command('GenerateTags', generate_ctags, {})
```

Source: vim.fandom.com

52.20 Generate Ctags for C/C++ Project with Dependencies

Category: plugins
Tags: ctags, development, code-navigation

Generate a comprehensive ctags file for C/C++ projects that includes symbols from source files and their included headers

Example: Example (Vim)

```
#!/bin/sh
gcc -M $* | sed -e 's/[\ ]/\n/g' | \
        sed -e '/^$/d' -e '/\.o:[ \t]*$/d' | ctags -L - --c++-kinds=+p
        ↪  --fields=+iaS --extra=+q
```

Example: Example (Neovim)

```
-- Lua equivalent (shell script, can be integrated with vim.fn.system())
local function generate_ctags(files)
  local cmd = string.format(
    'gcc -M %s | sed -e "s/[\\ ]/\n/g" | sed -e "/^$/d" -e "/\.o:[ \t]*$/d"
    ↪  | ctags -L - --c++-kinds=+p --fields=+iaS --extra=+q',
    table.concat(files, ' ')
  )
  vim.fn.system(cmd)
end

-- Usage example
generate_ctags({'file1.cpp', 'file2.c', 'file3.cpp'})
```

Source: vim.fandom.com)

52.21 Initialize Plugins Using VimEnter Autocmd

Category: plugins
Tags: plugin-initialization, autocmd, configuration

Safely initialize plugins after all startup processes complete using VimEnter autocmd, ensuring plugins are loaded before running initialization commands

Example: Example (Vim)

```
function! g:LoadPluginScript()
    if exists(":Tabularize")
        vnoremap <Leader>t& :Tabularize /&<CR>
        vnoremap <Leader>t| :Tabularize /|<CR>
    endif
endfunction

augroup plugin_initialize
```

```
    autocmd!
    autocmd VimEnter * call LoadPluginScript()
augroup END
```

Example: Example (Neovim)

```
local function load_plugin_script()
    if vim.fn.exists(':Tabularize') == 1 then
        vim.keymap.set('v', '<Leader>t&', ':Tabularize /&<CR>', { noremap =
        ↪  true })
        vim.keymap.set('v', '<Leader>t|', ':Tabularize /|<CR>', { noremap =
        ↪  true })
    end
end

vim.api.nvim_create_augroup('plugin_initialize', { clear = true })
vim.api.nvim_create_autocmd('VimEnter', {
    group = 'plugin_initialize',
    callback = load_plugin_script
})
```

Source: vim.fandom.com

52.22 Integrate Python Linters in Vim/Neovim

Category: plugins
Tags: python, linting, error-checking

Adds a function to run Pylint or Pychecker and display errors in a quickfix window, making it easy to navigate Python code issues

Example: Example (Vim)

```
function <SID>PythonGrep(tool)
  set lazyredraw
  cclose
  let &grepformat = '%f:%l:%m'
  if a:tool == "pylint"
    let &grepprg = 'pylint --output-format=parseable --reports=n'
  elseif a:tool == "pychecker"
    let &grepprg = 'pychecker --quiet -q'
  endif
  silent! grep! %
  belowright copen
  nnoremap <buffer> <silent> c :cclose<CR>
endfunction
```

Example: Example (Neovim)

```
function _G.python_lint(tool)
  vim.o.lazyredraw = true
  vim.cmd('cclose')

  local tools = {
    pylint = 'pylint --output-format=parseable --reports=n',
    pychecker = 'pychecker --quiet -q'
  }

  if not tools[tool] then
    vim.notify('Unknown linting tool', vim.log.levels.ERROR)
    return
  end

  vim.o.grepformat = '%f:%l:%m'
  vim.o.grepprg = tools[tool]

  vim.cmd('silent! grep! %')
  vim.cmd('belowright copen')

  vim.keymap.set('n', 'c', ':cclose<CR>', {buffer = true, silent = true})

  vim.o.lazyredraw = false
  vim.cmd('redraw!')
end

-- Map keys (example)
vim.keymap.set('n', '<F3>', function() _G.python_lint('pylint') end)
vim.keymap.set('n', '<F4>', function() _G.python_lint('pychecker') end)
```

Source: vim.fandom.com

52.23 LaTeX Backward Search Across Platforms

Category: plugins
Tags: latex, pdf-integration, development-workflow

Enables syncing between LaTeX source code and PDF viewer, allowing click-to-jump functionality across different operating systems

Example: Example (Neovim)

```
-- Recommended plugins for LaTeX backward search
-- Linux: Use Automatic LaTeX Plugin (ATP)
-- Mac: Use gvim-pdfsync
-- Requires additional PDF viewer configuration
```

Source: vim.fandom.com

52.24 LaTeX Backward Search Setup

Category: plugins
Tags: latex, pdf-integration, productivity

Enable backward search in LaTeX workflows, allowing you to click in a PDF and jump to the corresponding source location in Vim

Example: Example (Neovim)

```
-- Recommended plugins for LaTeX backward search
-- Use vim-plug or your preferred plugin manager
-- Plugins like 'Automatic LaTeX Plugin' or 'LaTeX Suite' support this
↪  feature

-- Example configuration (pseudo-code)
-- Requires additional setup with specific PDF viewers like Evince, Okular,
↪  or PDFView
```

Source: vim.fandom.com

52.25 Manage Plugins with Plugin Managers

Category: plugins
Tags: plugin-management, workflow, configuration

Plugin managers simplify the process of installing, updating, and managing Vim/Neovim plugins by providing a centralized configuration method.

Example: Example (Vim)

```
" Popular plugin managers:
" - vim-plug
" - Vundle
" - pathogen.vim
```

Example: Example (Neovim)

```
-- Popular Neovim plugin managers:
-- - lazy.nvim (recommended)
-- - packer.nvim
-- - nvim-pluginmanager
```

Source: vim.fandom.com

52.26 Native Vim Plugin Management

Category: plugins
Tags: plugin-management, configuration, vim-packages

Vim now includes native package management, allowing easier plugin installation and organization without external managers

Example: Example (Vim)

```
" Use native Vim package management
" Place plugins in ~/.vim/pack/vendor/start/
```

Example: Example (Neovim)

```
-- Neovim supports native package management
-- Place plugins in ~/.local/share/nvim/site/pack/vendor/start/
-- No additional configuration required
```

Source: vim.fandom.com

52.27 Optimize Startup by Delaying Plugin Loading

Category: plugins
Tags: performance, plugin-management, lazy-loading

Improve Vim startup performance by delaying non-critical plugin function calls

Example: Example (Vim)

```
" Recommended to use autoload or lazy loading techniques
```

Example: Example (Neovim)

```
-- Use a plugin manager like lazy.nvim for efficient plugin loading
require('lazy').setup({
  {'plugin/name', lazy = true}
})
```

Source: vim.fandom.com

52.28 Plugin Initialization in After Plugin Directory

Category: plugins
Tags: plugin-management, configuration, initialization

Initialize plugins by creating a similarly named file in the after/plugin directory, allowing

post-plugin-load configuration

Example: Example (Vim)

```
" In .vim/after/plugin/HiMtchBrkt.vim
norm \[i
```

Example: Example (Neovim)

```
-- In ~/.config/nvim/after/plugin/HiMtchBrkt.lua
vim.cmd('norm \[i')
```

Source: vim.fandom.com

52.29 Plugin Manager Alternatives

Category: plugins
Tags: plugin-management, nvim-plugins

Multiple modern plugin managers for Neovim, with packer.nvim and user.nvim being recommended for Neovim users

Example: Example (Neovim)

```
-- Recommended plugin managers for Neovim
-- packer.nvim: https://github.com/wbthomason/packer.nvim
-- user.nvim: https://github.com/faerryn/user.nvim
```

Source: vim.fandom.com

52.30 Reload Plugin During Development

Category: plugins
Tags: plugin-development, debugging, workflow

Simple method to reload buffer-specific files and test plugin changes by using :e command

Example: Example (Vim)

```
" Reload buffer to test plugin changes
:e
```

Example: Example (Neovim)

```
-- Reload current buffer to trigger plugin reloading
vim.cmd('edit')
```

Source: vim.fandom.com

52.31 Simplify Omni-Completion with SuperTab

Category: plugins
Tags: completion, tab-completion, productivity

Use SuperTab to invoke omni-completion more easily with Tab key

Example: Example (Vim)

```
" Use Tab for context-aware completion
let g:SuperTabDefaultCompletionType = "context"
```

Example: Example (Neovim)

```
-- Configure SuperTab for context-aware completion
vim.g.SuperTabDefaultCompletionType = "context"
```

Source: vim.fandom.com

52.32 Use Compiler Plugins for Linting

Category: plugins
Tags: linting, syntax-checking, compiler

Utilize Vim's compiler plugins to easily set up linters for different file types with minimal configuration

Example: Example (Vim)

```
" Set pylint as compiler for Python files
autocmd FileType python compiler pylint

" Check current file
:make %

" Check all Python files in directory
:make *.py
```

Example: Example (Neovim)

```
-- Set up pylint for Python files in Lua
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.cmd('compiler pylint')
```

```
  end
})

-- You can still use :make commands in Neovim
```

Source: vim.fandom.com

52.33 Use External Editor with Thunderbird

Category: plugins
Tags: email, external-editor, integration

Configure Vim as an external editor for Thunderbird, allowing email composition in Vim

Example: Example (Vim)

```
gvim.exe "+set ft=mail"
```

Example: Example (Neovim)

```
-- Configure external editor for Thunderbird
-- Use gvim with mail filetype
vim.g.thunderbird_external_editor = 'gvim --nofork "+set ft=mail"'
```

Source: vim.fandom.com

52.34 Use Vim as External Editor for Web Textareas

Category: plugins
Tags: browser-integration, external-editor, productivity

Use the Mozex plugin to edit web textareas and view source code directly in Vim/Gvim

Example: Example (Vim)

```
" Mozex configuration in user.js
user_pref("mozex.command.textarea", "C:\\vim\\gvim.exe %t");
user_pref("mozex.command.source", "C:\\vim\\gvim.exe %t");
```

Example: Example (Neovim)

```
-- Lua equivalent (configuration would be browser-specific)
-- Requires Mozex or similar browser extension
-- Example setup for reference
local mozex_config = {
  textarea_command = vim.fn.expand('~') .. '/bin/gvim %t',
  source_command = vim.fn.expand('~') .. '/bin/gvim %t'
```

```
}
```

Source: vim.fandom.com

CHAPTER 53

Registers

53.1 Advanced Copy and Paste with Named Registers

Category: registers
Tags: copy, paste, registers, clipboard

Use named registers to store and recall multiple pieces of text across different operations

Example: Example (Vim)

```
"ay  " copy text to register a
"by  " copy text to register b
"ap  " paste from register a after cursor
"bP  " paste from register b before cursor
```

Example: Example (Neovim)

```
-- Lua equivalents use vim.fn.setreg() and vim.fn.getreg()
-- Copy to register a
vim.fn.setreg('a', 'text to copy')

-- Paste from register a
vim.cmd('normal! "ap')  -- paste after cursor
vim.cmd('normal! "aP')  -- paste before cursor
```

Source: vim.fandom.com

53.2 Advanced Text Manipulation with Registers

Category: registers
Tags: copy-paste, text-manipulation, buffers

Use named registers for more flexible text copying and pasting

Example: Example (Neovim)

```
-- Register manipulation
vim.keymap.set('n', 'y', 'y', { desc = 'Yank (copy) text' })
```

```
vim.keymap.set('n', 'p', 'p', { desc = 'Paste after cursor' })
vim.keymap.set('n', 'P', 'P', { desc = 'Paste before cursor' })
-- Named register example
vim.keymap.set('n', '"ay5dd', '"ay5dd', { desc = 'Delete 5 lines to register
↪  a' })
```

Source: vim.fandom.com

53.3 Append Multiple Lines to a Register

Category: registers
Tags: text-manipulation, advanced-editing

Technique for collecting multiple lines into a single register by using uppercase register names

Example: Example (Vim)

```
"add    " Delete first line to register a
"Add    " Append next line to register a
"ap     " Paste contents of register a
```

Example: Example (Neovim)

```
-- Manual register manipulation can be done in Lua via vim.fn.setreg()
-- Example of appending to a register
vim.fn.setreg('A', vim.fn.getline('.'), 'l')
```

Source: vim.fandom.com

53.4 Append or Modify Existing Macros

Category: registers
Tags: macros, editing-tricks

Easily extend an existing macro without re-recording the entire sequence

Example: Example (Vim)

```
" Original macro in lowercase register
qa...q
" Append to macro using uppercase register
qA...q
```

Example: Example (Neovim)

```
-- Same approach works in Neovim
-- First macro: vim.fn.setreg('a', 'original_macro_commands')
-- Append to macro: vim.fn.setreg('A', 'additional_commands', 'a')
```

Source: vim.fandom.com

53.5 Append to Existing Macros

Category: registers
Tags: macro, editing, productivity

Easily extend an existing macro without re-recording the entire sequence

Example: Example (Vim)

```
" Initial macro recording
qa
... commands ...
q

" Append to existing macro
qA
... additional commands ...
q
```

Example: Example (Neovim)

```
-- Note: In Neovim, you can still use vim command mode for macro recording
-- Use uppercase register letter to append to existing macro
-- For more advanced macro manipulation, consider using vim.fn.setreg()
```

Source: vim.fandom.com

53.6 Append to Macro Registers Safely

Category: registers
Tags: macro-recording, register-manipulation

Use uppercase register names to safely append to existing macros without overwriting previous content

Example: Example (Vim)

```
" Append to 'q' register
qQ    " Start appending to register q
@q    " Execute existing macro
q     " Stop recording
```

Example: Example (Neovim)

```
-- Lua macro appending (still uses Vim commands)
-- vim.cmd('qQ')   " Start appending
-- vim.cmd('@q')   " Execute existing macro
-- vim.cmd('q')    " Stop recording
```

Source: vim.fandom.com

53.7 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: Example (Vim)

```
qAq    " clear register 'A'
qaq    " clear register 'a'
q1q    " clear register '1'
q:q    " clear command register
```

Source: Community contributed

53.8 Create Recursive Macros for Bulk Text Manipulation

Category: registers
Tags: macro, text-processing, automation

Record a macro that calls itself to perform repetitive transformations across an entire file, allowing complex text editing without complex regex

Example: Example (Vim)

```
" Record recursive macro steps
qqq   " Clear register q
qq    " Start recording
f2    " Find '2' character
D     " Delete rest of line
Fr    " Find last 'r'
p     " Paste deleted text
j     " Move to next line
@q    " Execute macro recursively
q     " Stop recording
```

Example: Example (Neovim)

```
-- Lua equivalent requires manual implementation
-- Use vim.fn.execute() for macro playback
-- Recommended to use :normal for similar tasks in Neovim
```

Source: vim.fandom.com

53.9 Cycle Through Numbered Registers

Category: registers
Tags: navigation, text-recovery, undo

Quickly cycle through previous yanks/deletes using repeat and undo commands

Example: Example (Vim)

```
"1p....  " Paste and cycle through registers
u.     " Undo last paste and paste next register
```

Example: Example (Neovim)

```
-- Note: Direct Lua equivalent requires more complex implementation
-- Recommended to use Vim commands or create a custom function
vim.cmd('"1p....')
vim.cmd('u.')
```

Source: vim.fandom.com

53.10 Cycle Through Vim Registers Easily

Category: registers
Tags: register-management, key-mapping, workflow

Create a mapping to easily swap and cycle through different registers, making register manipulation more convenient

Example: Example (Vim)

```
" Cycle registers: unnamed, a, b
nnoremap <Leader>s :let @x=@" | let @"=@a | let @a=@b | let @b=@x<CR>
```

Example: Example (Neovim)

```
-- Cycle registers in Neovim
vim.keymap.set('n', '<Leader>s', function()
  local x = vim.fn.getreg('"')
  vim.fn.setreg('"', vim.fn.getreg('a'))
```

```
  vim.fn.setreg('a', vim.fn.getreg('b'))
  vim.fn.setreg('b', x)
end, { desc = 'Cycle through registers' })
```

Source: vim.fandom.com

53.11 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: Example (Vim)

```
"_d  " delete to blackhole register
```

Source: Community contributed

53.12 Easy Word Replacement Using Registers

Category: registers
Tags: editing, registers, clipboard

Efficiently replace words using Vim registers without losing the original copied content

Example: Example (Vim)

```
" Remap visual mode paste to preserve original register
xnoremap <silent> p p:let @"=@0<CR>
```

Example: Example (Neovim)

```
-- Lua equivalent for preserving register during paste
vim.keymap.set('x', 'p', function()
  vim.fn.feedkeys('p', 'n')
  vim.fn.setreg('"', vim.fn.getreg('0'))
end, { noremap = true, silent = true })
```

Source: vim.fandom.com

53.13 Edit and Modify Recorded Macros

Category: registers
Tags: macro-editing, register-manipulation

Multiple ways to edit and modify recorded macros using registers

Example: Example (Vim)

```
qxq     # Clear register x
:let @x = ''  # Clear register x

# Edit macro by pasting into buffer
:new
"xp     # Paste macro
[edit]
0"xy$   # Yank modified macro back
```

Example: Example (Neovim)

```
-- Clear a register
vim.fn.setreg('x', '')

-- Edit macro programmatically
-- Create a new buffer
vim.cmd('new')

-- Paste macro content
vim.fn.setreg('"', vim.fn.getreg('x'))
vim.cmd('normal! p')

-- After editing, you can save back to register
vim.fn.setreg('x', vim.fn.getline(1))
```

Source: vim.fandom.com

53.14 Execute Macro Across Visual Block Selection

Category: registers
Tags: macros, visual-block, advanced-editing

Executes a macro stored in register 'a' across all lines in a visual block selection, enabling quick multi-line transformations

Example: Example (Vim)

```
" make ` execute the contents of the a register across visual selection
vnoremap ` :normal @a<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '`', ':normal @a<CR>', { desc = 'Execute macro from
↪  register a on visual block lines' })
```

Source: vim.fandom.com

53.15 Get Current File Name in Different Formats

Category: registers
Tags: file-operations, command-line, registers

Multiple ways to extract and display the current file's name, path, or components using expand() function and registers

Example: Example (Vim)

```
:echo @%           " Relative file path
:echo expand('%:t') " Filename only
:echo expand('%:p') " Full file path
:echo expand('%:p:h') " Directory containing file
:echo expand('%:r')  " Filename without extension
```

Example: Example (Neovim)

```
-- Get current file details
print(vim.fn.expand('%'))      -- Relative file path
print(vim.fn.expand('%:t'))    -- Filename only
print(vim.fn.expand('%:p'))    -- Full file path
print(vim.fn.expand('%:p:h'))  -- Directory containing file
print(vim.fn.expand('%:r'))    -- Filename without extension
```

Source: vim.fandom.com

53.16 Get Current Filename in Vim

Category: registers
Tags: filename, buffer, expansion

Multiple ways to extract and manipulate the current filename using registers and expand() function

Example: Example (Vim)

```
" Get filename relative to current directory
:echo @%
" Get just the filename
:echo expand('%:t')
" Get full path
:echo expand('%:p')
" Get directory of current file
:echo expand('%:p:h')
```

Example: Example (Neovim)

```
-- Get filename relative to current directory
print(vim.fn.expand('%'))
-- Get just the filename
print(vim.fn.expand('%:t'))
-- Get full path
print(vim.fn.expand('%:p'))
-- Get directory of current file
print(vim.fn.expand('%:p:h'))
```

Source: vim.fandom.com

53.17 Insert Register Content Across Multiple Lines

Category: registers
Tags: register-manipulation, multi-line-editing, clipboard

Insert contents of a register (like clipboard) across multiple lines in visual block mode

Example: Example (Vim)

```
" In visual block mode:
" I Ctrl-R + - Insert clipboard content
" I Ctrl-R " - Insert unnamed register
```

Example: Example (Neovim)

```
-- Same approach works in Neovim
-- Use Ctrl-V to select block
-- I Ctrl-R + to insert from clipboard
-- I Ctrl-R " to insert from unnamed register
```

Source: vim.fandom.com

53.18 Paste Over Text Without Losing Register

Category: registers
Tags: paste, text-replacement, key-mapping

Create a mapping to paste over text while preserving the original yank register, preventing unwanted register overwrite

Example: Example (Vim)

```
xnoremap <silent> p p:let @"=@0<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('x', 'p', function()
  vim.cmd('normal! p')
  vim.fn.setreg('"', vim.fn.getreg('0'))
end, { silent = true })
```

Source: vim.fandom.com

53.19 Paste Registers in Command and Search Lines

Category: registers
Tags: command-line, search, registers

Quickly insert register contents into command or search lines using Ctrl-R, avoiding clipboard operations

Example: Example (Vim)

```
" In command or search mode, press Ctrl-R then register name
" Example: Ctrl-R a to paste contents of register a
" Ctrl-R " to paste unnamed register
```

Example: Example (Neovim)

```
-- In Neovim, this works the same way in command and search modes
-- Use <C-r> followed by register name in command or search prompt
-- Example in Lua: No direct translation needed, built-in functionality
```

Source: vim.fandom.com

53.20 Paste Text as Separate Line with Easy Mapping

Category: registers
Tags: pasting, register, key-mapping

Easily paste text from the default register as a separate line above or below the current line

Example: Example (Vim)

```
nnoremap ,p :put "<CR>
nnoremap ,P :put! "<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', ',p', ':put "<CR>', { desc = 'Paste below current line'
↪  })
vim.keymap.set('n', ',P', ':put! "<CR>', { desc = 'Paste above current line'
↪  })
```

Source: vim.fandom.com

53.21 Paste from Clipboard with Precise Cursor Placement

Category: registers
Tags: clipboard, cursor-movement, advanced-paste

Paste from clipboard before cursor and set marks for the pasted text

Example: Example (Vim)

```
"+P
```

Example: Example (Neovim)

```
vim.api.nvim_put(vim.fn.getreg('+'), 'c', true, true)
```

Source: vim.fandom.com

53.22 Paste from Different Vim Registers Safely

Category: registers
Tags: clipboard, register-manipulation, paste

Safely paste contents from various Vim registers in different modes, with special focus on clipboard security

Example: Example (Vim)

```
" Paste from clipboard register in normal mode
"*p

" Paste last search or command
"/p
":p

" Safely paste in insert mode
inoremap <C-R>+ <C-R><C-R>+
```

Example: Example (Neovim)

```
-- Paste from clipboard in normal mode
vim.keymap.set('n', '<leader>p', '"*p', { desc = 'Paste from clipboard' })

-- Safely paste in insert mode
vim.keymap.set('i', '<C-R>+', '<C-R><C-R>+', { desc = 'Safely paste from
↪  clipboard' })
```

Source: vim.fandom.com

53.23 Preload Vim Registers with Frequently Used Commands

Category: registers
Tags: command-macros, productivity, configuration

Preload registers with frequently used command sequences to quickly execute complex commands with a single keystroke

Example: Example (Vim)

```
" Preload registers with common commands
let @m=":'a,'bs/"
let @s=":%!sort -u"

" Interesting recursive yank macro
let @y='yy@"'
```

Example: Example (Neovim)

```
-- Preload registers with common commands
vim.g.registers = {
  m = ":'a,'bs/",  -- Example find/replace between marks
  s = ":%!sort -u",  -- Sort unique lines
  y = 'yy@"'  -- Recursive yank macro
}

-- To use the preloaded register
-- Execute with @m, @s, etc.
```

Source: vim.fandom.com

53.24 Prevent Clipboard Hijacking in Vim

Category: registers
Tags: security, clipboard, insert-mode

Protect against potential malicious clipboard content that could execute unintended commands

Example: Example (Vim)

```
" Prevent clipboard hijacking
inoremap <C-R>+ <C-R><C-R>+
" Alternative safe paste methods
inoremap <C-R>+ <C-R><C-O>+
```

Example: Example (Neovim)

```
-- Prevent clipboard hijacking
vim.keymap.set('i', '<C-R>+', '<C-R><C-R>+', { desc = 'Safe clipboard paste'
↪  })
-- Alternative safe paste method
vim.keymap.set('i', '<C-R>+', '<C-R><C-O>+', { desc = 'Alternative safe
↪  clipboard paste' })
```

Source: vim.fandom.com

53.25 Quick Macro Recording and Playback

Category: registers
Tags: macro, recording, productivity

Simplifies macro recording by using 'q' register and mapping 'Q' to playback

Example: Example (Vim)

```
nnoremap Q @q
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'Q', '@q', { desc = 'Playback macro from q register' })
```

Source: vim.fandom.com

53.26 Record and Execute Vim Macros Efficiently

Category: registers
Tags: macros, productivity, recording, automation

Learn to record, run, and manage Vim macros across multiple lines and registers

Example: Example (Vim)

```
" Record macro to register d
qd
" Your complex commands here
q
```

```
" Execute macro
@d
" Repeat last macro
@@

" Run macro on visual selection
:normal @q
```

Example: Example (Neovim)

```
-- Recording macro to register d
-- Use normal mode mapping to simulate qd
vim.keymap.set('n', 'qd', 'qd', { desc = 'Start recording macro in d
↪  register' })

-- Execute macro
vim.keymap.set('n', '<leader>m', '@d', { desc = 'Execute macro from d
↪  register' })

-- Run macro on visual selection
vim.keymap.set('x', '<leader>m', ':normal @q<CR>', { desc = 'Run macro on
↪  visual selection' })
```

Source: vim.fandom.com

53.27 Record and Replay Macros Efficiently

Category: registers
Tags: macro, automation, recording

Quickly record a sequence of commands to a register and replay them across multiple lines or the entire file

Example: Example (Vim)

```
" Record macro to register d
qd
... commands ...
q

" Replay macro
@d
" Replay on multiple lines
:normal @d
```

Example: Example (Neovim)

```
-- Record macro to register d
-- Use vim.fn.feedkeys() to simulate recording
-- Replay macro
vim.cmd('normal @d')

-- Replay on visual selection
vim.cmd('normal! gv:normal @d<CR>')
```

Source: vim.fandom.com

53.28 Record and Replay Powerful Text Transformation Macros

Category: registers
Tags: macro-recording, text-transformation, automation

Quickly record complex text editing sequences that can be replayed across multiple lines or files

Example: Example (Vim)

```
qd  # Start recording macro to register d
...[editing commands]...
q   # Stop recording
@d  # Replay macro
2@@ # Replay macro twice
```

Example: Example (Neovim)

```
-- Macro recording in Lua is same as Vim
-- Use vim.fn.feedkeys() to simulate macro recording if needed
vim.cmd('qd')  -- Start macro recording
-- Perform editing commands
vim.cmd('q')   -- Stop recording
vim.cmd('@d')  -- Replay macro
```

Source: vim.fandom.com

53.29 Record, Edit, and Reuse Vim Macros

Category: registers
Tags: macros, productivity, automation

Efficiently record, edit, and replay complex editing sequences across multiple files or lines

Example: Example (Vim)

```
" Record macro to register d
qd
" Perform editing commands
q

" Replay macro
@d

" Replay on multiple lines
:normal @q
```

Example: Example (Neovim)

```
-- Note: Macro recording is largely the same in Neovim
-- Record macro: qq (start recording to q register)
-- Stop recording: q
-- Replay macro: vim.fn.feedkeys('@q')

-- Replay macro on visual selection
vim.cmd('normal! gv:normal @q')
```

Source: vim.fandom.com

53.30 Recover Previous Deletes Using Numbered Registers

Category: registers
Tags: yank, delete, clipboard, recovery

Vim keeps previous deletes/yanks in numbered registers, allowing you to recover text even after accidentally overwriting the default register

Example: Example (Vim)

```
" List all registers
:reg

" Paste from register 2
"2p  " Paste after cursor
"2P  " Paste before cursor
```

Example: Example (Neovim)

```
-- List all registers
vim.cmd('reg')

-- Paste from specific numbered register
vim.api.nvim_put(vim.fn.getreg('2'), '', true, true)
```

Source: vim.fandom.com

53.31 Recover Recently Deleted Text in Insert Mode

Category: registers
Tags: text-recovery, registers, insert-mode

Use the '.' register to recover text deleted by Ctrl-U or Ctrl-W

Example: Example (Vim)

```
:let @a = @.
"aP
```

Example: Example (Neovim)

```
vim.fn.setreg('a', vim.fn.getreg('.'))
vim.api.nvim_paste(vim.fn.getreg('a'), true, -1)
```

Source: vim.fandom.com

53.32 Redirect Ex Command Output to Register

Category: registers
Tags: command-output, registers, ex-commands

Capture command output directly to a register for later use or pasting

Example: Example (Vim)

```
:redir @a
:set all
:redir END
" now paste with "ap
```

Example: Example (Neovim)

```
-- Lua doesn't have direct :redir equivalent
-- Use vim.api.nvim_exec() instead
local output = vim.api.nvim_exec('set all', true)
vim.fn.setreg('a', output)
```

Source: vim.fandom.com

53.33 Reorder Lines Using Registers

Category: registers
Tags: line-manipulation, advanced-editing, registers

Efficiently reorder up to nine lines by using numbered registers and the dot command

Example: Example (Vim)

```
" Delete lines in desired final order
" Then paste from numbered registers using dot command
"1P
........
```

Example: Example (Neovim)

```
-- Note: This technique relies on Vim's register mechanism
-- Equivalent in Lua would be to use vim.fn.setreg() and vim.fn.getreg()
-- The specific implementation would mirror the Vimscript approach
```

Source: vim.fandom.com

53.34 Run Macro Across Visual Selection

Category: registers
Tags: macro-execution, bulk-editing, visual-mode

Execute a macro on every line in a visual selection, enabling rapid bulk transformations

Example: Example (Vim)

```
:normal @q  # Run macro from register q on each selected line
```

Example: Example (Neovim)

```
-- Run macro on visual selection
vim.cmd('normal! @q')
```

Source: vim.fandom.com

53.35 Set register manually

Category: Registers
Tags: register, set, manual

Use `:let @a='text'` to manually set the contents of register a.

Example: Example (Vim)

```
:let @a='hello world'  " set register a to 'hello world'
```

Source: Community contributed

53.36 Smart Visual Mode Paste without Register Overwrite

Category: registers
Tags: visual-mode, registers, clipboard

A mapping that allows pasting over selected text without overwriting the current register, preserving the original copied text for multiple pastes

Example: Example (Vim)

```
xnoremap <silent> p p:let @"=@0<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('x', 'p', function()
  vim.fn.feedkeys('p', 'n')
  vim.fn.setreg('"', vim.fn.getreg('0'))
end, { silent = true })
```

Source: vim.fandom.com

53.37 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: Example (Vim)

```
"+y  " yank to system clipboard
"+p  " paste from system clipboard
```

Source: Community contributed

53.38 Translate Null Characters in Vim Registers

Category: registers
Tags: debugging, register-manipulation, string-translation

Use strtrans() to visualize and understand how null characters are represented in Vim registers

Example: Example (Vim)

```
" Translate register contents
:echo strtrans(@a)
```

Example: Example (Neovim)

```
-- Lua equivalent for translating register contents
print(vim.fn.strtrans(vim.fn.getreg('a')))
```

Source: vim.fandom.com

53.39 View registers

Category: Registers
Tags: registers, clipboard, view

Use `:registers` to show the contents of all registers.

Example: Example (Vim)

```
:registers
```

Source: Community contributed

CHAPTER 54

Scripting

54.1 Handle UTF-8 Encoding in Python Vim Scripts

Category: scripting
Tags: python, unicode, encoding

Properly encode Unicode strings when passing them between Python and Vim to avoid encoding errors

Example: Example (Vim)

```
python << EOS
uniStr = u"\u2026"
str = uniStr.encode( vim.eval("&encoding") )
print str
EOS
```

Example: Example (Neovim)

```
-- Lua equivalent for handling UTF-8 encoding
-- Note: Neovim has better Unicode support by default
local uniStr = "…"
local encodedStr = vim.fn.iconv(uniStr, "utf-8", vim.o.encoding)
print(encodedStr)
```

Source: vim.fandom.com

54.2 Prevent Script Reloading with Function

Category: scripting
Tags: plugin-development, script-management, performance

Provides a simple function to prevent reloading of Vim scripts, which helps improve script loading performance and avoid duplicate definitions

Example: Example (Vim)

```
function DontLoadTwice(globalIdentifier)
  let dltcmd ="if exists(\"g:".a:globalIdentifier."\")
"
  let dltcmd.=" finish\n"
  let dltcmd.="endif\n"
  let dltcmd.="let g:".a:globalIdentifier."=1\n"
  return dltcmd
endfunction
```

Example: Example (Neovim)

```
function _G.dont_load_twice(identifier)
  if vim.g[identifier] then
    return
  end
  vim.g[identifier] = true
end
```

Source: vim.fandom.com

CHAPTER 55

Search

55.1 Abbreviated HTML Attribute Quoting

Category: search_replace
Tags: html, abbreviation, text-transformation

Create a command-line abbreviation for quoting HTML attributes with confirmation

Example: Example (Vim)

```
cabbrev reg1 %s/=\([^"][^> ]\{0,40\}\)/="\1"/gc
```

Example: Example (Neovim)

```
vim.cmd('cabbrev reg1 %s/=\([^"][^> ]\{0,40\}\)/="\1"/gc')
```

Source: vim.fandom.com

55.2 Advanced Duplicate Line Removal

Category: search_replace
Tags: text-manipulation, regex

Remove duplicate lines with more complex regex-based methods, preserving line order

Example: Example (Vim)

```
%s/^\(.*\)\(\n\1\)\+$/\1/
```

Example: Example (Neovim)

```
vim.cmd('%s/^\(.*\)(\n\1)+$/\1/')
```

Source: vim.fandom.com

55.3 Advanced Global Command Line Filtering

Category: search_replace
Tags: regex, global-command, line-filtering

Use lookahead-like global command to selectively delete lines based on complex pattern matching

Example: Example (Vim)

```
g#\(^\./help/\)\(de/\)\@!#d
```

Example: Example (Neovim)

```
-- Delete help lines not starting with './help/de/'
vim.api.nvim_exec([[g#\(^\.\/help\/\)\(de\/\)\@!#d]], false)
```

Source: vim.fandom.com

55.4 Advanced Negative Lookahead Searches

Category: search_replace
Tags: regex, advanced-search, pattern-matching

Use regex with negative lookahead to find words or lines excluding specific patterns

Example: Example (Vim)

```
/\<\(foo\>\)\@!\k\+\>

/\(foo.*\)\@<!bar
```

Example: Example (Neovim)

```
-- Find words excluding 'foo'
vim.fn.search('\<\(foo\>\)\@!\k\+\>', 'n')

-- Find 'bar' without preceding 'foo'
vim.fn.search('\(foo.*\)\@<!bar', 'n')
```

Source: vim.fandom.com

55.5 Advanced Pattern Matching with Global Command

Category: search_replace
Tags: regex, text-filtering, pattern-matching

Use \| (or) to filter lines based on multiple patterns, keeping only specific matches

Example: Example (Vim)

```
" Keep only lines with error, warn, or fail
:v/error\|warn\|fail/d
```

Example: Example (Neovim)

```
-- Keep only lines with error, warn, or fail
vim.cmd('v/error\|warn\|fail/d')
```

Source: vim.fandom.com

55.6 Advanced Range Selection Techniques

Category: search_replace
Tags: visual-mode, marks, pattern-matching

Use visual selection, marks, and search patterns to define flexible text editing ranges

Example: Example (Vim)

```
" Select paragraph and substitute
vip:s/old/new/g

" Using marks for range selection
ma  " Mark start of range
mb  " Mark end of range
:'a,'bs/old/new/g  " Substitute between marks
```

Example: Example (Neovim)

```
-- Similar approaches in Neovim
-- Visual paragraph selection and substitution
vim.cmd('normal vips/old/new/g')

-- Using marks for range selection
vim.cmd('ma')   -- Mark start of range
vim.cmd('mb')   -- Mark end of range
vim.cmd("'a,'bs/old/new/g")  -- Substitute between marks
```

Source: vim.fandom.com

55.7 Advanced Search and Replace with Confirmation

Category: search_replace
Tags: search, substitution, global-command

Perform global search and replace with interactive confirmation across entire file

Example: Example (Vim)

```
:%s/foo/bar/gc
```

Example: Example (Neovim)

```
vim.cmd('%s/foo/bar/gc')
```

Source: vim.fandom.com

55.8 Advanced Search with Case Sensitivity

Category: search_replace
Tags: search, case-sensitivity, configuration

Configure intelligent case-sensitive searching that adapts to your input pattern

Example: Example (Vim)

```
set ignorecase
set smartcase

" Force case sensitivity per search
" /the\c (always case-insensitive)
" /the\C (always case-sensitive)
```

Example: Example (Neovim)

```
vim.opt.ignorecase = true
vim.opt.smartcase = true

-- Note: \c and \C modifiers work the same in Lua search
```

Source: vim.fandom.com

55.9 Advanced Search with Case Sensitivity Controls

Category: search_replace
Tags: search, case-sensitivity, pattern-matching

Configure intelligent case-sensitive searching with smart case detection and custom mappings

Example: Example (Vim)

```
" Enable case-insensitive searching
set ignorecase
set smartcase

" Custom * search with smart case behavior
:nnoremap * /\<<C-R>=expand('<cword>')\>\><CR>
```

Example: Example (Neovim)

```
-- Enable case-insensitive and smart case searching
vim.opt.ignorecase = true
vim.opt.smartcase = true

-- Custom * search with smart case behavior
vim.keymap.set('n', '*', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.search('\<' .. word .. '\>', 'n')
end, { desc = 'Smart word search' })
```

Source: vim.fandom.com

55.10 Advanced Search with OR Patterns

Category: search_replace
Tags: regex, search, text-matching

Use \| to search for multiple whole words, with precise word boundary matching

Example: Example (Vim)

```
" Search for whole words 'red', 'green', 'blue'
/\<\(red\|green\|blue\)\>
```

Example: Example (Neovim)

```
-- Neovim search with multiple whole word options
vim.fn.search('\<\(red\|green\|blue\)\>')
```

Source: vim.fandom.com

55.11 Advanced Substitution with Capture Groups

Category: search_replace
Tags: regex, substitution, pattern-matching

Use capture groups to perform complex search and replace operations by referencing

matched subpatterns

Example: Example (Vim)

```
:g/\(M\)\([0-9]\)\([0-9]\)/s//\1[\2][\3]/g
```

Example: Example (Neovim)

```
-- In Lua, use vim.cmd for complex substitution
vim.cmd('g/\(M\)\([0-9]\)\([0-9]\)/s//\1[\2][\3]/g')
```

Source: vim.fandom.com

55.12 Advanced Word Highlight Search Navigation

Category: search_replace
Tags: search, navigation, highlight

Navigate between highlighted words with custom forward and backward search commands

Example: Example (Vim)

```
" Search next/previous highlighted word
map <leader>f :call FindNextHighlight()<CR>
map <leader>F :call FindPrevHighlight()<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>f', function()
  -- Find next highlighted word
end, { desc = 'Find next highlighted word' })

vim.keymap.set('n', '<leader>F', function()
  -- Find previous highlighted word
end, { desc = 'Find previous highlighted word' })
```

Source: vim.fandom.com

55.13 Advanced Word Search Mappings

Category: search_replace
Tags: search, key-mapping, word-navigation

Custom mappings for word search that respect smartcase behavior

Example: Example (Vim)

```
nnoremap * /\<<C-R>=expand('<cword>')\>\><CR>
nnoremap # ?\<<C-R>=expand('<cword>')\>\><CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '*', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.search('\<' .. word .. '\>')
end)

vim.keymap.set('n', '#', function()
  local word = vim.fn.expand('<cword>')
  vim.fn.search('\<' .. word .. '\>', 'b')
end)
```

Source: vim.fandom.com

55.14 Advanced Word Searching with Regular Expressions

Category: search_replace
Tags: regex, searching, word-boundary

Find whole words precisely using word boundary markers \< and \>

Example: Example (Vim)

```
" Search for whole word 'i'
/\<i\>

" Find words starting or ending with 'i'
/\<i
/i\>
```

Example: Example (Neovim)

```
-- Lua doesn't change regex syntax, use same patterns
-- Can use vim.fn.search() for programmatic searches
vim.fn.search('\<i\>')
```

Source: vim.fandom.com

55.15 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: Example (Vim)

```
:%s/\v(foo|bar)/baz/g  " replace foo or bar with baz
```

Source: Community contributed

55.16 Alternative Delimiters for Substitution

Category: search_replace
Tags: substitution, search, text-manipulation

Use alternative delimiters in substitution commands to handle paths with multiple slashes

Example: Example (Vim)

```
" Substitute using different delimiters
:s,/abc/def/ghi/,,en
```

Example: Example (Neovim)

```
-- Lua equivalent (directly use vim command)
vim.cmd.substitute(',/abc/def/ghi/,,e')
```

Source: vim.fandom.com_wich_include_backslashes.)

55.17 Avoid escaping slashes in search and replace operations

Category: Search
Tags: search, replace, escaping, vimgrep, regex, separator

Example: Example (Vim)

```
%s/http:\/\//fpt:\/\//gc
```

Source: Community contributed

55.18 Block-wise Visual Substitution Without Plugin

Category: search_replace
Tags: substitution, visual-selection

Use \%V to limit substitutions to visual selection area

Example: Example (Vim)

```
:%s/\%V{pattern}/{string}/g
```

Example: Example (Neovim)

```
-- Limit substitution to visual selection area
-- vim.cmd(':%s/\%V{pattern}/{string}/g')
```

Source: vim.fandom.com

55.19 CSV Column Search and Manipulation

Category: search_replace
Tags: csv, search, column-operations

Advanced searching within specific columns and ability to copy or delete columns in CSV files

Example: Example (Vim)

```
" Search within a specific column
:SC n=str  " Search 'str' in nth column
:SC str    " Search in current column

" Copy a column
:CC n x     " Copy nth column to register x

" Delete a column
:DC n       " Delete nth column
```

Example: Example (Neovim)

```
-- Note: This would typically require a custom Lua function or plugin
-- Example skeleton for column operations
local function csv_search_column(column, search_term)
  -- Implement column-specific search logic
end

local function csv_copy_column(column, register)
  -- Implement column copy logic
end

local function csv_delete_column(column)
  -- Implement column deletion logic
end
```

Source: vim.fandom.com

55.20 CSV Search Within Specific Column

Category: search_replace
Tags: csv, search, data-processing

Search for a specific value within a designated column in a CSV file

Example: Example (Vim)

```
" Search within a specific column
" :SC 2=john  " search for john in the 2nd column
" :SC john    " search in currently highlighted column
```

Example: Example (Neovim)

```
-- Note: Full implementation would require a custom Lua function
-- This is a conceptual representation
vim.api.nvim_create_user_command('SC', function(opts)
  local args = opts.args
  local column, search_term

  if args:match('=%w+') then
    column, search_term = args:match('(%d+)=(.+)')
  else
    search_term = args
  end

  -- Implement column-specific search logic
  print(string.format('Searching for %s in column %s', search_term, column
  ↪  or 'current'))
end, { nargs = '+' })
```

Source: vim.fandom.com

55.21 Case-Sensitive and Insensitive Search Replace

Category: search_replace
Tags: search, replace, case-sensitivity

Control case sensitivity during search and replace operations

Example: Example (Vim)

```
:%s/foo/bar/gci   # Case insensitive
:%s/foo/bar/gcI   # Case sensitive
```

Example: Example (Neovim)

```
vim.cmd(':%s/foo/bar/gci')    -- Case insensitive
vim.cmd(':%s/foo/bar/gcI')    -- Case sensitive
```

Source: vim.fandom.com

55.22 Change Case Using Regex Substitutions

Category: search_replace
Tags: regex, text-transformation, substitution

Powerful way to change text case using Vim's regex substitution with case modifiers

Example: Example (Vim)

```
" Lowercase entire file
:%s/.*/\L&/

" Uppercase words after HTML tags
:%s/<\(\w*\)/<\U\1/g
```

Example: Example (Neovim)

```
-- Lowercase entire file
vim.cmd(':%s/.*/\L&/')

-- Uppercase words after HTML tags
vim.cmd(':%s/<\(\w*\)/<\U\1/g')
```

Source: vim.fandom.com

55.23 Comment Lines Matching a Pattern

Category: search_replace
Tags: pattern-matching, line-editing, search-and-replace

Modify lines matching a specific pattern by adding a comment character or transforming text across multiple lines

Example: Example (Vim)

```
:'a,.g/foo/normal I*
```

Example: Example (Neovim)

```
-- Lua equivalent for commenting lines matching 'foo'
vim.cmd('g/foo/normal I*')
```

Source: vim.fandom.com

55.24 Compress Multiple Blank Lines

Category: search_replace
Tags: text-cleaning, whitespace

Reduce multiple consecutive blank lines to a single blank line

Example: Example (Vim)

```
# Collapse multiple blank lines
:v/./,/./-j
# Alternative substitution method
:%s/^$\n^$//g
```

Example: Example (Neovim)

```
-- Collapse multiple blank lines
vim.cmd('v/./,/./-j')
-- Alternative substitution method
vim.cmd('%s/^$\n^$//g')
```

Source: vim.fandom.com

55.25 Convenient Search and Highlight Clearing

Category: search_replace
Tags: key-mapping, search, highlight

Map Ctrl-L to clear search highlighting, making it easy to temporarily remove highlight after searching

Example: Example (Vim)

```
" Clear search highlighting with Ctrl-L
nnoremap <C-L> :nohl<CR><C-L>
```

Example: Example (Neovim)

```
-- Clear search highlighting
```

```
vim.keymap.set('n', '<C-L>', ':nohl<CR><C-L>', { noremap = true, silent =
↪  true })
```

Source: vim.fandom.com

55.26 Convert C++ Comments to C Comments

Category: search_replace
Tags: text-processing, regex, comment-conversion

Quickly convert single-line C++ comments (//) to multi-line C comments (/* */), useful for MISRA compliance or code style standardization

Example: Example (Vim)

```
:map <F5> %s,//\(.*\),/*\1 */,
```

Example: Example (Neovim)

```
-- Lua equivalent for converting C++ comments to C comments
vim.keymap.set('n', '<F5>', function()
  vim.cmd('%s,//\\(.*\\),/*\1 */,')
end, { desc = 'Convert C++ comments to C comments' })
```

Source: vim.fandom.com

55.27 Convert Decimal Encoded Characters

Category: search_replace
Tags: text-processing, character-encoding, substitution

Quickly convert decimal-encoded characters (like \nnn or &#nnn;) to their actual character representation in text

Example: Example (Vim)

```
:%s/\\\([0-9]*\)/\=nr2char(submatch(1))/g
:%s/&#\([0-9]*\);/\=nr2char(submatch(1))/g
```

Example: Example (Neovim)

```
vim.cmd(':%s/\\\([0-9]*\)/\=nr2char(submatch(1))/g')
vim.cmd(':%s/&#\([0-9]*\);/\=nr2char(submatch(1))/g')
```

Source: vim.fandom.com

55.28 Convert HTML Attributes to Lowercase

Category: search_replace
Tags: html, regex, text-transformation

Convert HTML tag attributes to lowercase while preserving attribute values

Example: Example (Vim)

```
:%s/\(<[^>]*\)\@<=\<\(\a*\)\ze=['"]*/\L\2/g
```

Example: Example (Neovim)

```
vim.cmd('%s/\(<[^>]*\)\@<=\<\(\a*\)\ze=['"]*/\L\2/g')
```

Source: vim.fandom.com

55.29 Convert HTML Tags to Lowercase

Category: search_replace
Tags: html, regex, text-transformation

Efficiently convert HTML tag names to lowercase using a powerful Vim regex substitution command

Example: Example (Vim)

```
:%s/<\/\?\zs\(\a\+\)\ze[ >]/\L\1/g
```

Example: Example (Neovim)

```
vim.cmd('%s/<\/\?\zs\(\a\+\)\ze[ >]/\L\1/g')
```

Source: vim.fandom.com

55.30 Copy Search Matches Efficiently

Category: search_replace
Tags: search, registers, clipboard

Quickly copy all text matching a search pattern to a register or clipboard, supporting multiline matches

Example: Example (Vim)

```
function! CopyMatches(reg)
  let hits = []
  %s//\=len(add(hits, submatch(0))) ? submatch(0) : ''/gne
  let reg = empty(a:reg) ? '+' : a:reg
  execute 'let @'.reg.' = join(hits, "\n") . "\n"'
endfunction
command! -register CopyMatches call CopyMatches(<q-reg>)
```

Example: Example (Neovim)

```
function _G.copy_matches(reg)
  local hits = {}
  local view = vim.fn.winsaveview()
  vim.cmd('normal! ggyiG')
  vim.fn.winrestview(view)

  vim.fn.setreg(reg or '+', table.concat(hits, '\n') .. '\n')
end

vim.api.nvim_create_user_command('CopyMatches', function(opts)
  _G.copy_matches(opts.args)
end, { nargs = '?' })
```

Source: vim.fandom.com

55.31 Copy Search Matches to Clipboard

Category: search_replace
Tags: search, clipboard, register

Efficiently copy search matches or matching lines to clipboard or registers, supporting multiline matches

Example: Example (Vim)

```
function! CopyMatches(reg)
  let hits = []
  %s//\=len(add(hits, submatch(0))) ? submatch(0) : ''/gne
  let reg = empty(a:reg) ? '+' : a:reg
  execute 'let @'.reg.' = join(hits, "\n") . "\n"'
endfunction
command! -register CopyMatches call CopyMatches(<q-reg>)
```

Example: Example (Neovim)

```
function CopyMatches(reg)
  local hits = {}
  vim.cmd('keepjumps %substitute//\=luaeval("table.insert(hits,
  ↪  vim.fn.submatch(0)) or vim.fn.submatch(0)")/ne')
```

```
  local register = reg == '' and '+' or reg
  vim.fn.setreg(register, table.concat(hits, '\n') .. '\n')
end

vim.api.nvim_create_user_command('CopyMatches', function(opts)
  CopyMatches(opts.args)
end, { nargs = '?' })
```

Source: vim.fandom.com

55.32 Count Pattern Matches in Buffer

Category: search_replace
Tags: search, count, pattern-matching

Quickly count the number of occurrences of a pattern in the current buffer using substitute command with 'n' flag

Example: Example (Vim)

```
:%s/pattern//gn  " Count all matches
:%s/pattern//n   " Count lines with matches
```

Example: Example (Neovim)

```
-- Use ex command from Lua
vim.cmd(':%s/pattern//gn')  -- Count all matches
vim.cmd(':%s/pattern//n')   -- Count lines with matches
```

Source: vim.fandom.com

55.33 Create Outline Using Vimgrep

Category: search_replace
Tags: search, navigation, outline

Quick way to generate an outline of a file using vimgrep, especially useful for files with fold markers

Example: Example (Vim)

```
:vimgrep /{{{/j %
:copen
```

Example: Example (Neovim)

```
vim.cmd('vimgrep /{{{/j %')
vim.cmd('copen')
```

Source: vim.fandom.com

55.34 Cross-Buffer Text Replacement via Substitute

Category: search_replace
Tags: search, global-replace, registers

Replace all instances of last searched text with last yanked text across entire buffer

Example: Example (Vim)

```
:%s//\=@"/g
```

Example: Example (Neovim)

```
vim.cmd('%%s//\\=@"/g')
```

Source: vim.fandom.com

55.35 Delete Lines Containing Search Matches

Category: search_replace
Tags: search, deletion, filtering

Safely delete lines containing search matches, including support for multiline patterns

Example: Example (Vim)

```
function! DeleteLines(pattern) range
  let delid = '<!DELETE!LINE!ID!>'
  if search(delid, 'cnw') > 0
    echo 'Error: buffer contains pattern used to delete lines'
    return
  endif
  let pattern = empty(a:pattern) ? @/ : a:pattern
  let sub = a:firstline . ',' . a:lastline . 's/' . escape(pattern, '/')
  let rep = '/\=delid . (submatch(0) =~ "\n$" ? "\r" : "")/e'
  execute sub . rep . (&gdefault ? '' : 'g')
  execute 'g/\C' . delid . '/' . 'd'
endfunction
command! -nargs=? -range=% DeleteLines k'|<line1>,<line2>call
↪  DeleteLines(<q-args>)
```

Example: Example (Neovim)

```
function DeleteLines(pattern, line1, line2)
  local delid = '<!DELETE!LINE!ID!>'
  if vim.fn.search(delid, 'cnw') > 0 then
    print('Error: buffer contains pattern used to delete lines')
    return
  end

  local search_pattern = pattern == '' and vim.fn.getreg('/') or pattern
  local cmd = string.format('%d,%ds/%s/%s/e%s',
    line1, line2,
    vim.fn.escape(search_pattern, '/'),
    delid .. (string.match(vim.fn.submatch(0), '\n$') and '\r' or ''),
    vim.o.gdefault and '' or 'g')

  vim.cmd(cmd)
  vim.cmd('global/\C' .. delid .. '/delete')
end

vim.api.nvim_create_user_command('DeleteLines', function(opts)
  DeleteLines(opts.args, opts.line1, opts.line2)
end, { nargs = '?', range = true })
```

Source: vim.fandom.com

55.36 Delete Lines with Selective Filtering

Category: search_replace
Tags: global-command, filtering, line-deletion

Efficiently delete lines matching a pattern while preserving exceptions using Vim's global command with conditional deletion

Example: Example (Vim)

```
:global:^./help/:if (match(getline(line(".")), '^./help/de/') == -1) |
↪  delete | endif
```

Example: Example (Neovim)

```
-- Delete help files except German ones
vim.api.nvim_exec([[g!/\.\/help\/de/d]], false)
```

Source: vim.fandom.com

55.37 Dynamic Highlight Search and Navigation

Category: search_replace
Tags: search, navigation, highlight

Add custom search commands to find next/previous highlighted words across the buffer, with flexible navigation options

Example: Example (Vim)

```
" Search next/previous highlighted words
map <leader>f :call FindNextHighlight()<CR>
map <leader>F :call FindPrevHighlight()<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<leader>f', function()
  -- Find next highlighted word
end, { desc = 'Find next highlighted word' })

vim.keymap.set('n', '<leader>F', function()
  -- Find previous highlighted word
end, { desc = 'Find previous highlighted word' })
```

Source: vim.fandom.com

55.38 Dynamic Number Substitution

Category: search_replace
Tags: substitution, search-replace, incremental-numbering

Replace text with incrementing numbers using a custom function

Example: Example (Vim)

```
function Inc(...)
  let result = g:i
  let g:i += a:0 > 0 ? a:1 : 1
  return result
endfunction

" Replace 'abc' with 'xyz_N'
:let i = 1 | %s/abc/\='xyz_' . Inc()/g
```

Example: Example (Neovim)

```
-- Lua equivalent
local i = 1
vim.cmd(':%s/abc/xyz_' .. i .. '/g')
-- Note: More complex incremental logic would require custom Lua function
```

Source: vim.fandom.com

55.39 Exclude Lines Containing Specific Pattern

Category: search_replace
Tags: search, filtering, regex

Use :v command to filter out lines matching a pattern, useful for log file analysis or text processing

Example: Example (Vim)

```
:v/Warning/p

:sav junk.log
:v/warning/d
```

Example: Example (Neovim)

```
-- Filter out lines not matching a pattern
-- Print lines not containing 'Warning'
vim.cmd('v/Warning/p')

-- Save filtered version of file
vim.cmd('sav junk.log')
vim.cmd('v/warning/d')
```

Source: vim.fandom.com

55.40 Find Whole Words in Search

Category: search_replace
Tags: regex, search, text-navigation

Use \<word\> to search for exact whole word matches, avoiding partial word matches

Example: Example (Vim)

```
# Search for whole word 'i'
/\<i\>
```

Example: Example (Neovim)

```
-- In Neovim, can use same regex pattern
vim.fn.search('\<i\>', 'n')
```

Source: vim.fandom.com

55.41 Find and Remove Duplicate Words

Category: search_replace
Tags: regex, text-cleanup, pattern-matching

Find and identify accidentally duplicated words in text

Example: Example (Vim)

```
" Find duplicated words
/\(\<\w\+\>\)\s*\<\1\>
```

Example: Example (Neovim)

```
-- Find duplicated words
vim.fn.search('\(\<\w\+\>\)\s*\<\1\>')
```

Source: vim.fandom.com

55.42 Flexible CSV Column Search and Manipulation

Category: search_replace
Tags: csv, search, column-operations

Advanced CSV column search, allowing searching within specific columns and performing operations like copying or deleting columns

Example: Example (Vim)

```
" Search within a specific column
:SC n=str  " search for 'str' in nth column

" Copy a specific column
:CC n x     " copy nth column to register x

" Delete a specific column
:DC n       " delete nth column
```

Example: Example (Neovim)

```
-- Implemented via custom Lua functions
-- Example search within column
function _G.search_column(column, search_term)
  local pattern = string.format('^([^,]*,){%d}[^,]*%s', column-1,
  ↪  search_term)
  vim.cmd('/' .. pattern)
end

-- Copy column function
```

```
function _G.copy_column(column, register)
  register = register or '"'
  -- Implement column extraction and copying logic
end

-- Delete column function
function _G.delete_column(column)
  -- Implement column deletion logic
end

-- Create user commands
vim.api.nvim_create_user_command('SC', function(opts)
  local args = opts.args:match('(%d*)=?(.*)') or 0
  _G.search_column(tonumber(args) or 0, args)
end, { nargs = 1 })
```

Source: vim.fandom.com

55.43 Flexible Keyword Search and Jump

Category: search_replace
Tags: search, keyword-navigation, prompt

Interactively search for keywords across files and jump to specific occurrences

Example: Example (Vim)

```
function! s:JumpPrompt()
  let keyword = input("Keyword to find: ")
  if strlen(keyword) > 0
    let v:errmsg = ""
    exe "ilist! " . keyword
    if strlen(v:errmsg) == 0
      let nr = input("Which one: ")
      if nr =~ '\d\+'
        exe "ijump! " . nr . keyword
      endif
    endif
  endif
endfunction

nnoremap <Leader>p :call <SID>JumpPrompt()<CR>
```

Example: Example (Neovim)

```
local function jump_prompt()
  local keyword = vim.fn.input('Keyword to find: ')
  if #keyword > 0 then
    vim.cmd('ilist! ' .. keyword)
    local nr = vim.fn.input('Which one: ')
    if nr:match('^%d+$') then
```

```
      vim.cmd('ijump! ' .. nr .. keyword)
    end
  end
end

vim.keymap.set('n', '<Leader>p', jump_prompt, { desc = 'Prompt for keyword
↪  and jump' })
```

Source: vim.fandom.com

55.44 Flexible Multi-Word Search

Category: search_replace
Tags: regex, search, text-navigation

Use \| to search for multiple alternative whole words in a single pattern

Example: Example (Vim)

```
# Search for whole words 'red', 'green', or 'blue'
/\<\(red\|green\|blue\)\>
```

Example: Example (Neovim)

```
-- Lua regex search for whole words
vim.fn.search('\v<(red|green|blue)>', 'n')
```

Source: vim.fandom.com

55.45 Flexible Multi-Word Search and Replace

Category: search_replace
Tags: regex, substitution, pattern-matching

Search or replace multiple whole words using OR operator

Example: Example (Vim)

```
" Search for multiple whole words
/\<\(red\|green\|blue\)\>

" Replace multiple words
:%s/\<\(red\|green\|blue\)\>/purple/g
```

Example: Example (Neovim)

```
-- Use vim.cmd for ex commands
vim.cmd(':%s/\<\(red\|green\|blue\)\>/purple/g')
```

Source: vim.fandom.com

55.46 Flexible Multiple Word Search

Category: search_replace
Tags: regex, search, text-processing

Search for multiple whole words using the \| (or) operator with word boundaries

Example: Example (Vim)

```
" Search for 'red', 'green', or 'blue' as whole words
/\<\(red\|green\|blue\)\>
```

Example: Example (Neovim)

```
-- Lua equivalent for multiple whole word search
local pattern = '\v<(red|green|blue)>'
vim.fn.search(pattern, 'n')
```

Source: vim.fandom.com

55.47 Flexible Search and Replace Delimiters

Category: search_replace
Tags: search, substitution, text-manipulation

Use different characters for search and replace operations to avoid escaping complex paths

Example: Example (Vim)

```
" Use different delimiters for search and replace
:g#/tmp/#s//#tmp#/
```

Example: Example (Neovim)

```
-- Lua equivalent for flexible search/replace
-- Note: Use vim.cmd for Ex commands
vim.cmd('g#/tmp/#s//#tmp#/')
```

Source: vim.fandom.com

55.48 Flexible Word Search Under Cursor

Category: search_replace
Tags: searching, text-objects, pattern-matching

Perform more flexible searches for words under the cursor, including partial matches

Example: Example (Vim)

```
" g* - search for word (including partial matches)
" g# - search backwards for word
```

Example: Example (Neovim)

```
-- These default Vim search commands work in Neovim
-- Useful for finding related words or partial matches
-- Can be enhanced with telescope or other search plugins
```

Source: vim.fandom.com

55.49 Global Command for Powerful Text Manipulation

Category: search_replace
Tags: ex-commands, global-command, text-processing

Use :g to perform complex operations on lines matching a pattern, such as deleting, copying, or modifying lines globally

Example: Example (Vim)

```
" Delete all lines containing 'pattern'
:g/pattern/d

" Delete all lines NOT containing 'pattern'
:g!/pattern/d

" Double space the entire file
:g/^/pu ="\n"
```

Example: Example (Neovim)

```
-- Lua equivalents require using vim.cmd for Ex commands
-- Delete lines containing pattern
vim.cmd('g/pattern/d')

-- Delete lines NOT containing pattern
vim.cmd('g!/pattern/d')

-- Double space file
```

```
vim.cmd('g/^/pu ="\n"')
```

Source: vim.fandom.com

55.50 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: Example (Vim)

```
:%s/foo/bar/g  " replace all 'foo' with 'bar'
```

Source: Community contributed

55.51 Grep Search Across Files with Quickfix Navigation

Category: search_replace
Tags: search, grep, quickfix, navigation

Extends Vim's search capabilities by using grep to search across files and navigate results using quickfix list

Example: Example (Vim)

```
" Set grep program with recursive, case-insensitive search
:set grepprg=grep\ -rinsE

function! Mosh_grep(...)
  if a:0 == 0
    :exec "grep '".@/."' *.*"
  elseif a:0 == 1
    :exec "grep '".@/."' " a:1
  elseif a:0 == 2
    :exec "grep" a:2 " " a:1
  endif
  :map <c-n> :cnext<CR>
  :map <c-p> :cprev<CR>
  :copen
endfunction

" Map g/ to trigger grep search
:map g/ :call Mosh_grep()<CR>
```

Example: Example (Neovim)

```
-- Set grep program
vim.opt.grepprg = 'grep -rinsE'

function _G.mosh_grep(dir, pattern)
  local search_pattern = pattern or vim.fn.getreg('/')
  local search_dir = dir or '*.*'

  vim.cmd('grep ' .. vim.fn.shellescape(search_pattern) .. ' ' ..
  ↪  search_dir)
  vim.cmd('copen')

  -- Map navigation keys
  vim.keymap.set('n', '<C-n>', ':cnext<CR>', {noremap = true, silent =
  ↪  true})
  vim.keymap.set('n', '<C-p>', ':cprev<CR>', {noremap = true, silent =
  ↪  true})
end

-- Map g/ to trigger grep search
vim.keymap.set('n', 'g/', function() _G.mosh_grep() end)
```

Source: vim.fandom.com

55.52 Hex/Decimal Conversion Across Text

Category: search_replace
Tags: number-conversion, text-manipulation, search-replace

User commands to convert decimal and hexadecimal numbers across text ranges, supporting visual selections

Example: Example (Vim)

```
command! -nargs=? -range Dec2hex call s:Dec2hex(<line1>, <line2>, '<args>')
function! s:Dec2hex(line1, line2, arg) range
  if empty(a:arg)
    let cmd = 's/\<\d\+\>/\=printf("0x%x",submatch(0)+0)/g'
    execute a:line1 . ',' . a:line2 . cmd
  else
    echo printf('%x', a:arg + 0)
  endif
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Dec2hex', function(opts)
  local line1, line2 = opts.line1, opts.line2
  local arg = opts.args

  if arg == '' then
```

```
    local cmd = 's/\d\+/\=printf("0x%x", submatch(0) + 0)/g'
    vim.cmd(line1 .. ',' .. line2 .. cmd)
  else
    print(string.format('%x', tonumber(arg)))
  end
end, { nargs = '?', range = true })
```

Source: vim.fandom.com

55.53 Highlight Current Word Without Moving

Category: search_replace
Tags: search, highlight, word-selection

Highlight all occurrences of the current word without jumping to next match

Example: Example (Vim)

```
nnoremap <F8> :let @/='\<<C-R>=expand("<cword>")<CR>\>'<CR>:set hls<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F8>', function()
  local cword = vim.fn.expand('<cword>')
  vim.fn.setreg('/', '\<' .. cword .. '\>')
  vim.opt.hlsearch = true
end, { desc = 'Highlight current word' })
```

Source: vim.fandom.com

55.54 Highlight Only Current Search Result

Category: search_replace
Tags: search, highlight, navigation

Customize search highlighting to focus only on the current match instead of all matches

Example: Example (Vim)

```
set nohlsearch

" highlight only current result of forward search when pressing n
:nnoremap <silent> <nowait> n <Esc>gn
:vnoremap <silent> <nowait> n <Esc>gn

" highlight the entire word of <cword> forward search result with *
:nnoremap <silent> <nowait> * <Esc>*gN
:vnoremap <silent> <nowait> * <Esc>*gN
```

Example: Example (Neovim)

```
-- Disable default highlight search
vim.opt.hlsearch = false

-- Highlight only current search result
vim.keymap.set('n', 'n', 'gn', { silent = true })
vim.keymap.set('v', 'n', 'gn', { silent = true })

vim.keymap.set('n', '*', '*gN', { silent = true })
vim.keymap.set('v', '*', '*gN', { silent = true })
```

Source: vim.fandom.com

55.55 Highlight Search Pattern in Line Listing

Category: search_replace
Tags: search, highlighting, custom-command

Create a custom command to print lines matching a search pattern with the match highlighted

Example: Example (Vim)

```
" Custom command to print lines with search pattern highlighted
command! -nargs=? -range -bar PP :call
↪  PrintWithSearchHighlighted(<line1>,<line2>,<q-args>)
function! PrintWithSearchHighlighted(line1,line2,arg)
  let line=a:line1
  while line <= a:line2
    echo ""
    if a:arg =~ "#"
      echohl LineNr
      echo strpart(" ",0,7-strlen(line)).line."\t"
      echohl None
    endif
    let l=getline(line)
    let index=0
    while 1
      let b=match(l,@/,index)
      if b==-1 |
        echon strpart(l,index)
        break
      endif
      let e=matchend(l,@/,index) |
      echon strpart(l,index,b-index)
      echohl Search
      echon strpart(l,b,e-b)
      echohl None
      let index = e
    endw
    let line=line+1
  endw
```

```
endfunction
```

Example: Example (Neovim)

```
-- Lua equivalent for highlighting search pattern in line listing
function _G.print_with_search_highlighted(line1, line2, arg)
  for line = line1, line2 do
    local l = vim.fn.getline(line)
    local index = 1
    local output = {}

    -- Print line number if # arg is provided
    if arg and arg:find("#") then
      print(string.format("%6d\t", line))
    end

    -- Highlight matching patterns
    while true do
      local start, finish = l:find(vim.fn.getreg("/"), index)
      if not start then
        table.insert(output, l:sub(index))
        break
      end

      -- Add text before match
      table.insert(output, l:sub(index, start-1))

      -- Add highlighted match
      table.insert(output, {vim.fn.matchstr(l, vim.fn.getreg("/"), index),
      ↪  "Search"})

      index = finish + 1
    end

    -- Print with highlighting
    vim.api.nvim_echo(output, false, {})
    print("") -- Add newline between results
  end
end

-- Create command
vim.api.nvim_create_user_command('PP', function(opts)
  _G.print_with_search_highlighted(opts.line1, opts.line2, opts.args)
end, { nargs = '?', range = true })
```

Source: vim.fandom.com

55.56 Highlight and Remove Unwanted Whitespace

Category: search_replace
Tags: whitespace, highlighting, text-cleanup

Easily display or remove whitespace at the end of lines or before tabs

Example: Example (Vim)

```
function ShowSpaces()
  let @/='\v(\s+$)|( +\ze\t)'
  let &hlsearch=!&hlsearch
endfunction

nnoremap <F12> :call ShowSpaces()<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F12>', function()
  vim.o.hlsearch = not vim.o.hlsearch
  vim.fn.setreg('/', '\v(\s+$)|( +\ze\t)')
end, { desc = 'Toggle whitespace highlighting' })
```

Source: vim.fandom.com

55.57 Incremental Search with Live Preview

Category: search_replace
Tags: search, ui, productivity

Enable live search preview that moves cursor while typing search pattern, helping visualize matches in real-time

Example: Example (Vim)

```
set incsearch
```

Example: Example (Neovim)

```
vim.opt.incsearch = true
```

Source: vim.fandom.com

55.58 Insert Newlines Around Patterns Flexibly

Category: search_replace
Tags: text-manipulation, search, substitution

Dynamically insert newlines before or after specific patterns in text, useful for formatting code or text

Example: Example (Vim)

```
command! -bang -nargs=* -range LineBreakAt <line1>,<line2>call
↪  LineBreakAt('<bang>', <f-args>)

function! LineBreakAt(bang, ...) range
  let save_search = @/
  if empty(a:bang)
    let before = ''
    let after = '\ze.'
    let repl = '&\r'
  else
    let before = '.\zs'
    let after = ''
    let repl = '\r&'
  endif
  let pat_list = map(deepcopy(a:000), "escape(v:val, '/\\.*$^~[')")
  let find = empty(pat_list) ? @/ : join(pat_list, '\|')
  let find = before . '\%(' . find . '\)' . after
  execute a:firstline . ',' . a:lastline . 's/'. find . '/' . repl . '/ge'
  let @/ = save_search
endfunction
```

Example: Example (Neovim)

```
function LineBreakAt(bang, ...)
  local save_search = vim.fn.getreg('/')
  local before = bang and '.\\zs' or ''
  local after = bang and '' or '\\ze.'
  local repl = bang and '\\r&' or '&\\r'

  local pat_list = vim.tbl_map(function(val)
    return vim.fn.escape(val, '/\.*$^~[')
  end, {...})

  local find = #pat_list > 0 and table.concat(pat_list, '\\|') or
  ↪  save_search
  find = before .. '\\%(' .. find .. '\\)' .. after

  vim.cmd(string.format('%s,%ss/%s/%s/ge',
    vim.fn.line('.'), vim.fn.line('.'), find, repl))

  vim.fn.setreg('/', save_search)
end

-- Usage:
-- :lua LineBreakAt(false, '(', ')')
-- :lua LineBreakAt(true, 'function')
```

Source: vim.fandom.com

55.59 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: Example (Vim)

```
/function\_s*name    " function followed by whitespace/newlines
/start\_.*end        " match start to end across lines
```

Source: Community contributed

55.60 Operate on Search Matches in Vim 7.4+

Category: search_replace
Tags: search, text-objects, editing

Use `gn` to visually select and operate on search matches easily. Works great with operators like delete, change, etc.

Example: Example (Vim)

```
" After searching, use gn to select and operate on matches
" dgn - delete next match
" cgn - change next match
" Repeat with .
```

Example: Example (Neovim)

```
-- Lua doesn't require special configuration for gn
-- It's a built-in Vim/Neovim feature
-- Example workflow:
-- /pattern  -- search for something
-- cgn       -- change next match
-- .         -- repeat change
```

Source: vim.fandom.com

55.61 Perform Operation on Matching Lines

Category: search_replace
Tags: macro, global-command, text-processing

Run a macro on all lines matching a specific pattern

Example: Example (Vim)

```
" Run macro 'q' on all lines matching 'pattern'
:g/pattern/normal @q
```

Example: Example (Neovim)

```
-- Run macro on matching lines
-- Note: Requires recording macro first
vim.cmd('g/pattern/normal @q')
```

Source: vim.fandom.com

55.62 Powerful Multi-File Search and Replace

Category: search_replace
Tags: search, replace, multi-file

Perform a search and replace across multiple files with additional flags for comprehensive editing

Example: Example (Vim)

```
" Search and replace with global, ignore case, and error-tolerant flags
:%s/\<pig\>/cow/gie|:update|:next
```

Example: Example (Neovim)

```
-- Equivalent multi-file search and replace
-- Note: This would typically be done with more modern Neovim methods like
↪  telescope or quickfix
vim.cmd('%s/\<pig\>/cow/gie')
vim.cmd('update')
vim.cmd('next')
```

Source: vim.fandom.com

55.63 Precise Word Replacement

Category: search_replace
Tags: search, replace, word-boundary

Replace only whole word matches to prevent partial word replacements

Example: Example (Vim)

```
:%s/\<foo\>/bar/gc
```

Example: Example (Neovim)

```
vim.cmd(':%s/\<foo\>/bar/gc')
```

Source: vim.fandom.com

55.64 Quick Count of Word Under Cursor

Category: search_replace
Tags: search, cursor-word, mapping

Create a quick mapping to count occurrences of the word under the cursor

Example: Example (Vim)

```
map ,* *<C-O>:%s///gn<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', ',*', '*<C-O>:%s///gn<CR>', { desc = 'Count word
↪  occurrences' })
```

Source: vim.fandom.com

55.65 Quick HTML Attribute Quoting

Category: search_replace
Tags: html, regex, text-transformation

Automatically add quotes to unquoted HTML attributes using a regex search and replace

Example: Example (Vim)

```
map <F9> :%s/\([^&^?]\)\(\<[[:alnum:]-]\{-}\)=\([[:alnum:]-#%]\+\)/\1\2="\3
↪  "/g<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F9>', ':%s/\([^&^?]\)\(\<[[:alnum:]-]\{-}\)=\([[:alnu
↪  m:]-#%]\+\)/\1\2="\3"/g<CR>', { desc = 'Quote unquoted HTML attributes'
↪  })
```

Source: vim.fandom.com

55.66 Quick Multi-Line Text Substitution

Category: search_replace
Tags: text-manipulation, substitution, global-edit

Use substitute commands to insert or modify text across multiple lines

Example: Example (Vim)

```
":s/^/new text /  " Insert at line start
":s/$/new text/  " Append at line end
":s/old/new/g  " Replace all occurrences
```

Example: Example (Neovim)

```
-- Similar substitution can be done in Neovim
-- Can use vim.cmd for Ex commands
vim.cmd('s/^/new text /')  -- Insert at start
vim.cmd('s/$/new text/')  -- Append at end
```

Source: vim.fandom.com

55.67 Quick Search for Word Under Cursor

Category: search_replace
Tags: search, navigation, cursor

Use * and # to search for the word under the cursor forward and backward

Example: Example (Vim)

```
* searches forward
# searches backward
```

Example: Example (Neovim)

```
-- Built-in Vim behavior, no specific Lua conversion needed
-- Can use vim.fn.expand('<cword>') to get word under cursor
 ↪  programmatically
```

Source: vim.fandom.com

55.68 Quick Text Replacement Using Command Line

Category: search_replace
Tags: substitution, text-editing, search

Efficiently replace long strings by inserting text from buffer into substitute command

Example: Example (Vim)

```
" 1. Start substitute command
" 2. Highlight text
" 3. Use Ctrl-R Ctrl-W to insert current word
```

Example: Example (Neovim)

```
-- Example of inserting current word in substitute command
vim.keymap.set('n', '<leader>s', function()
  local word = vim.fn.expand('<cword>')
  vim.api.nvim_feedkeys(':%s/' .. word .. '//g', 'n', true)
end)
```

Source: vim.fandom.com

55.69 Quick Toggle to Remove Search Highlights

Category: search_replace
Tags: search, highlights, utility

Easily turn off search highlighting during editing sessions

Example: Example (Vim)

```
set nohlsearch
```

Example: Example (Neovim)

```
vim.opt.hlsearch = false
```

Source: vim.fandom.com

55.70 Quick Word Search in Current File

Category: search_replace
Tags: search, quickfix, navigation

Define a command to search for the current word under the cursor in the current file and open results in quickfix window

Example: Example (Vim)

```
command GREP :execute 'vimgrep '.expand('<cword>').' '.expand('%') | :copen
↪  | :cc
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('GREP', function()
  vim.cmd('vimgrep ' .. vim.fn.expand('<cword>') .. ' ' ..
  ↪  vim.fn.expand('%'))
  vim.cmd('copen')
  vim.cmd('cc')
end, {})
```

Source: vim.fandom.com

55.71 Quickly Filter and List Search Matches

Category: search_replace
Tags: search, filtering, command-line

Create a custom command to filter and display lines matching a search pattern in a new buffer

Example: Example (Vim)

```
command! -nargs=? Filter let @a='' | execute 'g/<args>/y A' | new | setlocal
↪  bt=nofile | put! a
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Filter', function(opts)
  -- Clear register a
  vim.fn.setreg('a', '')

  -- Execute global command to yank matching lines to register a
  vim.cmd('g/' .. (opts.args or vim.fn.getreg('/') or '') .. '/y A')

  -- Create new scratch buffer and paste results
  vim.cmd('new')
  vim.wo.buftype = 'nofile'
  vim.cmd('put! a')
end, { nargs = '?' })
```

Source: vim.fandom.com

55.72 Recursive File Search with Vim

Category: search_replace
Tags: file-search, grep, recursive-search

Search recursively through files using ** wildcard, which enables searching in parent and subdirectories

Example: Example (Vim)

```
:vimgrep /dostuff()/j ../**/*.c
```

Example: Example (Neovim)

```
-- Lua equivalent (using vim.fn.execute)
vim.fn.execute('vimgrep /dostuff()/j ../**/*.c')
```

Source: vim.fandom.com

55.73 Redirect Search Results to a New Window

Category: search_replace
Tags: search, mapping, output-redirection

Create a mapping to redirect global search results to a new window using register redirection

Example: Example (Vim)

```
nnoremap <silent> <F3> :redir @a<CR>:g//<CR>:redir END<CR>:new<CR>:put!
↪  a<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F3>', function()
  -- Redirect output to register a
  vim.cmd('redir @a')
  -- Repeat last global command
  vim.cmd('g//')
  -- End redirection
  vim.cmd('redir END')
  -- Create new window and paste register contents
  vim.cmd('new')
  vim.cmd('put! a')
end, { silent = true })
```

Source: vim.fandom.com

55.74 Regex Lookahead and Lookbehind

Category: search_replace
Tags: regex, advanced-search, pattern-matching

Advanced regex techniques to match patterns with conditional context using lookahead and lookbehind assertions

Example: Example (Vim)

```
" Positive lookahead: match 'bar' only if preceded by 'foo'
/\(foo\)\@<=bar\(baz\)\@=
```

Example: Example (Neovim)

```
-- In Neovim, use vim.fn.search() with magic regex
vim.fn.search('\v(foo)@<=bar(baz)@=')
```

Source: vim.fandom.com

55.75 Remove Duplicate Lines Efficiently

Category: search_replace
Tags: text-processing, deduplication

Remove consecutive duplicate lines from a file using global command or unique filter

Example: Example (Vim)

```
# Remove duplicate consecutive lines
:g/^\(.*\)\(\r\?\n\1\)\+$/d
# Alternative method using uniq
:%!uniq
```

Example: Example (Neovim)

```
-- Remove duplicate consecutive lines
vim.cmd('g/^\(.*\)\(\r\?\n\1\)\+$/d')
-- Alternative method using uniq filter
vim.cmd('%!uniq')
```

Source: vim.fandom.com

55.76 Remove File Path Prefix with Advanced Search/Replace

Category: search_replace
Tags: regex, file-manipulation, global-command

Quickly remove path prefixes from file lists using a flexible regex substitution that preserves desired portions of file paths

Example: Example (Vim)

```
:%s/^.\{-}\ze///
```

Example: Example (Neovim)

```
vim.cmd('%s/^.\\{-}\\ze///g')
```

Source: vim.fandom.com

55.77 Remove Multiple Path Levels in File Lists

Category: search_replace
Tags: regex, file-manipulation, global-command

Remove multiple directory levels from file paths by adjusting regex quantifier

Example: Example (Vim)

```
:%s/^\((.\{-}\ze\/\)\{3}//
```

Example: Example (Neovim)

```
vim.cmd('%s/^\\((.\\{-}\\ze\\/\\)\\{3}//g')
```

Source: vim.fandom.com

55.78 Remove search highlighting

Category: Search
Tags: search, highlight, remove

Use `:nohl` to remove search highlighting after performing a search.

Example: Example (Vim)

```
:nohl
```

Source: Community contributed

55.79 Repeat Substitution from Cursor Position

Category: search_replace
Tags: substitute, custom-command, advanced-editing

Create a custom command to repeat substitution from the current cursor position, allowing more flexible text replacement

Example: Example (Vim)

```
" RS: repeat substitution command
com! -range -nargs=* RS call RepeatSubst(<q-args>)

fun! RepeatSubst(subexpr)
  if a:subexpr != ""
    let g:repeatsubst= a:subexpr
  endif
  let curcol= col(".")
  let sep = strpart(g:repeatsubst,0,1)
  let pat = substitute(g:repeatsubst,'^.\(.*\)'.sep.'.*$','\1','')
  s/\%#./\r&/
  let curcol= curcol + matchend(getline(".“),pat)
  exe "s".g:repeatsubst
  norm! k
  j!
  exe 'norm! '.curcol.'|'
endfun
```

Example: Example (Neovim)

```
-- Lua implementation of repeat substitution
function RepeatSubst(subexpr)
  if subexpr ~= "" then
    vim.g.repeatsubst = subexpr
  end

  local curcol = vim.fn.col('.')
  local sep = string.sub(vim.g.repeatsubst, 1, 1)
  local pat = string.match(vim.g.repeatsubst, '^.(.-)' .. sep)

  -- Perform substitution from current cursor position
  vim.cmd('s/\%#./\r&/')
  curcol = curcol + vim.fn.matchend(vim.fn.getline('.'), pat)
  vim.cmd('s' .. vim.g.repeatsubst)
  vim.cmd('norm! k')
  vim.cmd('j!')
  vim.cmd('norm! ' .. curcol .. '|')
end

-- Create a custom command
vim.api.nvim_create_user_command('RS', function(opts)
  RepeatSubst(opts.args)
end, { nargs = '*', range = true })
```

Source: vim.fandom.com

55.80 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: Example (Vim)

```
:%s//new_text/g  " replace last searched pattern with new_text
```

Source: Community contributed

55.81 Replace Spaces with Tabs Globally

Category: search_replace
Tags: whitespace, text-manipulation, global-replace

Replace all spaces with tabs, respecting current tabstop width

Example: Example (Vim)

```
:%s/ /\t/g
```

Example: Example (Neovim)

```
vim.cmd('%s/ /\t/g')
```

Source: vim.fandom.com

55.82 Replace Word Under Cursor Quickly

Category: search_replace
Tags: cursor-word, substitution, refactoring

Quickly replace all instances of the word under the cursor in the entire buffer

Example: Example (Vim)

```
:map <S-F4> :%s/<C-r><C-w>//gc<Left><Left><Left>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<S-F4>', ':%s/<C-r><C-w>//gc<Left><Left><Left>', { desc
↪  = 'Replace word under cursor globally' })
```

Source: vim.fandom.com

55.83 Search Only Within Visually Selected Range

Category: search_replace
Tags: visual-mode, search, custom-mapping

Implement a custom search function that limits search results to a visually selected range, allowing targeted searching within specific code blocks or regions

Example: Example (Vim)

```
function FindNextRegionWork()
  let l:flags = g:sdir == "f" ? "w" : "wb"
  call search(g:searchstr, l:flags)
  let l:srw = line(".")
  let l:scl = col(".")
  while l:srw != 0
    if (l:srw == g:sbr && l:scl >= g:sbc) ||
       (l:srw == g:ser && l:scl <= g:sec) ||
       (l:srw > g:sbr && l:srw < g:ser)
      break
    endif
    call search(g:searchstr, l:flags)
    let l:srw = line(".")
    let l:scl = col(".")
  endwhile
endfunction
```

Example: Example (Neovim)

```
function _G.find_next_region_work()
  local flags = vim.g.sdir == 'f' and 'w' or 'wb'
  vim.fn.search(vim.g.searchstr, flags)
  local srw = vim.fn.line('.')
  local scl = vim.fn.col('.')
  while srw ~= 0 do
    if (srw == vim.g.sbr and scl >= vim.g.sbc) or
       (srw == vim.g.ser and scl <= vim.g.sec) or
       (srw > vim.g.sbr and srw < vim.g.ser) then
      break
    end
    vim.fn.search(vim.g.searchstr, flags)
    srw = vim.fn.line('.')
    scl = vim.fn.col('.')
  end
end

-- Add keymappings
vim.keymap.set('n', '<Leader>/', function()
  vim.ui.input({prompt = 'Find what?: '}, function(input)
    if input then
      vim.g.searchstr = input
      vim.g.sbr = vim.fn.line("'<")
      vim.g.sbc = vim.fn.col("'<")
```

```
      vim.g.ser = vim.fn.line("'>")
      vim.g.sec = vim.fn.col("'>")
      _G.find_next_region_work()
    end
  end)
end)
```

Source: vim.fandom.com

55.84 Search Paths Without Escaping Slashes

Category: search_replace
Tags: search, command-line, text-editing

Use alternative delimiters like '?' or other characters to search file paths without escaping slashes

Example: Example (Vim)

```
" Search with alternative delimiters
:g ?c:/tmp/x/y/z/? d
```

Example: Example (Neovim)

```
-- Lua equivalent for alternative search delimiters
-- Note: Use vim.cmd for Ex commands
vim.cmd('g ?c:/tmp/x/y/z/? d')
```

Source: vim.fandom.com

55.85 Search and Find Lines Longer Than 80 Characters

Category: search_replace
Tags: search, text-analysis, regex

Quickly locate lines that exceed 80 characters in length, useful for code style checks

Example: Example (Vim)

```
/\%>80v.\+
```

Example: Example (Neovim)

```
vim.fn.search('\%>80v.\+')
```

Source: vim.fandom.com

55.86 Search and replace in multiple files (vimgrep+cdo)

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: 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: 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.

Example: Example (Vim)

```
:vimgrep /<a [^>]*text-red[^>]*>/gj **/*.html

:cdo s/text-red/text-blue/gc
```

Source: Community contributed

55.87 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: Example (Vim)

```
?hello  " search backward for 'hello'
n       " next match (backward)
N       " previous match (forward)
```

Source: Community contributed

55.88 Search in selection

Category: Search
Tags: replace, selection, range

Use `:'<,'>s/old/new/g` to replace only in visual selection.

Example: Example (Vim)

```
:'<,'>s/foo/bar/g  " replace in selection
```

Source: Community contributed

55.89 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: Example (Vim)

```
/function/+2     " position cursor 2 lines after 'function'
/end/-1          " position cursor 1 line before 'end'
```

Source: Community contributed

55.90 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: Example (Vim)

```
/\v(hello|world)  " search for 'hello' or 'world' (very magic)
/\<function\>     " search for exact word 'function'
/\vd+             " search for one or more digits
```

Source: Community contributed

55.91 Sequential Line Numbering with Counter

Category: search_replace
Tags: numbering, global-command, text-manipulation

Sequentially number lines in a range starting from 1 using a counter variable

Example: Example (Vim)

```
:let counter=0|10,20g/^/let counter=counter+1|s/^/\=counter."\t"
```

Example: Example (Neovim)

```
-- Initialize counter and number lines sequentially
vim.g.counter = 0
vim.cmd(':10,20g/^/let counter=counter+1')
vim.cmd(':10,20s/^/\\=g:counter."\t"')
```

Source: vim.fandom.com

55.92 Simple Regex Substitution with Capture Groups

Category: search_replace
Tags: regex, substitution, pattern-matching

Easily transform patterns using capture groups and references

Example: Example (Vim)

```
%s/\v(\d)(\d)/[\1][\2]
```

Example: Example (Neovim)

```
-- Use very magic mode for simplified regex
vim.cmd('%s/\v(\d)(\d)/[\1][\2]')
```

Source: vim.fandom.com

55.93 Smart Case Sensitivity in Search

Category: search_replace
Tags: search, case-sensitivity, configuration

Automatically detect whether to ignore case during searches based on query

Example: Example (Vim)

```
set smartcase
```

Example: Example (Neovim)

```
vim.opt.smartcase = true
```

Source: vim.fandom.com

55.94 Smart Case-Sensitive Search

Category: search_replace
Tags: search, case-sensitivity, intelligent-search

Automatically detect case sensitivity in searches for more flexible searching

Example: Example (Vim)

```
set smartcase
```

Example: Example (Neovim)

```
vim.opt.smartcase = true
```

Source: vim.fandom.com

55.95 Sort Lines by Selected Substring

Category: search_replace
Tags: selection, sorting, text-manipulation

Quickly sort log files or text by a selected substring using visual mode mappings

Example: Example (Vim)

```
" Sort by selection
:vmap 0 :<BS><BS><BS><BS><BS>g<M-x>\M<S-Insert><M-x>m0<CR>
:vmap $ :<BS><BS><BS><BS><BS>g<M-x>\M<S-Insert><M-x>m$<CR>
:vmap p :<BS><BS><BS><BS><BS>g<M-x>\M<S-Insert><M-x>p<CR>
```

Example: Example (Neovim)

```
-- Sort lines by selected text to top/bottom
vim.keymap.set('v', 'm0', function()
  local selected_text = vim.fn.getreg('*')
  vim.cmd('g/' .. selected_text .. '/m0')
end, { desc = 'Move matching lines to top' })

vim.keymap.set('v', 'm$', function()
  local selected_text = vim.fn.getreg('*')
  vim.cmd('g/' .. selected_text .. '/m$')
end, { desc = 'Move matching lines to bottom' })
```

Source: vim.fandom.com

55.96 Speed Up Recursive Searches

Category: search_replace
Tags: performance, search, optimization

Use noautocmd to significantly speed up recursive file searches by disabling autocommands

Example: Example (Vim)

```
:noautocmd vimgrep /{pattern}/[flags] {file(s)}
```

Example: Example (Neovim)

```
vim.cmd('noautocmd vimgrep /{pattern}/[flags] {file(s)}')
```

Source: vim.fandom.com

55.97 Swap Adjacent Words with Regex

Category: search_replace
Tags: regex, text-manipulation, advanced-editing

Use a sophisticated regex pattern to swap adjacent words, including handling line breaks and preserving surrounding characters

Example: Example (Vim)

```
" Swap next word
s/\v(<\k*%#\k*>)(\_.\.{-})(<\k+>)/\3\2\1/

" Swap previous word
s/\v(<\k+>)(.{-})(<\k*%#\k*>)/\3\2\1/
```

Example: Example (Neovim)

```
-- Swap next word with cursor position
vim.cmd('s/\v(<\k*%#\k*>)(\_.\.{-})(<\k+>)/\3\2\1/')

-- Swap previous word with cursor position
vim.cmd('s/\v(<\k+>)(.{-})(<\k*%#\k*>)/\3\2\1/')
```

Source: vim.fandom.com

55.98 Title Case Conversion with Substitute

Category: search_replace
Tags: text-transformation, regex, substitution

Advanced substitution command to convert text to title case, preserving first letter of each word as uppercase

Example: Example (Vim)

```
" Convert line to Title Case
:s/\<\(\w\)\(\w*\)\>/\u\1\L\2/g
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.cmd for Ex command
vim.cmd('s/\v(\w)(\S*)/\u\1\L\2/g')
```

Source: vim.fandom.com

55.99 Toggle Highlighting for Current Word

Category: search_replace
Tags: search, highlight, toggle

Toggle highlighting on and off for the current word under cursor

Example: Example (Vim)

```
let g:highlighting = 0
function! Highlighting()
  if g:highlighting == 1 && @/ =~ '^\<'.expand('<cword>').'\>$'
    let g:highlighting = 0
    return ":silent nohlsearch\<CR>"
  endif
  let @/ = '\<'.expand('<cword>').'\'>
  let g:highlighting = 1
  return ":silent set hlsearch\<CR>"
endfunction
nnoremap <silent> <expr> <CR> Highlighting()
```

Example: Example (Neovim)

```
vim.g.highlighting = false

local function toggle_highlighting()
  local cword = vim.fn.expand('<cword>')
  local current_search = vim.fn.getreg('/')
```

```
  if vim.g.highlighting and current_search:match('^\<' .. cword .. '\>$')
  ↪  then
    vim.g.highlighting = false
    vim.cmd.nohlsearch()
  else
    vim.fn.setreg('/', '\<' .. cword .. '\>')
    vim.g.highlighting = true
    vim.opt.hlsearch = true
  end
end

vim.keymap.set('n', '<CR>', toggle_highlighting, { silent = true, desc =
↪  'Toggle word highlighting' })
```

Source: vim.fandom.com

55.100 Toggle Search Highlight with Single Keypress

Category: search_replace
Tags: search, highlight, mapping

Add a quick way to toggle search highlighting on and off, improving code readability and focus

Example: Example (Vim)

```
" Press F4 to toggle highlighting on/off, and show current value.
:noremap <F4> :set hlsearch! hlsearch?<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F4>', function()
  vim.o.hlsearch = not vim.o.hlsearch
  print('hlsearch: ' .. tostring(vim.o.hlsearch))
end, { desc = 'Toggle search highlighting' })
```

Source: vim.fandom.com

55.101 Toggle Search Highlighting Easily

Category: search_replace
Tags: search, highlight, key-mapping

Add a quick way to toggle search highlighting and clear messages

Example: Example (Vim)

```
" Press Space to turn off highlighting and clear messages
nnoremap <silent> <Space> :nohlsearch<Bar>:echo<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Space>', function()
  vim.cmd.nohlsearch()
  vim.cmd.echo()
end, { silent = true, desc = 'Clear search highlighting' })
```

Source: vim.fandom.com

55.102 Use Alternate Delimiters in Search/Replace

Category: search_replace
Tags: search, replace, delimiter

Use alternative delimiters in search and replace commands to simplify working with paths or complex patterns that contain slashes

Example: Example (Vim)

```
:s#/usr/local/#/opt/#
```

Example: Example (Neovim)

```
-- Example of using alternative delimiter in Neovim
vim.cmd('s#/usr/local/#/opt/#')
```

Source: vim.fandom.com

55.103 Use Expressions in Substitute Commands

Category: search_replace
Tags: substitution, advanced-editing, expressions

Use \'=' in substitute commands to evaluate expressions dynamically, enabling powerful text transformations

Example: Example (Vim)

```
:%s/^/\=line('.')."\t"/
:10,20s/^/\=line('.')."\t"/
```

Example: Example (Neovim)

```
-- Number all lines with line number
vim.cmd(':%s/^/\\=line(".")."\t"/')

-- Number lines 10-20 with line number
vim.cmd(':10,20s/^/\\=line(".")."\t"/')
```

Source: vim.fandom.com

55.104 Use Perl-Style Regex in Vim Substitutions

Category: search_replace
Tags: regex, substitution, advanced-search

Leverage Perl-compatible regular expressions for more powerful text substitutions using external Perl processing

Example: Example (Vim)

```
" Define a custom Perl substitution command
function s:Substitute(line1, line2, sstring)
  let l:lines=getline(a:line1, a:line2)
  let l:sysresult=system("perl -e 'use utf8;' -e '#line 1 ""perl
  ↪  substitution""' -pe ".shellescape("s".escape(a:sstring,"%!").";"),
  ↪  l:lines)
  if v:shell_error
    echo l:sysresult
    return
  endif
  let l:result=split(l:sysresult, "\n", 1)
  execute a:line1.",".a:line2." normal ""_dd"
  call append(a:line1-1, l:result)
  call cursor(a:line1, 1)
endfunction

command -range -nargs=1 S call s:Substitute(<line1>, <line2>, <q-args>)
```

Example: Example (Neovim)

```
-- Lua equivalent of Perl substitution function
local function perl_substitute(line1, line2, sstring)
  -- Get lines from the specified range
  local lines = vim.api.nvim_buf_get_lines(0, line1-1, line2, false)

  -- Prepare Perl command
  local cmd = string.format("perl -e 'use utf8;' -e '#line 1 "perl
  ↪  substitution"' -pe 's%s;'", sstring)

  -- Execute Perl command
  local result = vim.fn.system(cmd, lines)
```

```
  -- Check for shell errors
  if vim.v.shell_error ~= 0 then
    print(result)
    return
  end

  -- Split and replace lines
  local new_lines = vim.split(result, "\n", {plain=true})
  vim.api.nvim_buf_set_lines(0, line1-1, line2, false, new_lines)
end

-- Create a command for Perl substitution
vim.api.nvim_create_user_command('S', function(opts)
  perl_substitute(opts.line1, opts.line2, opts.args)
end, {range = true, nargs = 1})
```

Source: vim.fandom.com

55.105 Use Very-Magic Regex Mode

Category: search_replace
Tags: regex, search, advanced-patterns

Enable Perl-like regex behavior with \v, making regular expressions more intuitive and less escape-heavy

Example: Example (Vim)

```
" Use very-magic mode for more natural regex
nnoremap / /\v
nnoremap ,/ /
```

Example: Example (Neovim)

```
-- Set up very-magic mode mapping
vim.keymap.set('n', '/', '/\v', { desc = 'Search with very-magic regex' })
vim.keymap.set('n', ',/', '/', { desc = 'Normal search' })
```

Source: vim.fandom.com

55.106 Word Replacement with Confirmation

Category: search_replace
Tags: search, replace, confirmation, global

Replace all occurrences of the word under the cursor with global confirmation

Example: Example (Vim)

```
nnoremap <Leader>s :%s/\<<C-r><C-w>\>//g<Left><Left>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<Leader>s', function()
  local word = vim.fn.expand('<cword>')
  vim.cmd(':%s/\<' .. word .. '\>//g')
end, { desc = 'Replace word with confirmation' })
```

Source: vim.fandom.com

55.107 Zero-Width Matching with \zs and \ze

Category: search_replace
Tags: regex, pattern-matching, zero-width

Most performant way to do lookaround matching, marking start/end of match without including those parts

Example: Example (Vim)

```
" Match 'bar' between 'foo' and 'baz'
/foo\zsbar\ze baz
```

Example: Example (Neovim)

```
-- Use similar regex pattern with Lua search functions
vim.fn.search('foo\zsbar\ze baz')
```

Source: vim.fandom.com

CHAPTER 56

Session

56.1 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: Example (Vim)

```
: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
```

Source: Community contributed

56.2 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: Example (Vim)

```
: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
```

Source: Community contributed

56.3 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: Example (Vim)

```
:wshada          " write shada file
:rshada          " read shada file
:wshada backup.shada  " save to specific file
:rshada backup.shada  " read from specific file
```

Source: Community contributed

56.4 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: Example (Vim)

```
: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
```

Source: Community contributed

56.5 Save and Restore Vim Session

Category: session-management
Tags: session, file-history, persistence

Automatically save and restore Vim session when opening and closing without arguments

Example: Example (Vim)

```
" Save session on exit
autocmd VimLeave * nested if (!isdirectory($HOME . "/.vim")) |
    \ call mkdir($HOME . "/.vim") |
    \ endif |
```

```
    \ execute "mksession! " . $HOME . "/.vim/Session.vim"

" Restore session on startup
autocmd VimEnter * nested if argc() == 0 && filereadable($HOME .
↪  "/.vim/Session.vim") |
    \ execute "source " . $HOME . "/.vim/Session.vim"
```

Example: Example (Neovim)

```
local session_file = vim.fn.expand('$HOME') .. '/.vim/Session.vim'

vim.api.nvim_create_autocmd('VimLeave', {
  nested = true,
  callback = function()
    if not vim.fn.isdirectory(vim.fn.expand('$HOME') .. '/.vim') then
      vim.fn.mkdir(vim.fn.expand('$HOME') .. '/.vim', 'p')
    end
    vim.cmd('mksession! ' .. session_file)
  end
})

vim.api.nvim_create_autocmd('VimEnter', {
  nested = true,
  callback = function()
    if vim.fn.argc() == 0 and vim.fn.filereadable(session_file) == 1 then
      vim.cmd('source ' .. session_file)
    end
  end
})
```

Source: vim.fandom.com

56.6 Session management

Category: Session
Tags: session, save, restore

Use `:mksession!` to save session and `:source Session.vim` to restore it.

Example: Example (Vim)

```
:mksession!        " save session
:source Session.vim " restore session
```

Source: Community contributed

CHAPTER 57

Syntax

57.1 Custom Syntax Highlighting for MPI/PVM Functions

Category: syntax
Tags: syntax-highlighting, programming, code-analysis

Add custom syntax highlighting for MPI and PVM function calls to improve code readability in scientific computing and parallel programming

Example: Example (Vim)

```
syn match cCommunicator "MPI_[A-Z][A-Z_a-z2 ]*("
syn match cCommunicator "MPIO_[A-Z][A-Z_a-z ]*("
syn match cCommunicator "pvm_[a-z ]*("

syn keyword cType MPI_Group MPI_Status MPI_Request MPI_Win MPI_Aint
syn keyword cType MPI_Info MPI_Op MPI_Datatype MPI_Comm

HiLink cCommunicator Communicator
```

Example: Example (Neovim)

```
-- In Neovim, use vim.treesitter or nvim-highlight for more advanced
↪  highlighting
local function setup_mpi_highlighting()
  vim.cmd([[
    syn match cCommunicator "MPI_[A-Z][A-Z_a-z2 ]*("
    syn match cCommunicator "MPIO_[A-Z][A-Z_a-z ]*("
    syn match cCommunicator "pvm_[a-z ]*("

    syn keyword cType MPI_Group MPI_Status MPI_Request MPI_Win MPI_Aint
    syn keyword cType MPI_Info MPI_Op MPI_Datatype MPI_Comm

    highlight link cCommunicator Communicator
  ]])
end

-- Optional: Set up highlighting for specific colorscheme
vim.api.nvim_create_autocmd('ColorScheme', {
  callback = setup_mpi_highlighting
})
```

Source: vim.fandom.com

57.2 Enhanced Heredoc Highlighting for INI Files

Category: syntax
Tags: highlighting, configuration, syntax-extension

Add advanced syntax highlighting support for heredoc-style parameters in INI files

Example: Example (Vim)

```
syn match dosinicomment "^;.*$\|^#.*$"
syn match dosinilabel "^\s\+ ="

" Heredoc syntax handling
syn region dosiniHereDoc matchgroup=dosiniStringStartEnd
↪  start=+<<\z(\I\i*\)+      end=+^\z1$+ contains=@dosiniInterpDQ
```

Example: Example (Neovim)

```
vim.cmd([[syn match dosinicomment "^;.*$\|^#.*$"])
vim.cmd([[syn match dosinilabel "^\s\+ ="])

-- Heredoc syntax handling via Vim command
vim.cmd([[syn region dosiniHereDoc matchgroup=dosiniStringStartEnd
↪  start=+<<\z(\I\i*\)+      end=+^\z1$+ contains=@dosiniInterpDQ]])
```

Source: vim.fandom.com

57.3 Highlight C++ Method Names in Definitions

Category: syntax
Tags: syntax-highlighting, c++, code-navigation

Adds custom syntax highlighting for C++ method names in their definitions, making code more readable

Example: Example (Vim)

```
" Add highlighting for function definition in C++
function! EnhanceCppSyntax()
  syn match cppFuncDef "::\~\?\zs\h\w*\ze([^)]*\(()\s*\(const\)?\)?\)$"
  hi def link cppFuncDef Special
endfunction

autocmd Syntax cpp call EnhanceCppSyntax()
```

Example: Example (Neovim)

```
-- Highlight C++ method names
vim.api.nvim_create_autocmd('Syntax', {
```

```
  pattern = 'cpp',
  callback = function()
    vim.fn.matchadd('Special', '::~\?\h\w*\ze([^)]*\(()\s*\(const\)?\)?\)$')
  end
})
```

Source: vim.fandom.com)

57.4 Highlight Duplicate Words in Text

Category: syntax
Tags: syntax-highlighting, error-detection, text-editing

Automatically highlight repeated words across lines to catch editing mistakes

Example: Example (Vim)

```
syn match texDoubleWord "\c\<\(\a\+\)\s\+\1\>"
hi def link texDoubleWord Error
```

Example: Example (Neovim)

```
vim.cmd([[syn match texDoubleWord "\c\<\(\a\+\)\s\+\1\>"]])
vim.cmd([[hi def link texDoubleWord Error]])
```

Source: vim.fandom.com

57.5 Highlight Python Syntax Errors

Category: syntax
Tags: python, syntax-highlighting, error-detection

Add custom syntax highlighting to detect common Python syntax mistakes like missing colons or incorrect control flow statements

Example: Example (Vim)

```
" Syntax match for Python syntax errors
syn match pythonError "^\s*\(if\|elif\)[^:]*$" display
syn match pythonError
↪  "^\s*\(class\|def\|for\|while\|try\|except\|finally\|if\|elif\|else\)$"
↪  display
```

Example: Example (Neovim)

```
-- Lua equivalent for syntax highlighting
vim.cmd([[syn match pythonError "^\s*\(if\|elif\)[^:]*$" display]])
vim.cmd([[syn match pythonError
↪  "^\s*\(class\|def\|for\|while\|try\|except\|finally\|if\|elif\|else\)$"
↪  display]])
```

Source: vim.fandom.com

57.6 Python Triple-Quoted Comments Syntax Highlighting

Category: syntax
Tags: python, syntax-highlighting, customization

Modify Python syntax highlighting to treat triple-quoted strings after a colon as comments

Example: Example (Vim)

```
syn region pythonComment
      \ start=+\%(:\n\s*\)\@<=\z('''\|"""\)+ end=+\z1+ keepend
      \ contains=pythonEscape,pythonTodo,@Spell
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.treesitter.highlighter.hl_map['comment'] = {
      start_triple_quote = function()
        -- Custom logic to highlight triple quotes as comments after colons
      end
    }
  end
})
```

Source: vim.fandom.com

CHAPTER 58

System

58.1 Async shell commands

Category: System
Tags: async, shell, lua

Use `vim.loop.spawn()` to run shell commands asynchronously without blocking Neovim.

Example: Example (Vim)

```
:lua vim.loop.spawn("ls", {args={"-la"}}, function() print("Done!") end)
```

58.2 Confirm dangerous operations

Category: System
Tags: confirm, dialog, save, quit, dangerous

Use `:confirm {command}` to show confirmation dialog for potentially dangerous operations.

Example: Example (Vim)

```
: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
```

58.3 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: Example (Vim)

```
:!ls              " run ls command
:!make            " run make command
:!!               " repeat last external command
:silent !make     " run make without showing output
```

58.4 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: Example (Vim)

```
:!mkdir newdir        " create directory
:!rm file.txt         " delete file
:!mv old.txt new.txt  " rename file
:!cp file.txt backup.txt  " copy file
```

58.5 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: Example (Vim)

```
: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
```

58.6 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: Example (Vim)

```
:shell         " start interactive shell
:cd /home/user " change to directory
:pwd           " show current directory
:lcd ~/project " change local directory for current window
```

58.7 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: Example (Vim)

```
!!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
```

58.8 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: Example (Vim)

```
: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
```

58.9 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: Example (Vim)

```
: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
```

58.10 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: Example (Vim)

```
: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 59
Tabs

59.1 Close tab

Category: Tabs
Tags: tab, close, exit

Use `:tabclose` to close current tab.

Example: Example (Vim)

```
:tabclose
```

59.2 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: Example (Vim)

```
gt   " next tab
gT   " previous tab
2gt  " go to tab 2
```

59.3 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: Example (Vim)

```
: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
```

59.4 Open new tab

Category: Tabs
Tags: tab, new, open

Use `:tabnew` or `:tabedit {file}` to open a new tab, optionally with a file.

Example: Example (Vim)

```
:tabnew
:tabedit file.txt
```

CHAPTER 60

Terminal

60.1 Capture SQL Query Output in Vim

Category: terminal
Tags: sql, command-line, external-tools

Pipe SQL query results directly into Vim for easy viewing and manipulation of database query outputs

Example: Example (Vim)

```
" MySQL CLI: Use pager to send query results to Vim
:pager vim -
```

Example: Example (Neovim)

```
-- Equivalent approach would be to use vim.fn.system() or terminal buffer
-- Example with MySQL CLI
-- In terminal: mysql -u user -p database
-- Then: pager vim -
```

Source: vim.fandom.com

60.2 Check Existing Vim Server Instances

Category: terminal
Tags: server, cli, remote-editing

List active Vim server instances to check if a server is already running

Example: Example (Vim)

```
vim --serverlist
```

Example: Example (Neovim)

```
-- Check server list
-- Note: May require external command execution
vim.fn.system('vim --serverlist')
```

Source: vim.fandom.com

60.3 Create Custom Terminal Vim Wrapper Script

Category: terminal
Tags: external-editor, terminal-integration, shell-script

Create a wrapper script to open Vim in a terminal with support for command-line arguments

Example: Example (Vim)

```
#!/bin/sh
ARGS="$@"
gnome-terminal "vim $ARGS"
```

Example: Example (Neovim)

```
-- Lua equivalent would typically be a shell script or can be implemented
↪  using vim.fn.system()
-- Example Lua wrapper function:
function OpenVimInTerminal(args)
  vim.fn.system(string.format('gnome-terminal "vim %s"', args))
end
```

Source: vim.fandom.com

60.4 Dynamically Rename Konsole Tab in Vim

Category: terminal
Tags: terminal-integration, session-management, kde

Automatically rename Konsole tab to current file name or working directory, providing context in terminal sessions

Example: Example (Vim)

```
" KDE3 Rename to current file
autocmd BufReadPost * :silent !dcop $KONSOLE_DCOP_SESSION renameSession %

" Rename to working directory when leaving Vim
autocmd VimLeavePre * :silent !dcop $KONSOLE_DCOP_SESSION renameSession $PWD
```

Example: Example (Neovim)

```
-- KDE4/5 Konsole tab renaming
vim.api.nvim_create_autocmd('BufReadPost', {
  callback = function()
    vim.fn.system('update_konsole_tab set "vim: ' .. vim.fn.expand('%:t') ..
    ↪  '"')
  end
})

vim.api.nvim_create_autocmd('VimLeavePre', {
  callback = function()
    vim.fn.system('update_konsole_tab clean')
  end
})
```

Source: vim.fandom.com

60.5 Enable Vim Server in Terminal

Category: terminal
Tags: client-server, remote-editing, vim-interaction

Add servername capability to Vim in terminal, allowing remote file editing and server interactions

Example: Example (Vim)

```
vim --servername vim
```

Example: Example (Neovim)

```
-- Check if server is running
vim.fn.serverlist()

-- Start a named server
vim.fn.servername('myserver')
```

Source: vim.fandom.com

60.6 Integrate PyUnit Testing in Vim/Neovim

Category: terminal
Tags: python, testing, quickfix, integration

Configure Vim/Neovim to run Python unit tests and navigate test results using quickfix mode

Example: Example (Vim)

```
" Set PyUnit as compiler
:compiler pyunit

" Set test runner script
:setlocal makeprg=./alltests.py
```

Example: Example (Neovim)

```
-- Set PyUnit compiler for Python files
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'python',
  callback = function()
    vim.cmd('compiler pyunit')
    vim.opt_local.makeprg = './alltests.py'
  end
})
```

Source: vim.fandom.com

60.7 Interactive Database Querying with Perl in Vim

Category: terminal
Tags: database, perl, external-integration

Use Vim's embedded Perl interpreter to connect to databases and manipulate data directly within the editor

Example: Example (Vim)

```
" connect to perl's dbi module:
:perl use dbi;
" connect to the database:
:perl $dbh = dbi->connect( "DBI:mysql:$DBNAME:$HOST",$USER,$PASSWORD,
 { raiseerror => 1});
" perform a simple query:
:perl $result = $dbh->selectall_arrayref("show tables;");
" insert the list of tables into the current buffer's top:
:perl $curbuf->Append(0, map($_->[0], @{$result}));
```

Example: Example (Neovim)

```
-- Note: Requires Vim to be compiled with Perl support
-- Use :terminal and external psql/mysql commands in Neovim
vim.api.nvim_command('!psql -d my_database -c "show tables;"')
```

Source: vim.fandom.com

60.8 Keep Terminal Open After Vim Exit

Category: terminal
Tags: terminal-integration, workflow

Modify terminal wrapper to keep the terminal window open after exiting Vim

Example: Example (Vim)

```
#!/bin/sh
ARGS="$@"
gnome-terminal "vim $ARGS; $SHELL"
```

Example: Example (Neovim)

```
-- Lua equivalent for shell script behavior
function OpenVimWithShell(args)
  vim.fn.system(string.format('gnome-terminal "vim %s; $SHELL"', args))
end
```

Source: vim.fandom.com

60.9 Open terminal in current window

Category: Terminal
Tags: terminal, open, current, window

Use `:terminal` or `:term` to open terminal in current window.

Example: Example (Vim)

```
:terminal     " open terminal in current window
:term         " shorthand for :terminal
:term bash    " open specific shell
```

Source: Community contributed

60.10 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: Example (Vim)

```
:sp | terminal   " horizontal split terminal
:vsp | terminal  " vertical split terminal
:tabe | terminal " terminal in new tab
```

Source: Community contributed

60.11 Quick IPython Integration with Vim

Category: terminal
Tags: python, integration, terminal-commands

Send commands from Vim to an IPython session using screen or terminal control

Example: Example (Vim)

```
" Save and run current buffer in IPython session
nnoremap ' :wa<CR>:!screen -x ipython_vim -X stuff $'%run "%:p"
'<CR><CR>

" Open a new terminal with IPython in screen
com OpenIPython :!konsole -e screen -S ipython_vim ipython
```

Example: Example (Neovim)

```
-- Lua equivalent for IPython integration
vim.keymap.set('n', "'", function()
  vim.cmd('wa')  -- Save all buffers
  vim.fn.system('screen -x ipython_vim -X stuff "\%run ' ..
  ↪  vim.fn.expand('%:p') .. '\n"')
end, { desc = 'Run current file in IPython' })
```

Source: vim.fandom.com

60.12 Quick Python Code Execution in Vim

Category: terminal
Tags: python, execution, quick-run

Simple mappings to quickly run Python scripts or selected code directly from Vim

Example: Example (Vim)

```
" Save and run current file
noremap <F5> <ESC>:w<CR>:silent execute "!python %"<CR><CR>

" Run selected lines in Python
vnoremap <f5> :!python<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F5>', function()
  vim.cmd.write()
  vim.cmd.silent('!python %')
end, { desc = 'Run current Python file' })

vim.keymap.set('v', '<F5>', ':!python<CR>', { desc = 'Run selected Python
↪  code' })
```

Source: vim.fandom.com

60.13 Quick SQL Query Execution in Vim

Category: terminal
Tags: sql, database, external-command

Filter selected SQL through an external database command and replace text with query results

Example: Example (Vim)

```
:!psql -d my_database -f %
```

Example: Example (Neovim)

```
-- Execute SQL file through psql
vim.api.nvim_command('!psql -d my_database -f ' .. vim.fn.expand('%:p'))
```

Source: vim.fandom.com

60.14 Quick Shell Command Switching with Ctrl-Z

Category: terminal
Tags: shell-interaction, command-line, workflow

Use Ctrl-Z to suspend Vim and return to shell, then 'fg' to resume editing. Useful for quickly running shell commands without opening multiple terminal windows.

Example: Example (Vim)

```
# Shell feature, not Vimscript specific
# Ctrl-Z to suspend
# fg to return to Vim
```

Example: Example (Neovim)

```
-- Lua can't directly implement shell suspend, but demonstrates shell
↪  interaction
-- Use :terminal or external shell commands
vim.keymap.set('n', '<leader>sh', ':terminal<CR>', { desc = 'Open terminal'
↪  })
```

Source: vim.fandom.com

60.15 Quick Terminal Window Maximize/Restore

Category: terminal
Tags: terminal, window-management

Simple terminal control sequences to maximize and restore window size

Example: Example (Vim)

```
" Maximize window
echo -ne '\e[9;1t'

" Restore window
echo -ne '\e[9;0t'
```

Example: Example (Neovim)

```
-- Maximize window
vim.fn.system('echo -ne "\e[9;1t"')

-- Restore window
vim.fn.system('echo -ne "\e[9;0t"')
```

Source: vim.fandom.com

60.16 Quick Version Control Commands in Vim

Category: terminal
Tags: git, version-control, command-line

Create shortcut commands for running version control operations directly from Vim

Example: Example (Vim)

```
command! -complete=file -nargs=* Git call s:RunShellCommand('git '.<q-args>)
command! -complete=file -nargs=* Svn call s:RunShellCommand('svn '.<q-args>)
command! -complete=file -nargs=* Hg call s:RunShellCommand('hg '.<q-args>)
```

Example: Example (Neovim)

```
local function run_vcs_command(vcs, args)
  vim.api.nvim_create_user_command(vcs:upper(), function(opts)
    local full_cmd = string.format('%s %s', vcs, opts.args)
    vim.fn.system(full_cmd)
    print(full_cmd .. ' executed')
  end, { nargs = '*', complete = 'file' })
end

run_vcs_command('git')
run_vcs_command('svn')
run_vcs_command('hg')
```

Source: vim.fandom.com

60.17 Run External Commands Asynchronously in Windows

Category: terminal
Tags: external-commands, async, windows

Run console or GUI applications asynchronously without blocking Vim, with options to pause or keep the window open

Example: Example (Vim)

```
" Run compiled program for current source file
nnoremap <silent> <F5> :!start cmd /c "%:p:r:s,$,.exe," & pause<CR>
```

Example: Example (Neovim)

```
-- Run compiled program asynchronously
vim.keymap.set('n', '<F5>', function()
  vim.cmd('!start cmd /c "' .. vim.fn.expand('%:p:r') .. '.exe" & pause')
end, { silent = true })
```

Source: vim.fandom.com

60.18 Run MATLAB Scripts from Vim

Category: terminal
Tags: matlab, python-integration, external-execution

Execute MATLAB scripts directly from Vim using Python and Win32 COM interface

Example: Example (Vim)

```
if has("win32")
  :py from win32com.client import Dispatch
  :py import vim
  :py h=Dispatch('matlab.application')
  map <buffer> ,r :w<CR>:py print h.Execute(vim.eval('expand("%:t:r")'))<CR>
endif
```

Example: Example (Neovim)

```
if vim.fn.has('win32') == 1 then
  local dispatch = require('pythonx.win32com.client').Dispatch
  local h = dispatch('matlab.application')
  vim.keymap.set('n', ',r', function()
    vim.cmd('write')
    local script_name = vim.fn.expand('%:t:r')
    print(h:Execute(script_name))
  end, { buffer = true })
end
```

Source: vim.fandom.com

60.19 Run Shell Commands in Vim Scratch Buffer

Category: terminal
Tags: shell-command, command-line, buffer-management

Create a custom command to run shell commands in a new scratch buffer, allowing easy command execution and output viewing within Vim

Example: Example (Vim)

```
command! -complete=shellcmd -nargs=+ Shell call s:RunShellCommand(<q-args>)

function! s:RunShellCommand(cmdline)
  botright new
  setlocal buftype=nofile bufhidden=wipe nobuflisted noswapfile nowrap
  call setline(1, 'You entered:    ' . a:cmdline)
  execute '$read !'. a:cmdline
  setlocal nomodifiable
  1
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Shell', function(opts)
  local cmdline = opts.args
  local bufnr = vim.api.nvim_create_buf(false, true)
  vim.api.nvim_command('botright split')
  vim.api.nvim_win_set_buf(0, bufnr)
```

```
  vim.api.nvim_buf_set_option(bufnr, 'buftype', 'nofile')
  vim.api.nvim_buf_set_option(bufnr, 'bufhidden', 'wipe')
  vim.api.nvim_buf_set_option(bufnr, 'swapfile', false)

  vim.api.nvim_buf_set_lines(bufnr, 0, -1, false, {'You entered: ' ..
  ↪  cmdline})

  local output = vim.fn.systemlist(cmdline)
  vim.api.nvim_buf_set_lines(bufnr, 1, -1, false, output)
  vim.api.nvim_buf_set_option(bufnr, 'modifiable', false)
  vim.api.nvim_win_set_cursor(0, {1, 0})
end, {nargs = '+', complete = 'shellcmd'})
```

Source: vim.fandom.com

60.20 Screen Session Vim Integration

Category: terminal
Tags: screen, multi-terminal, workflow

Open Vim inside a screen session with a small delay to prevent file deletion issues

Example: Example (Vim)

```
" Bash script for screen vim integration
screen -X screen /usr/bin/vim $@
sleep .2 # Prevent file deletion
```

Example: Example (Neovim)

```
-- Lua equivalent using vim.fn and os.execute
local function open_vim_in_screen(file)
  os.execute('screen -X screen /usr/bin/vim ' .. file)
  vim.fn.system('sleep 0.2')  -- Small delay
end
```

Source: vim.fandom.com

60.21 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: Example (Vim)

```
:lua vim.api.nvim_chan_send(terminal_job_id, "ls\n")
```

Source: Community contributed

60.22 Smart Vim Server Function for Shell

Category: terminal
Tags: shell-function, client-server, workflow

Intelligent shell function to always use a single Vim server, creating one if none exists

Example: Example (Vim)

```
function! VimServerWrapper()
  if serverlist() == ''
    call server2client('vim --servername VIM')
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.smart_vim_server()
  local servers = vim.fn.serverlist()
  if servers == '' then
    vim.cmd('vim --servername VIM')
  end
end
```

Source: vim.fandom.com

60.23 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: Example (Vim)

```
" From terminal normal mode:
i   " enter terminal mode at cursor
a   " enter terminal mode after cursor
A   " enter terminal mode at end of line
```

Source: Community contributed

60.24 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: Example (Vim)

```
" In terminal mode:
Ctrl+\ Ctrl+o  " execute one normal mode command
```

Source: Community contributed

60.25 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: Example (Vim)

```
" In terminal mode:
Ctrl+\ Ctrl+n  " exit to normal mode
```

Source: Community contributed

60.26 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: Example (Vim)

```
" In terminal mode:
ls<Enter>           " sent to terminal
Ctrl+c              " sent to terminal (interrupt)
Ctrl+\ Ctrl+n       " Neovim command (exit to normal)
```

Source: Community contributed

60.27 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: Example (Vim)

```
" In terminal normal mode (after Ctrl+\ Ctrl+n):
gg    " go to top of scrollback
G     " go to bottom
/text " search in terminal output
```

Source: Community contributed

60.28 tmux-Compatible Cursor Configuration

Category: terminal
Tags: tmux, cursor, terminal-config

Special configuration for cursor appearance and color in tmux environment

Example: Example (Vim)

```
if exists('$TMUX')
  " set insert mode to cyan vertical line
  let &t_SI .= "\<esc>Ptmux;\<esc>\<esc>[6 q\<esc>\\"
  let &t_SI .= "\<esc>Ptmux;\<esc>\<esc>]12;cyan\x7\<esc>\\"

  " set normal mode to green block
  let &t_EI .= "\<esc>Ptmux;\<esc>\<esc>[2 q\<esc>\\"
  let &t_EI .= "\<esc>Ptmux;\<esc>\<esc>]12;green\x7\<esc>\\"
endif
```

Example: Example (Neovim)

```
if vim.env.TMUX then
  local function tmux_cursor_style()
    -- Insert mode: cyan vertical line
    vim.cmd('let &t_SI .= "\\<esc>Ptmux;\\<esc>\\<esc>[6 q\\<esc>\\\\"')
    vim.cmd('let &t_SI .=
    ↪  "\\<esc>Ptmux;\\<esc>\\<esc>]12;cyan\\x7\\<esc>\\\\"')

    -- Normal mode: green block
    vim.cmd('let &t_EI .= "\\<esc>Ptmux;\\<esc>\\<esc>[2 q\\<esc>\\\\"')
    vim.cmd('let &t_EI .=
    ↪  "\\<esc>Ptmux;\\<esc>\\<esc>]12;green\\x7\\<esc>\\\\"')
  end
```

```
  tmux_cursor_style()
end
```

Source: vim.fandom.com

CHAPTER 61
Text manipulation

61.1 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: Example (Vim)

```
" Select lines with numbers, then:
:'<,'>s/\(\d\+\)\.\(\d\+\)/\=printf("%6.2f", submatch(0))/
" Or use Align plugin:
:'<,'>Align \.
```

61.2 Binary number operations

Category: Text Manipulation
Tags: binary, numbers, conversion, base

Convert and manipulate binary numbers using expressions and external tools.

Example: Example (Vim)

```
" 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
```

61.3 Comment and uncomment blocks

Category: Text Manipulation
Tags: comment, uncomment, code, blocks

Add or remove comment markers from blocks of code.

Example: Example (Vim)

```
" 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
```

61.4 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: Example (Vim)

```
:retab              " convert tabs to spaces
:set expandtab | retab  " convert and set expandtab
```

61.5 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: Example (Vim)

```
" Select multiple lines with numbers, then:
g<C-a>    " creates 1,2,3,4... sequence
<C-a>     " would increment all by 1
```

61.6 Delete blank lines

Category: Text Manipulation
Tags: delete, blank, lines

Use `:g/^$/d` to delete all blank/empty lines in the buffer.

Example: Example (Vim)

```
:g/^$/d  " delete blank lines
```

61.7 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: Example (Vim)

```
x   " delete character under cursor
X   " delete character before cursor
5x  " delete 5 characters forward
```

61.8 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: Example (Vim)

```
:v/TODO/d  " delete lines that don't contain 'TODO'
```

61.9 Duplicate lines or selections

Category: Text Manipulation
Tags: duplicate, copy, lines, repeat

Duplicate current line or selected text efficiently.

Example: Example (Vim)

```
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
```

61.10 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: Example (Vim)

```
!}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
```

61.11 Format paragraph

Category: Text Manipulation
Tags: format, paragraph, wrap

Use `gqap` to format/wrap a paragraph according to textwidth.

Example: Example (Vim)

```
gqap  " format around paragraph
```

61.12 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: Example (Vim)

```
" 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
```

61.13 Handle common typos

Category: Text Manipulation
Tags: typos, abbreviations, correction, auto

Create abbreviations to automatically fix common typing mistakes.

Example: Example (Vim)

```
iab teh the
iab adn and
iab recieve receive
iab seperate separate
iab definately definitely
```

61.14 Increment/decrement numbers

Category: Text Manipulation
Tags: increment, decrement, numbers, math

Modify numbers in text using increment and decrement operations.

Example: Example (Vim)

```
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
```

61.15 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: Example (Vim)

```
Ctrl+V  " start visual block
I       " insert at beginning of block
```

```
text    " type text to insert
Esc     " apply to all selected lines
```

61.16 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: Example (Vim)

```
:put =range(1,10)  " insert numbers 1-10
" Or for selected lines:
:'<,'>s/^/\=line('.')-line("'<")+1.'. '/
```

61.17 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: Example (Vim)

```
:'<,'>s/\n/, /g     " join lines with ", "
:'<,'>s/\n/ | /g    " join lines with " | "
```

61.18 Lowercase/uppercase current line

Category: Text Manipulation
Tags: case, line, transform

Use `guu` to lowercase current line or `gUU` to uppercase current line.

Example: Example (Vim)

```
guu  " lowercase current line
gUU  " uppercase current line
```

61.19 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: Example (Vim)

```
p  " put text after cursor
P  " put text before cursor
"ap " put from register 'a' after cursor
```

61.20 ROT13 encoding

Category: Text Manipulation
Tags: rot13, encoding, cipher, transform

Apply ROT13 cipher to selected text using `g?` operator or external command.

Example: Example (Vim)

```
g??         " ROT13 current line
g?ap        " ROT13 around paragraph
:'<,'>!tr 'A-Za-z' 'N-ZA-Mn-za-m'  " ROT13 using external tr
```

61.21 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: Example (Vim)

```
:%!sort -u          " sort and remove duplicates
:sort u             " vim's built-in sort unique
" Or manually:
:g/^\(.*\)$\n\1$/d  " remove consecutive duplicates
```

61.22 Remove trailing whitespace

Category: Text Manipulation
Tags: whitespace, trailing, clean

Use `:%s/\s\+$//` to remove trailing whitespace from all lines.

Example: Example (Vim)

```
:%s/\s\+$//  " remove trailing whitespace
```

61.23 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: Example (Vim)

```
R    " enter Replace mode
ra   " replace character under cursor with 'a'
3rx  " replace 3 characters with 'x'
```

61.24 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: Example (Vim)

```
:g/^/m0              " reverse all lines
:'<,'>g/^/m'<-1      " reverse selected lines
```

61.25 Text alignment and padding

Category: Text Manipulation
Tags: align, pad, format, columns, spacing

Align text in columns and add padding for better formatting.

Example: Example (Vim)

```
" 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
```

61.26 Text statistics

Category: Text Manipulation
Tags: statistics, analysis, count, metrics

Get detailed text statistics including character, word, and line counts.

Example: Example (Vim)

```
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
```

61.27 Transpose characters

Category: Text Manipulation
Tags: transpose, swap, characters, exchange

Swap adjacent characters or transpose text elements efficiently.

Example: Example (Vim)

```
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
```

61.28 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: Example (Vim)

```
u      " undo last change
Ctrl+r " redo (undo the undo)
U      " undo all changes on current line
```

61.29 Uppercase current word

Category: Text Manipulation
Tags: case, word, uppercase

Use `gUw` to uppercase current word.

Example: Example (Vim)

```
gUw  " uppercase current word
```

61.30 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: Example (Vim)

```
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
```

61.31 Work with CSV files

Category: Text Manipulation
Tags: csv, columns, data, tabular

Use CSV plugin commands to navigate and manipulate comma-separated data.

Example: Example (Vim)

```
" 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 62

Text objects

62.1 Select around parentheses

Category: Text Objects
Tags: select, parentheses, around

Use `va(` to select text around parentheses (including the parentheses).

Example: Example (Vim)

```
va(  " select around parentheses
```

62.2 Select inside quotes

Category: Text Objects
Tags: select, quotes, inside

Use `vi"` to select text inside double quotes or `vi'` for single quotes.

Example: Example (Vim)

```
vi"  " select inside double quotes
vi'  " select inside single quotes
```

62.3 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: Example (Vim)

```
cit  " change inside HTML tag
dat  " delete around HTML tag
yit  " yank inside tag content
vat  " select around tag including markup
```

62.4 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: Example (Vim)

```
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{
```

62.5 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: Example (Vim)

```
ci<  " change inside angle brackets
da>  " delete around angle brackets
vi<  " select inside angle brackets
```

62.6 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: Example (Vim)

```
ca(  " change around parentheses
da[  " delete around square brackets
ya{  " yank around curly braces
```

62.7 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: Example (Vim)

```
ci(  " change inside parentheses
di[  " delete inside square brackets
yi{  " yank inside curly braces
```

62.8 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: Example (Vim)

```
ci"  " change inside double quotes
da'  " delete around single quotes
yi`  " yank inside backticks
```

62.9 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: Example (Vim)

```
cis  " change inside sentence
das  " delete around sentence
vip  " select inside paragraph
yap  " yank around paragraph
```

62.10 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: Example (Vim)

```
ci[  " change inside square brackets
da]  " delete around square brackets
yi[  " yank inside square brackets
```

62.11 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: Example (Vim)

```
ciw  " change inside word
daw  " delete around word (includes space)
yiW  " yank inside WORD (space-separated)
```

62.12 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: Example (Vim)

```
=ap  " format around paragraph
>i{  " indent inside curly braces
<as  " unindent around sentence
```

CHAPTER 63

Treesitter

63.1 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: Example (Neovim)

```
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, {})
```

63.2 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: Example (Neovim)

```
-- 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, {})
```

63.3 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: Example (Neovim)

```
-- 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"})
```

63.4 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: Example (Neovim)

```
-- 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
```

63.5 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: Example (Neovim)

```
-- 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
```

63.6 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: Example (Neovim)

```
-- 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
```

63.7 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: Example (Neovim)

```
-- 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
```

63.8 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: Example (Neovim)

```
-- 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))
]])
```

63.9 Highlight custom patterns with treesitter

Category: Treesitter
Tags: treesitter, highlight, queries, custom

Use treesitter queries to add custom syntax highlighting beyond default parsers.

Example: Example (Neovim)

```
-- 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,
})
```

63.10 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: Example (Neovim)

```
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
```

63.11 Treesitter folding

Category: Treesitter
Tags: treesitter, folding, code, structure

Set `foldmethod=expr` and `foldexpr=nvim_treesitter#foldexpr()` to use treesitter-based folding.

Example: Example (Vim)

```
:set foldmethod=expr
:set foldexpr=nvim_treesitter#foldexpr()
```

63.12 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: Example (Vim)

```
Ctrl-space  " start/expand selection
Ctrl-x      " shrink selection (if configured)
```

63.13 Treesitter install parser

Category: Treesitter
Tags: treesitter, install, parser, language

Use `:TSInstall <language>` to install treesitter parser for a specific language.

Example: Example (Vim)

```
:TSInstall lua        " install Lua parser
:TSInstall javascript " install JavaScript parser
:TSInstall all        " install all maintained parsers
```

63.14 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: Example (Vim)

```
]f  " next function
[f  " previous function
]c  " next class
[c  " previous class
```

63.15 Treesitter playground

Category: Treesitter
Tags: treesitter, playground, debug, explore

Use `:TSPlaygroundToggle` to open treesitter playground for exploring syntax tree interactively.

Example: Example (Vim)

```
:TSPlaygroundToggle  " toggle treesitter playground
```

63.16 Treesitter query predicates and directives

Category: Treesitter
Tags: treesitter, queries, predicates, advanced

Use query predicates to add conditions to treesitter pattern matching.

Example: Example (Neovim)

```
-- 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
```

63.17 Treesitter swap nodes

Category: Treesitter
Tags: treesitter, swap, parameters, arguments

Use treesitter to swap function parameters or other syntax nodes using configured keymaps.

Example: Example (Vim)

```
gs  " swap with next parameter/node
gS  " swap with previous parameter/node
```

63.18 Use treesitter for smart text editing

Category: Treesitter
Tags: treesitter, editing, smart, refactoring

Leverage treesitter for intelligent code refactoring and editing operations.

Example: Example (Neovim)

```
-- 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"})
```

63.19 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: Example (Neovim)

```
-- 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 64

Ui

64.1 Always Show Statusline in Single Window

Category: ui
Tags: statusline, ui-configuration

Ensure the statusline is always visible, even in single window mode

Example: Example (Vim)

```
set laststatus=2
```

Example: Example (Neovim)

```
vim.opt.laststatus = 2
```

Source: vim.fandom.com

64.2 Beautiful transparent theme with background image

Category: UI
Tags: theme, transparency, background, aesthetic, kitty, tokyonight

Create a beautiful working environment by combining a transparent Neovim theme with a tinted background image in your terminal emulator. **Neovim configuration (Tokyonight with transparency): Kitty terminal configuration (~/.config/kitty/kitty.conf):** This creates a beautiful aesthetic where your code appears over a subtly tinted background image, combining terminal customization with Neovim's transparent theme support.

Example: Example (Neovim)

```
{
  "folke/tokyonight.nvim",
  lazy = false,
  priority = 2000,
  opts = { transparent = true }  -- Enable transparency
}
```

Source: Community contributed

64.3 Change Cursor Color and Shape in Different Modes

Category: ui
Tags: cursor, terminal-config, mode-specific

Customize cursor color and shape to visually distinguish between Vim modes, improving user awareness

Example: Example (Vim)

```
if &term =~ "xterm\|rxvt"
  " use an orange cursor in insert mode
  let &t_SI = "\<Esc>]12;orange\x7"
  " use a red cursor otherwise
  let &t_EI = "\<Esc>]12;red\x7"
endif

" Cursor shape changes
if &term =~ '^xterm\|rxvt'
  let &t_SI .= "\<Esc>[4 q"  " solid underscore in insert mode
  let &t_EI .= "\<Esc>[2 q"  " solid block in normal mode
endif
```

Example: Example (Neovim)

```
local function set_cursor_style()
  if vim.env.TERM:match("xterm") or vim.env.TERM:match("rxvt") then
    -- Cursor color changes
    vim.cmd('let &t_SI = "\\<Esc>]12;orange\\x7"')
    vim.cmd('let &t_EI = "\\<Esc>]12;red\\x7"')

    -- Cursor shape changes
    vim.cmd('let &t_SI .= "\\<Esc>[4 q"')  -- Underscore in insert mode
    vim.cmd('let &t_EI .= "\\<Esc>[2 q"')  -- Block in normal mode
  end
end

vim.api.nvim_create_autocmd('VimEnter', {
  callback = set_cursor_style
})
```

Source: vim.fandom.com

64.4 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: Example (Vim)

```
:hi Comment guifg=#ffaa00 gui=italic
```

Source: Community contributed

64.5 Check highlight groups

Category: UI
Tags: highlight, groups, colors

Use `:hi` to view all highlight groups and their current settings.

Example: Example (Vim)

```
:hi  " show all highlight groups
```

Source: Community contributed

64.6 Color Scheme for Different Directories

Category: ui
Tags: color-scheme, autocmd, file-context

Change color scheme based on the current file's directory to quickly identify which project or directory you're working in

Example: Example (Vim)

```
autocmd BufEnter * if match(@%,'/otherdir/')>=0 | colorscheme oceanblack |
↪  else | colorscheme inkpot | end
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('BufEnter', {
  callback = function()
    local current_file = vim.fn.expand('%:p')
    if current_file:match('/otherdir/') then
      vim.cmd('colorscheme oceanblack')
    else
      vim.cmd('colorscheme inkpot')
    end
  end
})
```

Source: vim.fandom.com

64.7 Console-Like Fonts for Neovim on Windows

Category: ui
Tags: fonts, display, windows

Recommended console-style fonts for improving Neovim's visual appearance on Windows

Example: Example (Neovim)

```
-- Recommended fonts in Neovim configuration
-- Add to init.lua or init.vim
vim.opt.guifont = 'FixedSys Excelsior:h10'  -- Unicode-friendly console font
-- Or
vim.opt.guifont = 'ProFont:h10'  -- Another recommended console-style font
```

Source: vim.fandom.com

64.8 Cross-Platform Font Size Management

Category: ui
Tags: cross-platform, font-size, gui

Robust multiplatform implementation for font size adjustment supporting Linux, Windows, and macOS

Example: Example (Vim)

```
function! GetFontInfo()
  let l:font_name_pattern = '\v(^.{-1,})( \d+$)@='
  let l:font_size_pattern = '\v(\d+$)'
  let l:font_name = matchstr(&guifont, l:font_name_pattern)
  let l:font_size = matchstr(&guifont, l:font_size_pattern)
  return { 'name': l:font_name, 'size': l:font_size }
endfunction
```

Example: Example (Neovim)

```
local function get_font_info()
  local guifont = vim.o.guifont
  local font_name = guifont:match('^(.+)%s*%d+$')
  local font_size = guifont:match('(%d+)$')

  return {
    name = font_name or '',
    size = font_size or ''
  }
end

-- Optional mouse wheel zoom
```

```
vim.keymap.set('n', '<C-ScrollWheelUp>', function()
  local font_info = get_font_info()
  -- Implement font size increase logic
end)

vim.keymap.set('n', '<C-ScrollWheelDown>', function()
  local font_info = get_font_info()
  -- Implement font size decrease logic
end)
```

Source: vim.fandom.com

64.9 Custom statusline

Category: UI
Tags: statusline, custom, display

Use `vim.opt.statusline` to set a custom statusline format.

Example: Example (Vim)

```
:lua vim.opt.statusline = "%f %y %m %= %l:%c"
```

Source: Community contributed

64.10 Customize Mouse Cursor Shape in Vim

Category: ui
Tags: mouse, cursor, customization

Change the mouse cursor shape in different Vim modes for improved visual feedback

Example: Example (Vim)

```
:set mouseshape=n:pencil

" For different modes, you can use various shapes
" Check :help 'mouseshape' for more options
```

Example: Example (Neovim)

```
-- Note: This is X11/Linux specific
-- Neovim doesn't have direct mouseshape setting
-- May require external configuration or GUI settings
```

Source: vim.fandom.com

64.11 Customize Omni-Completion Popup Menu Colors

Category: ui
Tags: completion, color-scheme, customization

Improve readability of the completion popup menu by customizing its background and style

Example: Example (Vim)

```
" GUI mode color customization
highlight Pmenu guibg=brown gui=bold

" Console Vim color customization
highlight Pmenu ctermbg=238 gui=bold
```

Example: Example (Neovim)

```
-- Customize completion menu colors
vim.cmd.highlight('Pmenu guibg=brown gui=bold')
vim.cmd.highlight('Pmenu ctermbg=238 gui=bold')
```

Source: vim.fandom.com

64.12 Customize Toolbar for Quick Launching

Category: ui
Tags: toolbar, customization, windows

Replace existing toolbar icon to launch explorer with selected URL/path

Example: Example (Vim)

```
vnoremenu 1.140 ToolBar.New "wy:!start explorer <C-R>w<CR>
```

Example: Example (Neovim)

```
vim.cmd('vnoremenu 1.140 ToolBar.New "wy:!start explorer <C-R>w<CR>')
```

Source: vim.fandom.com

64.13 Customize Vim Statusline with Dynamic Information

Category: ui
Tags: statusline, customization, ui-enhancement

Create a comprehensive statusline that displays file details, cursor position, and times-

tamp dynamically

Example: Example (Vim)

```
set statusline=%F%m%r%h%w\ [FORMAT=%{&ff}]\ [TYPE=%Y]\ [POS=%l,%v][%p%%]\
↪  %{strftime("%d/%m/%y\ -\ %H:%M")}
```

Example: Example (Neovim)

```
vim.opt.statusline = table.concat({
  '%F%m%r%h%w',
  ' [FORMAT=%{&ff}]',
  ' [TYPE=%Y]',
  ' [POS=%l,%v][%p%%]',
  ' %{strftime("%d/%m/%y - %H:%M")}'
}, '')
```

Source: vim.fandom.com

64.14 Disable Cursor Blinking in Vim/Neovim

Category: ui
Tags: cursor, display-config

Disable cursor blinking to reduce visual distraction and improve focus

Example: Example (Vim)

```
" Disable all blinking:
:set guicursor+=a:blinkon0

" Restore default:
:set guicursor&
```

Example: Example (Neovim)

```
-- Disable cursor blinking
vim.opt.guicursor:append('a:blinkon0')

-- Alternatively, to restore default
vim.opt.guicursor = vim.o.guicursor
```

Source: vim.fandom.com

64.15 Disable Mouse Interaction in Vim

Category: ui
Tags: mouse, configuration

Completely disable mouse interaction in Vim to prevent unintended mode switches or selections

Example: Example (Vim)

```
set mouse=
```

Example: Example (Neovim)

```
vim.opt.mouse = ''
```

Source: vim.fandom.com

64.16 Display Non-Native File Format in Statusline

Category: ui
Tags: statusline, file-format, customization

Add a custom function to highlight non-native file formats in the statusline, making it easy to identify files with different line endings

Example: Example (Vim)

```
function ShowFileFormatFlag(var)
  if ( a:var == 'dos' )
    return '[dos]'
  elseif ( a:var == 'mac' )
    return '[mac]'
  else
    return ''
  endif
endfunction

hi User1 term=bold cterm=bold ctermfg=red ctermbg=darkblue

set statusline+=%1*%{ShowFileFormatFlag(&fileformat)}%*
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('ShowFileFormatFlag', function()
  local fileformat = vim.o.fileformat
  if fileformat == 'dos' then
    return '[dos]'
  elseif fileformat == 'mac' then
    return '[mac]'
  else
    return ''
  end
end, {})
```

```
vim.cmd.highlight('User1 term=bold cterm=bold ctermfg=red ctermbg=darkblue')

vim.o.statusline = vim.o.statusline .. '%1*%{v:lua.ShowFileFormatFlag()}%*'
```

Source: vim.fandom.com

64.17 Dynamic Cursor Shape in Different Modes

Category: ui
Tags: terminal-integration, cursor-customization, mode-indicator

Change cursor shape dynamically based on Vim mode, providing visual feedback for insert, replace, and normal modes

Example: Example (Vim)

```
" Cursor shape settings
let &t_SI = "\<Esc>[6 q"  " Insert mode: vertical bar
let &t_SR = "\<Esc>[4 q"  " Replace mode: underline
let &t_EI = "\<Esc>[2 q"  " Normal mode: block
```

Example: Example (Neovim)

```
-- Cursor shape configuration
vim.cmd([[let &t_SI = "\<Esc>[6 q"  " Insert mode: vertical bar
let &t_SR = "\<Esc>[4 q"  " Replace mode: underline
let &t_EI = "\<Esc>[2 q"  " Normal mode: block]])
```

Source: vim.fandom.com

64.18 Dynamically Adjust Font Size in GUI

Category: ui
Tags: font-size, gui, configuration

Provides a flexible function to incrementally adjust font size in GUI Vim/Neovim with min and max size constraints

Example: Example (Vim)

```
function! AdjustFontSize(amount)
  if has("gui_gtk2") && has("gui_running")
    let fontname = substitute(&guifont, s:pattern, '\1', '')
    let cursize = substitute(&guifont, s:pattern, '\2', '')
    let newsize = cursize + a:amount
    if (newsize >= s:minfontsize) && (newsize <= s:maxfontsize)
      let newfont = fontname . newsize
      let &guifont = newfont
```

```
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.adjust_font_size(amount)
  if vim.fn.has('gui_running') == 1 then
    local guifont = vim.o.guifont
    local font_name = guifont:match('^(.*):h')
    local current_size = tonumber(guifont:match(':h(%d+)'))
    local new_size = current_size + amount

    if new_size >= 6 and new_size <= 16 then
      vim.o.guifont = string.format('%s:h%d', font_name, new_size)
    end
  end
end

-- Optional mappings
vim.keymap.set('n', '<C-Up>', function() _G.adjust_font_size(1) end)
vim.keymap.set('n', '<C-Down>', function() _G.adjust_font_size(-1) end)
```

Source: vim.fandom.com

64.19 Dynamically Switch Vim Color Schemes

Category: ui
Tags: color-scheme, key-mapping, customization

Easily cycle through available color schemes with keyboard shortcuts, including random selection and time-based color scheme switching

Example: Example (Vim)

```
" Key mappings for color scheme switching
nnoremap <F8> :call NextColor(1)<CR>
nnoremap <S-F8> :call NextColor(-1)<CR>
nnoremap <A-F8> :call NextColor(0)<CR>
```

Example: Example (Neovim)

```
-- Color scheme switching in Neovim
vim.keymap.set('n', '<F8>', function()
  vim.fn.NextColor(1)
end, { desc = 'Next color scheme' })

vim.keymap.set('n', '<S-F8>', function()
  vim.fn.NextColor(-1)
end, { desc = 'Previous color scheme' })
```

```
vim.keymap.set('n', '<A-F8>', function()
  vim.fn.NextColor(0)
end, { desc = 'Random color scheme' })
```

Source: vim.fandom.com

64.20 Flash 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: Example (Vim)

```
:au TextYankPost *
↪  lua=vim.highlight.on_yank{higroup='IncSearch',timeout=200}
```

Source: Community contributed

64.21 Full Mouse Support in Console Vim

Category: ui
Tags: mouse, terminal, interaction

Enable full mouse support in console Vim, including scrolling and clicking, making terminal Vim more interactive

Example: Example (Vim)

```
set mouse=a
```

Example: Example (Neovim)

```
vim.opt.mouse = 'a'
```

Source: vim.fandom.com

64.22 Hide/Toggle GUI Elements in Neovim

Category: ui
Tags: gui, interface, customization

Easily hide or toggle menu bar, toolbar, and scrollbars to maximize screen space and

simplify the interface

Example: Example (Vim)

```
" Remove GUI elements
set guioptions-=m  " remove menu bar
set guioptions-=T  " remove toolbar
set guioptions-=r  " remove right-hand scroll bar
set guioptions-=L  " remove left-hand scroll bar

" Toggle menu bar with Ctrl-F1
nnoremap <C-F1> :if &go=~#'m'<Bar>set go-=m<Bar>else<Bar>set
↪  go+=m<Bar>endif<CR>
```

Example: Example (Neovim)

```
-- Remove GUI elements
vim.o.guioptions = vim.o.guioptions:gsub('m', '')  -- remove menu bar
vim.o.guioptions = vim.o.guioptions:gsub('T', '')  -- remove toolbar
vim.o.guioptions = vim.o.guioptions:gsub('r', '')  -- remove right scrollbar
vim.o.guioptions = vim.o.guioptions:gsub('L', '')  -- remove left scrollbar

-- Toggle menu bar with Ctrl-F1
vim.keymap.set('n', '<C-F1>', function()
  local go = vim.o.guioptions
  vim.o.guioptions = go:find('m') and go:gsub('m', '') or go .. 'm'
end, { desc = 'Toggle menu bar' })
```

Source: vim.fandom.com

64.23 Highlight Inserted Text in Insert Mode

Category: ui
Tags: highlight, insert-mode, visual-feedback

Dynamically highlight text as you type in insert mode, providing visual feedback and helping track text insertion

Example: Example (Vim)

```
highlight Inserted ctermfg=red guifg=red
function! s:AddHighlight() abort
  let [_, lnum, col; rest] = getpos('.')
  let w:insert_hl = matchadd('Inserted', '\%'.col.'c\%'.lnum.'l\_.*\%#')
endfunction

function! s:DeleteHighlight() abort
  if exists('w:insert_hl')
      call matchdelete(w:insert_hl)
  endif
endfunction
```

```
augroup InsertHL
  autocmd!
  autocmd InsertEnter * call s:AddHighlight()
  autocmd InsertLeave * call s:DeleteHighlight()
augroup END
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('InsertHL', { clear = true })

vim.api.nvim_create_autocmd('InsertEnter', {
  group = 'InsertHL',
  callback = function()
    local col = vim.fn.col('.')
    local row = vim.fn.line('.')
    vim.api.nvim_buf_add_highlight(0, -1, 'Inserted', row-1, col-1, -1)
  end
})

vim.api.nvim_create_autocmd('InsertLeave', {
  group = 'InsertHL',
  callback = function()
    vim.api.nvim_buf_clear_namespace(0, -1, 0, -1)
  end
})

vim.cmd('highlight Inserted ctermfg=red guifg=red')
```

Source: vim.fandom.com

64.24 Highlight Matching Braces Instantly

Category: ui
Tags: highlight, editing

Configure instant brace matching with configurable display time

Example: Example (Vim)

```
set showmatch
set matchtime=3  " Highlight for 300ms
```

Example: Example (Neovim)

```
vim.opt.showmatch = true
vim.opt.matchtime = 3
```

Source: vim.fandom.com

64.25 Highlight groups

Category: UI
Tags: highlight, groups, fun

Use the following code to create command `HLList`. When run, the command creates a scratch buffer with one line per highlight group, with each line styled with its own group.

Example: Example (Vim)

```
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
```

Source: Community contributed

64.26 Identify Active Vim Instance by Color

Category: ui
Tags: focus, colorscheme, multi-instance

Change color scheme when Vim gains or loses focus to visually distinguish active window

Example: Example (Vim)

```
" Change color scheme on window focus
:autocmd FocusLost * :colorscheme desert
:autocmd FocusGained * :colorscheme default
```

Example: Example (Neovim)

```
vim.api.nvim_create_augroup('VimFocusColorScheme', { clear = true })

vim.api.nvim_create_autocmd('FocusLost', {
  group = 'VimFocusColorScheme',
  callback = function()
    vim.cmd('colorscheme desert')
  end
})

vim.api.nvim_create_autocmd('FocusGained', {
  group = 'VimFocusColorScheme',
  callback = function()
    vim.cmd('colorscheme default')
  end
})
```

Source: vim.fandom.com

64.27 Mouse and UI Enhancements

Category: ui
Tags: interface, mouse-support, display

Enable mouse support and improve user interface with minimal configuration

Example: Example (Vim)

```
" UI and mouse settings
if has('mouse')
  set mouse=a
endif
set number
set ruler
set laststatus=2
set cmdheight=2
```

Example: Example (Neovim)

```
-- Neovim UI and mouse configuration
if vim.fn.has('mouse') == 1 then
  vim.opt.mouse = 'a'
end
vim.opt.number = true
vim.opt.ruler = true
vim.opt.laststatus = 2
vim.opt.cmdheight = 2
```

Source: vim.fandom.com

64.28 Non-Blinking Block Cursor in Linux Console

Category: ui
Tags: terminal, cursor, console

Disable cursor blinking in Linux console for a cleaner visual experience

Example: Example (Vim)

```
if &term == "linux"
  set t_ve+=^[[?81;0;112c
endif
```

Example: Example (Neovim)

```
if vim.o.term == "linux" then
  vim.o.t_ve = vim.o.t_ve .. "\x1b[?81;0;112c"
end
```

Source: vim.fandom.com

64.29 Prevent Mouse Drag from Entering Visual Mode

Category: ui
Tags: mouse, visual-mode, configuration

Disable automatic visual mode selection when mouse dragging in Vim, preventing unintended text selection

Example: Example (Vim)

```
noremap <LeftDrag> <LeftMouse>
noremap! <LeftDrag> <LeftMouse>
```

Example: Example (Neovim)

```
vim.keymap.set({'n', '!'}, '<LeftDrag>', '<LeftMouse>', { noremap = true })
```

Source: vim.fandom.com

64.30 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: Example (Vim)

```
:lua print(vim.treesitter.get_captures_at_cursor()[1] or "NONE")
```

Source: Community contributed

64.31 Quick Font Selector Keymapping

Category: ui
Tags: key-mapping, font, interactive

Provides a quick keyboard shortcut to open the font selection dialog in GUI Vim

Example: Example (Vim)

```
map <F3> <Esc>:set guifont=*<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F3>', ':set guifont=*<CR>', { desc = 'Open Font
↪  Selector' })
```

Source: vim.fandom.com

64.32 Quick Window Maximization Shortcut

Category: ui
Tags: keyboard-shortcut, fullscreen

Use F11 or custom KDE shortcut to toggle window maximization

Example: Example (Neovim)

```
-- Configure a custom mapping if needed
vim.keymap.set('n', '<F11>', ':set invfullscreen<CR>', { desc = 'Toggle
↪  Fullscreen' })
```

Source: vim.fandom.com

64.33 Quickly Adjust Font Size in Neovim GUI

Category: ui
Tags: font-size, gui, mapping

Provides a flexible, cross-platform method to dynamically adjust GUI font size using key mappings or mouse wheel

Example: Example (Vim)

```
function! AdjustFontSize(amount)
    if !has("gui_running")
        return
    endif

    let l:min_font_size = 5
    let l:max_font_size = 64

    let l:font_info = GetFontInfo()
    let l:font_name = l:font_info.name
    let l:font_size = l:font_info.size

    if a:amount == '-'
        let l:font_size = l:font_size - 1
    elseif a:amount == '+'
        let l:font_size = l:font_size + 1
    endif
```

```
    let l:font_size = max([l:min_font_size, min([l:max_font_size,
    ↪  l:font_size])])
    let &guifont = substitute(&guifont, l:font_size_pattern, l:font_size,
    ↪  '')
endfunction
```

Example: Example (Neovim)

```
function _G.adjust_font_size(amount)
    if not vim.g.gui_running then
        return
    end

    local min_font_size = 5
    local max_font_size = 64

    local font_info = get_font_info()
    local font_name = font_info.name
    local font_size = tonumber(font_info.size)

    if amount == '-' then
        font_size = font_size - 1
    elseif amount == '+' then
        font_size = font_size + 1
    end

    font_size = math.max(min_font_size, math.min(max_font_size, font_size))
    vim.o.guifont = vim.fn.substitute(vim.o.guifont, '\d+',
    ↪  tostring(font_size), '')
end

-- Example mappings
vim.keymap.set('n', '<F11>', function() _G.adjust_font_size('-') end)
vim.keymap.set('n', '<F12>', function() _G.adjust_font_size('+') end)
vim.keymap.set('n', '<C-ScrollWheelDown>', function()
↪  _G.adjust_font_size('-') end)
vim.keymap.set('n', '<C-ScrollWheelUp>', function() _G.adjust_font_size('+')
↪  end)
```

Source: vim.fandom.com

64.34 Remove Specific Toolbar Buttons

Category: ui
Tags: customization, toolbar, interface

Remove specific buttons from the GUI toolbar to simplify interface

Example: Example (Vim)

```
" Remove specific toolbar buttons
aunmenu ToolBar.FindNext
aunmenu ToolBar.FindPrev
```

Example: Example (Neovim)

```
-- Remove specific toolbar buttons
-- Note: This might require a plugin or custom implementation in Neovim
-- Consult your specific GUI configuration for exact method
```

Source: vim.fandom.com

64.35 Show Matching Braces Briefly

Category: ui
Tags: syntax-highlighting, editing-experience

Configure showmatch to briefly highlight matching braces when typing closing brackets

Example: Example (Vim)

```
set showmatch
set matchtime=3
```

Example: Example (Neovim)

```
vim.opt.showmatch = true
vim.opt.matchtime = 3
```

Source: vim.fandom.com

64.36 Show Non-Native File Format in Statusline

Category: ui
Tags: statusline, file-format, customization

Display a visual indicator when a file's format differs from the native system format, helping identify potential cross-platform issues

Example: Example (Vim)

```
function ShowFileFormatFlag(var)
  if ( a:var == 'dos' )
    return '[dos]'
  elseif ( a:var == 'mac' )
    return '[mac]'
```

```
  else
    return ''
  endif
endfunction

hi User1 term=bold cterm=bold ctermfg=red ctermbg=darkblue

set statusline+=%1*%{ShowFileFormatFlag(&fileformat)}%*
```

Example: Example (Neovim)

```
vim.cmd('hi User1 term=bold cterm=bold ctermfg=red ctermbg=darkblue')

local function show_file_format_flag()
  local ff = vim.bo.fileformat
  if ff == 'dos' then
    return '[dos]'
  elseif ff == 'mac' then
    return '[mac]'
  else
    return ''
  end
end

vim.opt.statusline:append('%1*%{v:lua.show_file_format_flag()}%*')
```

Source: vim.fandom.com

64.37 Statusline Color Indicator for Vim Modes

Category: ui
Tags: statusline, mode-indicator, autocmd

Dynamically change statusline color to visually indicate current Vim mode (normal/insert/replace)

Example: Example (Vim)

```
" Enable statusline always
set laststatus=2

" Change statusline color based on mode
function! InsertStatuslineColor(mode)
  if a:mode == 'i'
    hi statusline guibg=magenta
  elseif a:mode == 'r'
    hi statusline guibg=blue
  else
    hi statusline guibg=red
  endif
endfunction
```

```
au InsertEnter * call InsertStatuslineColor(v:insertmode)
au InsertChange * call InsertStatuslineColor(v:insertmode)
au InsertLeave * hi statusline guibg=green

" Default statusline to green on Vim entry
hi statusline guibg=green
```

Example: Example (Neovim)

```
-- Change statusline color based on mode
function InsertStatuslineColor(mode)
  if mode == 'i' then
    vim.cmd('hi statusline guibg=magenta')
  elseif mode == 'r' then
    vim.cmd('hi statusline guibg=blue')
  else
    vim.cmd('hi statusline guibg=red')
  end
end

-- Create autocmds for mode color changes
vim.api.nvim_create_augroup('ModeStatusline', { clear = true })
vim.api.nvim_create_autocmd('InsertEnter', {
  group = 'ModeStatusline',
  callback = function()
    InsertStatuslineColor(vim.v.insertmode)
  end
})
vim.api.nvim_create_autocmd('InsertChange', {
  group = 'ModeStatusline',
  callback = function()
    InsertStatuslineColor(vim.v.insertmode)
  end
})
vim.api.nvim_create_autocmd('InsertLeave', {
  group = 'ModeStatusline',
  callback = function()
    vim.cmd('hi statusline guibg=green')
  end
})

-- Default statusline to green on Vim entry
vim.cmd('hi statusline guibg=green')
```

Source: vim.fandom.com

64.38 Time-Based Color Scheme Selection

Category: ui
Tags: color-scheme, time-based, theme

Automatically change color scheme based on time of day, providing visual variety throughout the day

Example: Example (Vim)

```
function! s:HourColor()
  let hr = str2nr(strftime('%H'))
  if hr <= 3
    let i = 0
  elseif hr <= 7
    let i = 1
  elseif hr <= 14
    let i = 2
  elseif hr <= 18
    let i = 3
  else
    let i = 4
  endif
  let nowcolors = 'elflord morning desert evening pablo'
  execute 'colorscheme '.split(nowcolors)[i]
endfunction
```

Example: Example (Neovim)

```
local function hour_color()
  local hour = tonumber(os.date('%H'))
  local color_index = 0

  if hour <= 3 then color_index = 0
  elseif hour <= 7 then color_index = 1
  elseif hour <= 14 then color_index = 2
  elseif hour <= 18 then color_index = 3
  else color_index = 4 end

  local nowcolors = {'elflord', 'morning', 'desert', 'evening', 'pablo'}
  vim.cmd('colorscheme ' .. nowcolors[color_index + 1])
end

-- Usage: call hour_color() at startup or create a command
vim.api.nvim_create_user_command('TimeColorScheme', hour_color, {})
```

Source: vim.fandom.com

64.39 Toggle GUI Elements Dynamically

Category: ui
Tags: gui, interface, customization

Easily toggle visibility of menu bar, toolbar, and scrollbars in Neovim GUI

Example: Example (Vim)

```
" Toggle menu bar
nnoremap <C-F1> :if &go=~#'m'<Bar>set go-=m<Bar>else<Bar>set
↪  go+=m<Bar>endif<CR>

" Toggle toolbar
nnoremap <C-F2> :if &go=~#'T'<Bar>set go-=T<Bar>else<Bar>set
↪  go+=T<Bar>endif<CR>

" Toggle right scrollbar
nnoremap <C-F3> :if &go=~#'r'<Bar>set go-=r<Bar>else<Bar>set
↪  go+=r<Bar>endif<CR>
```

Example: Example (Neovim)

```
-- Toggle menu bar
vim.keymap.set('n', '<C-F1>', function()
  local current_opts = vim.o.guioptions
  vim.o.guioptions = current_opts:find('m') and current_opts:gsub('m', '')
  ↪  or current_opts .. 'm'
end, { desc = 'Toggle menu bar' })

-- Toggle toolbar
vim.keymap.set('n', '<C-F2>', function()
  local current_opts = vim.o.guioptions
  vim.o.guioptions = current_opts:find('T') and current_opts:gsub('T', '')
  ↪  or current_opts .. 'T'
end, { desc = 'Toggle toolbar' })

-- Toggle right scrollbar
vim.keymap.set('n', '<C-F3>', function()
  local current_opts = vim.o.guioptions
  vim.o.guioptions = current_opts:find('r') and current_opts:gsub('r', '')
  ↪  or current_opts .. 'r'
end, { desc = 'Toggle right scrollbar' })
```

Source: vim.fandom.com

64.40 Toggle Line Numbers Easily

Category: ui
Tags: toggle, display, line-numbers

Quickly toggle line numbers on and off with a simple command

Example: Example (Vim)

```
set nu!
:set nu!
```

Example: Example (Neovim)

```
vim.opt.number = not vim.opt.number:get()
```

Source: vim.fandom.com

64.41 Use Tearoff Menus with Keyboard Shortcuts

Category: ui
Tags: gui, menu-navigation, keyboard-shortcut

Activate tearoff menus using Alt+Menu Hotletter to open menus in a separate window, enabling faster keyboard-driven menu interaction

Example: Example (Vim)

```
" Enable tearoff menus
set guioptions+=t
```

Example: Example (Neovim)

```
-- Enable tearoff menus in Neovim
vim.opt.guioptions:append('t')
```

Source: vim.fandom.com

CHAPTER 65

Vimscript fundamentals

65.1 Autocommand creation in script

Category: Vim Script
Tags: autocmd, autocommand, event, group, script

Create autocommands programmatically with proper grouping and cleanup.

Example: Example (Vim)

```
" 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
```

65.2 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: Example (Vim)

```
: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
```

65.3 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: Example (Vim)

```
" 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
```

65.4 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: Example (Vim)

```
:let score = 85
:if score >= 90
  echo "Excellent"
:elseif score >= 70
  echo "Good"
:else
  echo "Needs improvement"
:endif

" Comparison operators: == != > < >= <= =~ !~
```

65.5 Debugging Vim scripts

Category: Vim Script
Tags: debug, echo, echom, verbose, profile

Debug Vim scripts using various techniques and built-in tools.

Example: Example (Vim)

```
" 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
```

65.6 Error handling with try-catch

Category: Vim Script
Tags: try, catch, finally, error, exception

Handle errors gracefully using try-catch-finally blocks.

Example: Example (Vim)

```
: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
```

65.7 Event handling and callbacks

Category: Vim Script
Tags: event, callback, timer, async, handler

Handle events and create callbacks using timers and autocommands.

Example: Example (Vim)

```
" 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()
```

65.8 File and buffer operations

Category: Vim Script
Tags: file, buffer, read, write, operation

Read/write files and manipulate buffers programmatically.

Example: Example (Vim)

```
" 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
```

65.9 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: Example (Vim)

```
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
```

65.10 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: Example (Vim)

```
" 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
```

65.11 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: Example (Vim)

```
" 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
```

65.12 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: Example (Vim)

```
" 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
```

65.13 Mappings in Vim script

Category: Vim Script
Tags: mapping, keymap, bind, shortcut, script

Create key mappings programmatically with different modes and options.

Example: Example (Vim)

```
" 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
```

65.14 Option manipulation

Category: Vim Script
Tags: option, set, setlocal, global, buffer

Get and set Vim options programmatically using & syntax.

Example: Example (Vim)

```
" 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
```

65.15 Regular expressions in Vim script

Category: Vim Script
Tags: regex, pattern, match, substitute, search

Use `=~` and `!~` operators for pattern matching, `substitute()` for replacements.

Example: Example (Vim)

```
: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
```

65.16 Script sourcing and modules

Category: Vim Script
Tags: source, module, include, script, organization

Source other scripts and create modular code organization.

Example: Example (Vim)

```
" 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
```

65.17 String formatting and printf

Category: Vim Script
Tags: string, format, printf, sprintf, output

Format strings using `printf()` and `string()` functions.

Example: Example (Vim)

```
: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
```

65.18 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: Example (Vim)

```
: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
```

65.19 System command execution

Category: Vim Script
Tags: system, command, execute, shell, external

Execute system commands and capture output using `system()` and related functions.

Example: Example (Vim)

```
" 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
```

65.20 User command definition

Category: Vim Script
Tags: command, user-command, define, custom

Create custom user commands with parameters and completion.

Example: Example (Vim)

```
" 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 66
Visual mode

66.1 Alternate Mouse Selection Method

Category: visual_mode
Tags: mouse-selection, key-mapping, visual-mode

Alternative method for blockwise selection using different modifiers

Example: Example (Vim)

```
noremap <S-RightMouse> <LeftMouse><Esc><C-V>
noremap <S-RightDrag> <LeftDrag>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<S-RightMouse>', '<LeftMouse><Esc><C-V>', { noremap =
↪  true })
vim.keymap.set('n', '<S-RightDrag>', '<LeftDrag>', { noremap = true })
```

Source: vim.fandom.com

66.2 Apply Substitutions to Visual Block Selections

Category: visual_mode
Tags: substitution, visual-block, editing

Apply substitutions specifically to a visual block selection without affecting entire lines

Example: Example (Vim)

```
:'<,'>B s/pattern/newtext/g
```

Example: Example (Neovim)

```
-- Using command-line mode for visual block substitution
-- Select block with <C-v>, then use:
-- :'<,'>B s/pattern/newtext/g
```

Source: vim.fandom.com

66.3 Block Replacement in Visual Mode

Category: visual_mode
Tags: block-edit, advanced-editing, replacement

Efficiently replace text in a visual block by changing the first line and applying to entire selection

Example: Example (Vim)

```
" In visual block mode (Ctrl-v)
c   " Change first line
<Esc><Esc>  " Apply changes to entire block
```

Example: Example (Neovim)

```
-- Block replacement can be done directly in Neovim
-- No specific Lua configuration needed
-- The default Vim behavior works in Neovim
```

Source: vim.fandom.com

66.4 Blockwise Edit Multiple Lines Simultaneously

Category: visual_mode
Tags: editing, multi-line, visual-block

Insert or modify text across multiple lines using visual block mode

Example: Example (Vim)

```
^<Ctrl-v>jjjjI#<ESC>  # Add comment at start of multiple lines
```

Example: Example (Neovim)

```
-- Similar approach in Lua
-- Use vim.cmd to execute visual block commands
vim.cmd('normal! ^<C-v>jjjjI#<Esc>')
```

Source: vim.fandom.com

66.5 Create Rectangular Block of Exact Size

Category: visual_mode
Tags: visual-block, precise-selection

Create a rectangular block selection of exactly the same size as a previous block, using current cursor as top-left corner

Example: Example (Vim)

```
" Use 1<Ctrl-V> instead of <Ctrl-V> move
```

Example: Example (Neovim)

```
-- Key sequence translation, use with caution in multibyte encodings
```

Source: vim.fandom.com

66.6 Drag Text Using Arrow Key Mappings

Category: visual_mode
Tags: key-mapping, text-movement, visual-editing

Enhanced mappings to move selected text left, right, up, and down using arrow keys in visual mode

Example: Example (Vim)

```
" Arrow key movements in visual mode
vnoremap <Left> h
vnoremap <Right> l
vnoremap <Up> k
vnoremap <Down> j

" Drag text with Ctrl-arrows
vnoremap <C-Right> lholhxp`[1v<Space>
vnoremap <C-Left> hlohlxhP`[1v<Space>
vnoremap <C-Down> jkojkxjzvP`[1v<Space>
vnoremap <C-Up> kjokjxkzvP`[1v<Space>
```

Example: Example (Neovim)

```
-- Arrow key movements in visual mode
vim.keymap.set('v', '<Left>', 'h', { noremap = true })
vim.keymap.set('v', '<Right>', 'l', { noremap = true })
vim.keymap.set('v', '<Up>', 'k', { noremap = true })
vim.keymap.set('v', '<Down>', 'j', { noremap = true })

-- Drag text with Ctrl-arrows
```

```
vim.keymap.set('v', '<C-Right>', 'lholhxp`[1v<Space>', { noremap = true })
vim.keymap.set('v', '<C-Left>', 'hlohlxhP`[1v<Space>', { noremap = true })
vim.keymap.set('v', '<C-Down>', 'jkojkxjzvP`[1v<Space>', { noremap = true })
vim.keymap.set('v', '<C-Up>', 'kjokjxkzvP`[1v<Space>', { noremap = true })
```

Source: vim.fandom.com

66.7 Easy Blockwise Mouse Selection in Vim

Category: visual_mode
Tags: mouse-selection, key-mapping, visual-mode

Simplify blockwise visual selection using Alt-Mouse or alternative modifier keys

Example: Example (Vim)

```
noremap <M-LeftMouse> <LeftMouse><Esc><C-V>
noremap <M-LeftDrag> <LeftDrag>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-LeftMouse>', '<LeftMouse><Esc><C-V>', { noremap =
↪  true })
vim.keymap.set('n', '<M-LeftDrag>', '<LeftDrag>', { noremap = true })
```

Source: vim.fandom.com

66.8 Flexible Line Selection with Visual Mode

Category: visual_mode
Tags: selection, editing, copying

Use visual mode to select and manipulate lines without precise line counting

Example: Example (Vim)

```
V       " Enter visual line mode
        " Select range of lines
ay      " Yank selected lines
```

Example: Example (Neovim)

```
-- Enter visual line mode
vim.cmd('normal V')
-- Yank selected lines
vim.cmd('normal ay')
```

Source: vim.fandom.com

66.9 Generate Incrementing Column Numbers

Category: visual_mode
Tags: column-editing, number-generation, visual-block

Easily create a column of incrementing numbers in a visual block selection

Example: Example (Vim)

```
# In visual block mode (Ctrl-v), select column
:I   # Increments numbers starting from first line
```

Example: Example (Neovim)

```
-- Note: Requires visincr.vim plugin
-- In visual block mode, use :I to increment numbers
-- Additional variants:
-- :I #    - Increment by specified number
-- :II     - Left-pad numbers
-- :II #   - Left-pad and increment by number
```

Source: vim.fandom.com

66.10 Handle Shift Key Mistakes in Visual Mode

Category: visual_mode
Tags: key-mapping, visual-mode, navigation

Prevent cursor jumping and unexpected behavior when accidentally holding Shift key in visual mode

Example: Example (Vim)

```
" Normalize Shift+arrow key behavior
nnoremap <S-Up> V
nnoremap <S-Down> V
xnoremap <S-Up> k
xnoremap <S-Down> j
```

Example: Example (Neovim)

```
-- Lua mappings to handle Shift key in visual mode
vim.keymap.set('n', '<S-Up>', 'V', { noremap = true })
vim.keymap.set('n', '<S-Down>', 'V', { noremap = true })
vim.keymap.set('x', '<S-Up>', 'k', { noremap = true })
```

```
vim.keymap.set('x', '<S-Down>', 'j', { noremap = true })
```

Source: vim.fandom.com

66.11 Highlight Text Inside Matching Parentheses

Category: visual_mode
Tags: text-selection, navigation, key-mapping

Quickly highlight text inside matching parentheses or other block delimiters with a custom key mapping

Example: Example (Vim)

```
" Highlight block and pause briefly
nmap <C-p> m[vab:sleep 350m<CR>`[
imap <C-p> <Esc>m[vab:sleep 350m<CR>`[a
```

Example: Example (Neovim)

```
-- Highlight block and pause briefly in Neovim
vim.keymap.set('n', '<C-p>', function()
  vim.cmd('normal! m[vab')
  vim.cmd('sleep 350m')
  vim.cmd('normal! `[')
end, { desc = 'Highlight matching block' })

vim.keymap.set('i', '<C-p>', function()
  vim.cmd('normal! <Esc>m[vab')
  vim.cmd('sleep 350m')
  vim.cmd('normal! `[a')
end, { desc = 'Highlight matching block in insert mode' })
```

Source: vim.fandom.com

66.12 Increment Numbers in Visual Block Mode

Category: visual_mode
Tags: number-increment, visual-block, editing-trick

Easily increment numbers in a visual block selection, creating sequential lists

Example: Example (Vim)

```
" Normal mode: g Ctrl-A to create sequential numbers
" Vim 8+ supports Ctrl-A to increment first number
```

Example: Example (Neovim)

```
-- In Neovim, use same Vim keybindings
-- Ctrl-V to enter visual block mode
-- Select numbers
-- g Ctrl-A or Ctrl-A to increment
```

Source: vim.fandom.com

66.13 Insert Text Across Multiple Lines in Visual Block Mode

Category: visual_mode
Tags: multi-line-editing, text-manipulation, visual-block

Quickly insert or append text at the same column across multiple lines using visual block mode

Example: Example (Vim)

```
" Select block with Ctrl-V, then:
" I - Insert at start of block
" A - Append at end of block
```

Example: Example (Neovim)

```
-- In Neovim, same key bindings work in visual block mode
-- Ctrl-V to enter visual block
-- I or A to insert/append text
```

Source: vim.fandom.com

66.14 Pad Lines for Easy Visual Block Editing

Category: visual_mode
Tags: text-manipulation, visual-block, editing-trick

Easily pad lines with trailing blanks to enable consistent visual block editing, especially when lines have different lengths

Example: Example (Vim)

```
" Pad all lines with trailing blanks to 'limit' length
function! AtOnce( limit )
  norm mm
  g/^/norm 100A
  g/^/call Truncate( getline('.'), limit )
  let @/=""
  norm 'm
endfunc
```

Example: Example (Neovim)

```
function PadLinesTrailingBlanks(limit)
  local current_pos = vim.fn.getcurpos()
  vim.cmd('g/^/normal! 100A')
  vim.cmd('g/^/call cursor(line("."), ' .. limit .. ') | normal! d$')
  vim.fn.setpos('.', current_pos)
end
```

Source: vim.fandom.com

66.15 Powerful Visual Selection and Paste Techniques

Category: visual_mode
Tags: visual-selection, copy-paste, editing

Comprehensive guide to using visual modes for efficient text selection, copying, and pasting with multiple options

Example: Example (Vim)

```
" Visual selection modes
v   " character-based visual selection
V   " whole line visual selection
Ctrl-v " block visual selection

" Paste operations
p   " paste after cursor
P   " paste before cursor
```

Example: Example (Neovim)

```
-- Lua equivalents are built-in with Neovim's default keymappings
-- No additional configuration needed

-- To enhance visual selection, you might add:
vim.keymap.set('n', 'gv', '`[v`]', { desc = 'Restore previous visual
↪  selection' })
```

Source: vim.fandom.com

66.16 Quick Brace Matching Visual Selection

Category: visual_mode
Tags: text-objects, selection

Efficient ways to select code blocks delimited by braces using different methods

Example: Example (Vim)

```
" Select inside/around braces
viB  " inner block
vaB  " block with braces
```

Example: Example (Neovim)

```
-- Lua uses same Vim text objects
-- Can be combined with other Neovim features like treesitter selection
```

Source: vim.fandom.com

66.17 Quick Visual Block Commenting

Category: visual_mode
Tags: commenting, block-editing, productivity

Quickly comment/uncomment code using visual block mode

Example: Example (Vim)

```
" Comment in visual block mode
<c-v>
... move around ...
0 (first column)
I (insert)
// (comment character)
<Esc>
```

Example: Example (Neovim)

```
-- In Lua, this is typically done via key mapping
vim.keymap.set('x', '<leader>c', function()
  local mode = vim.fn.mode()
  if mode == '\<C-V>' then
    vim.cmd('normal! I// ')
  end
end, { desc = 'Comment visual block' })
```

Source: vim.fandom.com

66.18 Quickly Reselect Last Visual Selection

Category: visual_mode
Tags: visual-mode, selection, productivity

Allows you to quickly reselect the last visual selection without having to manually recre-

ate the selection

Example: Example (Vim)

```
gv
```

Example: Example (Neovim)

```
vim.cmd('normal! gv')
```

Source: vim.fandom.com

66.19 Repeat Last Edit Across Visual Block Selection

Category: visual_mode
Tags: editing, visual-block, productivity

Allows repeating the last edit (.) across all lines in a visual block selection, making multi-line editing faster and more efficient

Example: Example (Vim)

```
" allow the . to execute once for each line of a visual selection
vnoremap . :normal .<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '.', ':normal .<CR>', { desc = 'Repeat last edit on
↪  visual block lines' })
```

Source: vim.fandom.com

66.20 Repeat Visual Block Changes Across Lines

Category: visual_mode
Tags: visual-mode, editing, productivity

Apply the same edit across multiple lines in a visual block selection

Example: Example (Vim)

```
vnoremap <silent> . :normal .<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', '.', function()
  vim.cmd('normal! .')
end, { desc = 'Repeat edit across visual block lines' })
```

Source: vim.fandom.com

66.21 Replace Visual Block with Another Block

Category: visual_mode
Tags: visual-block, text-replacement, advanced-editing

Efficiently replace one visual block of text with another by using yank and paste operations in visual block mode

Example: Example (Vim)

```
" Step 1: Select first block
ctrl-v move "ay

" Step 2: Select and replace second block
ctrl-v move c ctrl-o "aP <Esc>
```

Example: Example (Neovim)

```
-- Note: These are key sequences, so direct Lua translation isn't
↪  straightforward
-- But the concept involves visual block mode and register manipulation
-- Can be implemented with vim.fn.feedkeys() if needed
```

Source: vim.fandom.com

66.22 Reverse Selected Text in Visual Mode

Category: visual_mode
Tags: text-manipulation, visual-mode, mapping

Quickly reverse the characters in a visually selected text on a single line

Example: Example (Vim)

```
vnoremap ;rv c<C-O>:set revins<CR><C-R>"<Esc>:set norevins<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('v', ';rv', function()
  -- Save the current register content
  local original_reg = vim.fn.getreg('"')

  -- Reverse the selected text
  vim.cmd('normal! d')
  vim.o.revins = true
  vim.api.nvim_put({original_reg:reverse()}, '', true, true)
  vim.o.revins = false
end, { desc = 'Reverse selected text' })
```

Source: vim.fandom.com

66.23 Select Blocks with % Motion

Category: visual_mode
Tags: text-objects, selection, advanced-navigation

Quickly select entire code blocks using % motion in visual mode

Example: Example (Vim)

```
" Select block with v%
" Use viB for inner block
" Use vaB for block with braces
```

Example: Example (Neovim)

```
-- These motions work the same in Neovim
-- v% selects to matching brace
-- viB selects inner block
-- vaB selects block with braces
```

Source: vim.fandom.com

66.24 Select Different Block Types Easily

Category: visual_mode
Tags: text-selection, block-navigation

Quick mappings to select different types of text blocks using Vim's text objects

Example: Example (Vim)

```
" Mappings for selecting various block types
nmap <C-P>9 m[va(:sleep 350m<CR>`[
nmap <C-P>[ m[va[:sleep 350m<CR>`[
```

```
nmap <C-P>] m[va{:sleep 350m<CR>`[
nmap <C-P>, m[va<:sleep 350m<CR>`[
```

Example: Example (Neovim)

```
-- Mappings for selecting various block types in Neovim
local block_types = {
  ['9'] = '(',
  ['['] = '[',
  [']'] = '{',
  [','] = '<'
}

for key, block in pairs(block_types) do
  vim.keymap.set('n', '<C-P>' .. key, function()
    vim.cmd('normal! m[va' .. block)
    vim.cmd('sleep 350m')
    vim.cmd('normal! `[')
  end, { desc = 'Highlight ' .. block .. ' block' })
end
```

Source: vim.fandom.com

66.25 Select Entire Function with Visual Commands

Category: visual_mode
Tags: text-selection, navigation, code-editing

Quickly select an entire function, including comments and body, using visual mode and search commands

Example: Example (Vim)

```
V/{<CR>%  " Select function from comment to closing brace
```

Example: Example (Neovim)

```
vim.cmd('normal V/{<CR>%')  -- Select function from comment to closing brace
```

Source: vim.fandom.com

66.26 Select Recently Pasted Text Easily

Category: visual_mode
Tags: text-selection, mapping, paste

Create a quick mapping to select text just pasted, which is useful for further manipula-

tion like indenting or formatting

Example: Example (Vim)

```
nnoremap gp `[v`]
```

Example: Example (Neovim)

```
vim.keymap.set('n', 'gp', '`[v`]', { desc = 'Select last pasted text' })
```

Source: vim.fandom.com

66.27 Use Virtual Edit for Block Selection

Category: visual_mode
Tags: virtual-edit, block-selection, editing-mode

Enable virtual edit mode to allow block selection beyond line endings, making column editing more flexible

Example: Example (Vim)

```
set ve+=block  " Enable block virtual edit
```

Example: Example (Neovim)

```
vim.opt.virtualedit:append('block')  -- Enable block virtual edit
```

Source: vim.fandom.com

66.28 Visual Mode Block Indent Preservation

Category: visual_mode
Tags: indentation, visual-mode, reselect

Keeps visual block selection active after indenting, allowing multiple indents

Example: Example (Vim)

```
vmap > >gv
vmap < <gv
```

Example: Example (Neovim)

```
vim.keymap.set('v', '>', '>gv', { desc = 'Indent and reselect' })
vim.keymap.set('v', '<', '<gv', { desc = 'Unindent and reselect' })
```

Source: vim.fandom.com

66.29 Visual Mode Bracket Surrounding

Category: visual_mode
Tags: text-objects, editing, brackets

Quickly surround visually selected text with brackets or quotes

Example: Example (Vim)

```
" Surround visually selected text with parentheses
:vnoremap _( <Esc>`>a)<Esc>`<i(<Esc>
```

Example: Example (Neovim)

```
-- Surround visual selection with brackets
vim.keymap.set('v', '_(',
  function()
    vim.cmd('normal! `>a)')
    vim.cmd('normal! `<i(')
  end,
  { desc = 'Surround with parentheses' }
)
```

Source: vim.fandom.com

66.30 Visual Mode Text Swapping

Category: visual_mode
Tags: text-manipulation, visual-mode, exchange

Easily swap two pieces of text using visual mode and Ctrl-X

Example: Example (Vim)

```
" Swap visually selected text
:vnoremap <C-X> <Esc>`.``gvP``P
```

Example: Example (Neovim)

```
-- Swap visually selected text
vim.keymap.set('v', '<C-X>', function()
  local start_pos = vim.fn.getpos("'<")
  local end_pos = vim.fn.getpos("'>")
  local text1 = vim.fn.getline(start_pos[2], end_pos[2])
  vim.cmd('normal! gvp')
  vim.fn.setpos('.', start_pos)
```

```
    vim.fn.setline(start_pos[2], text1)
  end, { desc = 'Swap visually selected text' })
```

Source: vim.fandom.com

66.31 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: Example (Vim)

```
Ctrl+v  " select visual block
A       " append to end of all lines
text    " type text to append
Esc     " apply to all lines
```

Source: Community contributed

66.32 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: Example (Vim)

```
" In visual mode:
o   " move cursor to opposite corner of selection
O   " move to other corner (block mode only)
```

Source: Community contributed

66.33 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: Example (Vim)

```
" After making visual selection:
J    " join lines with spaces
gJ   " join lines without spaces
s    " substitute selected text
rx   " replace each character with 'x'
```

Source: Community contributed

66.34 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: Example (Vim)

```
" 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
```

Source: Community contributed

66.35 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: Example (Vim)

```
" Copy text first, then select other text:
p   " replace selection with register contents
P   " same as p in visual mode
```

Source: Community contributed

66.36 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: Example (Vim)

```
" After selecting text:
Ctrl+]  " jump to tag of selected text
K       " run help/man on selected keyword
```

Source: Community contributed

66.37 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: Example (Vim)

```
" 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
```

Source: Community contributed

66.38 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: Example (Vim)

```
y  " yank selected text
d  " delete selected text
```

Source: Community contributed

66.39 Yank highlighting

Category: Visual
Tags: yank, highlight, autocmd

Create an autocmd to highlight yanked text briefly for visual feedback.

Example: Example (Vim)

```
:lua vim.api.nvim_create_autocmd("TextYankPost", {callback = function()
↪  vim.highlight.on_yank() end})
```

Source: Community contributed

CHAPTER 67

Visual mode (advanced)

67.1 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: Example (Vim)

```
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)
```

67.2 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: Example (Vim)

```
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
```

67.3 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: Example (Vim)

```
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
```

67.4 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: Example (Vim)

```
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
```

67.5 Visual line operations

Category: Visual Mode Advanced
Tags: visual, line, operation, move, duplicate

Perform line-level operations on visual line selections.

Example: Example (Vim)

```
" 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
```

67.6 Visual mode column operations

Category: Visual Mode Advanced
Tags: visual, column, arithmetic, calculation, block

Perform arithmetic operations on columns of numbers.

Example: Example (Vim)

```
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...
```

67.7 Visual mode incremental selection

Category: Visual Mode Advanced
Tags: visual, incremental, selection, extend, treesitter

Use incremental selection for smart code selection (with treesitter).

Example: Example (Vim)

```
" 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
```

67.8 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: Example (Vim)

```
" After visual selection:
>               " indent selected text right
<               " indent selected text left
=               " auto-format selected text
gq              " format text to textwidth
```

67.9 Visual mode line manipulation

Category: Visual Mode Advanced
Tags: visual, line, manipulate, duplicate, move

Advanced line manipulation techniques in visual mode.

Example: Example (Vim)

```
" 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
```

67.10 Visual mode macro application

Category: Visual Mode Advanced
Tags: visual, macro, apply, lines, batch

Apply macros to each line of a visual selection.

Example: Example (Vim)

```
" 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
```

67.11 Visual mode pattern matching

Category: Visual Mode Advanced
Tags: visual, pattern, select, match, extend

Select text based on patterns and extend selections intelligently.

Example: Example (Vim)

```
/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
```

67.12 Visual mode rectangle operations

Category: Visual Mode Advanced
Tags: visual, rectangle, block, operation, column

Advanced rectangle/block operations for precise editing.

Example: Example (Vim)

```
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
```

67.13 Visual mode register operations

Category: Visual Mode Advanced
Tags: visual, register, yank, paste, specific

Work with specific registers in visual mode.

Example: Example (Vim)

```
" 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)
```

67.14 Visual mode search and replace

Category: Visual Mode Advanced
Tags: visual, search, replace, substitute, scope

Perform search and replace operations within visual selections.

Example: Example (Vim)

```
" 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
```

67.15 Visual mode smart selection

Category: Visual Mode Advanced
Tags: visual, smart, selection, expand, contract

Smart selection expansion and contraction techniques.

Example: Example (Vim)

```
" 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
```

67.16 Visual mode sorting and filtering

Category: Visual Mode Advanced
Tags: visual, sort, filter, lines, unique

Apply sorting and filtering operations to visual selections.

Example: Example (Vim)

```
" 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
```

67.17 Visual mode text transformation

Category: Visual Mode Advanced
Tags: visual, transform, text, case, format

Apply various text transformations to visual selections.

Example: Example (Vim)

```
" 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
```

67.18 Visual mode text wrapping

Category: Visual Mode Advanced
Tags: visual, wrap, text, format, width

Control text wrapping and formatting in visual selections.

Example: Example (Vim)

```
" After visual selection:
gw              " wrap lines to textwidth
gwap            " wrap around paragraph
gqq             " format current line
gqap            " format around paragraph
```

67.19 Visual mode with external filters

Category: Visual Mode Advanced
Tags: visual, filter, external, command, process

Filter visual selections through external commands.

Example: Example (Vim)

```
" 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
```

67.20 Visual mode with folds

Category: Visual Mode Advanced
Tags: visual, fold, unfold, selection, code

Work with code folds in visual mode.

Example: Example (Vim)

```
" After visual selection:
zf              " create fold from selection
:fold           " create fold from selected lines
zo              " open fold under cursor
zc              " close fold under cursor
```

67.21 Visual mode with global commands

Category: Visual Mode Advanced
Tags: visual, global, command, pattern, execute

Execute global commands on visual selections.

Example: Example (Vim)

```
" 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'
```

67.22 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: Example (Vim)

```
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
```

67.23 Visual mode with marks

Category: Visual Mode Advanced
Tags: visual, mark, position, range, selection

Use marks to create precise visual selections.

Example: Example (Vim)

```
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'
```

67.24 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: Example (Vim)

```
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
```

67.25 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: Example (Vim)

```
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 68

Window management

68.1 Alt Key Window Management Shortcuts

Category: window_management
Tags: key-mapping, window-navigation, productivity

Efficient window resizing and navigation using Alt key combinations without moving hands from home row

Example: Example (Vim)

```
map <silent> <A-h> <C-w><
map <silent> <A-j> <C-W>-
map <silent> <A-k> <C-W>+
map <silent> <A-l> <C-w>>

map <silent> <A-n> <C-w><C-w>
map <silent> <A-p> <C-w><S-w>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<A-h>', '<C-w><', { silent = true })
vim.keymap.set('n', '<A-j>', '<C-W>-', { silent = true })
vim.keymap.set('n', '<A-k>', '<C-W>+', { silent = true })
vim.keymap.set('n', '<A-l>', '<C-w>>', { silent = true })

vim.keymap.set('n', '<A-n>', '<C-w><C-w>', { silent = true })
vim.keymap.set('n', '<A-p>', '<C-w><S-w>', { silent = true })
```

Source: vim.fandom.com

68.2 Alternative Window Navigation Shortcuts

Category: window_management
Tags: key-mapping, window-navigation, productivity

Use Ctrl-J/K to move between horizontal splits and maximize them quickly

Example: Example (Vim)

```
map <C-J> <C-W>j<C-W>_
map <C-K> <C-W>k<C-W>_
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-J>', '<C-W>j<C-W>_', { desc = 'Move down and
↪  maximize window' })
vim.keymap.set('n', '<C-K>', '<C-W>k<C-W>_', { desc = 'Move up and maximize
↪  window' })
```

Source: vim.fandom.com

68.3 Auto-Resize Window for Diff Mode

Category: window_management
Tags: diff, window-resize, gui-settings

Automatically expand window width when in diff mode for better side-by-side comparison

Example: Example (Vim)

```
" Window settings for diff mode
if &diff
  let &columns = ((&columns*2 > 172)? 172: &columns*2)
endif
```

Example: Example (Neovim)

```
-- Lua equivalent for diff mode window resizing
if vim.o.diff then
  local columns = vim.o.columns
  vim.o.columns = math.min(columns * 2, 172)
end
```

Source: vim.fandom.com

68.4 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: Example (Vim)

```
" Vimscript:
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>
```

Example: Example (Neovim)

```
-- Lua:
local function better_gm()
  vim.cmd('normal! ^')
  local first_col = vim.fn.virtcol('.')
  vim.cmd('normal! g_')
  local last_col = vim.fn.virtcol('.')
  vim.cmd('normal! ' .. math.floor((first_col + last_col) / 2) .. '|')
end

vim.keymap.set('n', 'gm', better_gm, { silent = true, desc = 'Go to middle
↪  of line' })
vim.keymap.set('o', 'gm', better_gm, { silent = true, desc = 'Go to middle
↪  of line' })
```

Source: Community contributed

68.5 Buffer Split Management

Category: window_management
Tags: splits, buffers, window-navigation

Create and navigate buffer splits with flexible positioning

Example: Example (Vim)

```
:split
:vsplit
:vertical sb 3
Ctrl-W hjkl
```

Example: Example (Neovim)

```
-- Create splits
vim.cmd('split')  -- Horizontal split
vim.cmd('vsplit')  -- Vertical split
```

```
-- Navigate splits
vim.keymap.set('n', '<C-w>h', '<C-w>h', { desc = 'Move to left split' })
vim.keymap.set('n', '<C-w>j', '<C-w>j', { desc = 'Move to bottom split' })
vim.keymap.set('n', '<C-w>k', '<C-w>k', { desc = 'Move to top split' })
vim.keymap.set('n', '<C-w>l', '<C-w>l', { desc = 'Move to right split' })
```

Source: vim.fandom.com

68.6 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: Example (Vim)

```
" Vimscript:
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
```

Example: Example (Neovim)

```
-- Lua:
vim.opt.guicursor = 'n-v-c:block,i-ci-ve:ver25,r-cr:hor20,o:hor50'
-- Block in normal, vertical bar in insert, horizontal in replace
```

Source: Community contributed

68.7 Close Tab and Focus Left Tab Automatically

Category: window_management
Tags: tab-navigation, key-mapping, workflow

Provides a custom mapping to close a tab and automatically focus on the left tab, improving tab navigation efficiency

Example: Example (Vim)

```
noremap <silent><C-S-w> :if tabpagenr() !=
↪  tabpagenr('$')<CR>:tabclose<CR>:if tabpagenr() >
↪  1<CR>:tabprev<CR>:endif<CR>:else<CR>:tabclose<CR>:endif<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-S-w>', function()
  local current_tab = vim.fn.tabpagenr()
  local total_tabs = vim.fn.tabpagenr('$')

  if current_tab ~= total_tabs then
    vim.cmd('tabclose')
    if vim.fn.tabpagenr() > 1 then
      vim.cmd('tabprev')
    end
  else
    vim.cmd('tabclose')
  end
end, { silent = true })
```

Source: vim.fandom.com

68.8 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: Example (Vim)

```
:only   " close all other windows (keep current)
:on     " short form of :only
Ctrl+w o " normal mode shortcut for :only
```

Source: Community contributed

68.9 Control New Window Split Position

Category: window_management
Tags: splits, window-layout, configuration

Customize default window split behavior by controlling where new windows appear when splitting horizontally or vertically

Example: Example (Vim)

```
set splitbelow
set splitright
```

Example: Example (Neovim)

```
vim.opt.splitbelow = true
vim.opt.splitright = true
```

Source: vim.fandom.com

68.10 Customize GVim Popup Menu for Window Management

Category: window_management
Tags: mouse, popup-menu, window-control

Add custom menu options to close windows directly from the right-click popup menu in GVim

Example: Example (Vim)

```
:set mousemodel=popup
:amenu PopUp.Close.Window :confirm close<CR>
:amenu PopUp.Close.Other :confirm only<CR>
```

Example: Example (Neovim)

```
-- Set mouse model for popup menu
vim.opt.mousemodel = 'popup'

-- Add menu items for closing windows
vim.cmd('amenu PopUp.Close.Window :confirm close<CR>')
vim.cmd('amenu PopUp.Close.Other :confirm only<CR>')
```

Source: vim.fandom.com

68.11 Display Python Docs in Preview Window

Category: window_management
Tags: python, documentation, preview-window

Create a custom command to show Python documentation in a preview window, making it easy to view help without leaving the current buffer

Example: Example (Vim)

```
command! -nargs=1 -bar Pyhelp :call ShowPydoc(<f-args>)
function! ShowPydoc(what)
  let path = $TEMP . '/' . a:what . '.pydoc'
  call system(s:pydoc_path . " " . a:what . (&shellredir))
  execute "pedit" fnameescape(path)
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_user_command('Pyhelp', function(opts)
  local what = opts.args
  local path = vim.fn.tempname() .. '/' .. what .. '.pydoc'
  vim.fn.system('pydoc ' .. what .. ' > ' .. path)
  vim.cmd('pedit ' .. path)
end, { nargs = 1 })
```

Source: vim.fandom.com

68.12 Dynamic Tag Preview with Full-Height Vertical Window

Category: window_management
Tags: tags, preview, navigation

Flexible function to open tag previews in vertical or horizontal windows, with custom height and positioning

Example: Example (Vim)

```
function PreviewTag(top)
  set previewheight=25
  exe "silent! pclose"
  if &previewwindow
    return
  endif
  let w = expand("<cword>")
  exe "ptjump " . w
  if a:top
    return
  endif
  exe "silent! wincmd P"
  if &previewwindow
    if has("folding")
      silent! .foldopen
    endif
    wincmd L
    wincmd p
    if !&previewwindow
      wincmd _
    endif
  endif
endfunction
```

Example: Example (Neovim)

```
function _G.preview_tag(top)
  vim.o.previewheight = 25
  vim.cmd('silent! pclose')

  if vim.wo.previewwindow then
```

```
    return
  end

  local word = vim.fn.expand('<cword>')
  vim.cmd('ptjump ' .. word)

  if top then
    return
  end

  vim.cmd('silent! wincmd P')
  if vim.wo.previewwindow then
    if vim.wo.foldenable then
      vim.cmd('silent! .foldopen')
    end
    vim.cmd('wincmd L')
    vim.cmd('wincmd p')

    if not vim.wo.previewwindow then
      vim.cmd('wincmd _')
    end
  end
end

-- Key mappings
vim.keymap.set('n', '<C-]>', function() _G.preview_tag(false) end)
vim.keymap.set('n', '<M-]>', function() _G.preview_tag(true) end)
vim.keymap.set('n', '<M-[>', ':pc<CR>')
```

Source: vim.fandom.com

68.13 Dynamic Window Resizing Keymaps

Category: window_management
Tags: key-mapping, window-resize, navigation

Quickly resize vertical windows by moving focus and expanding width with custom keymaps

Example: Example (Vim)

```
" Resize vertical windows with Ctrl-h and Ctrl-l
nmap <c-h> <c-w>h<c-w>_<c-w>><c-w>>
nmap <c-l> <c-w>l<c-w>_<c-w>><c-w>>
```

Example: Example (Neovim)

```
-- Resize vertical windows dynamically
vim.keymap.set('n', '<C-h>', '<C-w>h<C-w>_<C-w>><C-w>>', { desc = 'Resize
↪  window left' })
```

```
vim.keymap.set('n', '<C-l>', '<C-w>l<C-w>_<C-w>><C-w>>', { desc = 'Resize
↪  window right' })
```

Source: vim.fandom.com

68.14 Dynamic Window Resizing for Line Numbers

Category: window_management
Tags: window-resize, line-numbers, toggle

Automatically adjust window columns when toggling line numbers to maintain layout

Example: Example (Vim)

```
function! Toggle_num()
  if !exists("g:grow")
    let g:grow = &numberwidth
  endif
  set number!
  if &number
    exec "set columns+=" . g:grow
  else
    exec "set columns-=" . g:grow
  endif
endfunction
map <M-n> :call Toggle_num()<CR>
```

Example: Example (Neovim)

```
local function toggle_num()
  local grow = vim.o.numberwidth
  vim.wo.number = not vim.wo.number
  if vim.wo.number then
    vim.o.columns = vim.o.columns + grow
  else
    vim.o.columns = vim.o.columns - grow
  end
end

vim.keymap.set('n', '<M-n>', toggle_num, { desc = 'Toggle line numbers and
↪  adjust window' })
```

Source: vim.fandom.com

68.15 Dynamic Window Resizing with F1 Key

Category: window_management
Tags: key-mapping, window-resize, gui

Cycle through predefined window sizes and toggle line numbers using F1 key, useful for adjusting workspace layout

Example: Example (Vim)

```
nmap <F1> :call ResizeWindow()<CR>
imap <F1> <Esc><F1>a

function! ResizeWindow()
  if (has("gui_running"))
    if s:selectedsize == 1
      let s:selectedsize = 2
      set number
      set columns=88
      set lines=35
    elseif s:selectedsize == 2
      set number
      let s:selectedsize = 3
      set columns=98
      set lines=45
    else
      let s:selectedsize = 1
      set nonumber
      set columns=80
      set lines=25
    endif
  endif
endfunction

let s:selectedsize=1
call ResizeWindow()
```

Example: Example (Neovim)

```
local selectedSize = 1

local function resizeWindow()
  if vim.g.neovide or vim.fn.has('gui_running') == 1 then
    if selectedSize == 1 then
      selectedSize = 2
      vim.o.number = true
      vim.o.columns = 88
      vim.o.lines = 35
    elseif selectedSize == 2 then
      selectedSize = 3
      vim.o.number = true
      vim.o.columns = 98
      vim.o.lines = 45
    else
      selectedSize = 1
      vim.o.number = false
      vim.o.columns = 80
      vim.o.lines = 25
    end
```

```
  end
end

vim.keymap.set({'n', 'i'}, '<F1>', resizeWindow)
```

Source: vim.fandom.com

68.16 Dynamically Resize Quickfix Window Height

Category: window_management
Tags: quickfix, window-resize, autocmd

Automatically adjust quickfix window height to fit contents, with min and max limits

Example: Example (Vim)

```
au FileType qf call AdjustWindowHeight(3, 10)
function! AdjustWindowHeight(minheight, maxheight)
  exe max([min([line("$"), a:maxheight]), a:minheight]) . "wincmd _"
endfunction
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'qf',
  callback = function()
    local last_line = vim.fn.line('$')
    local height = math.max(math.min(last_line, 10), 3)
    vim.cmd(height .. 'wincmd _')
  end
})
```

Source: vim.fandom.com

68.17 Easily Compare Two Files in Vim/Neovim

Category: window_management
Tags: diff-mode, file-comparison, window-split

Quick methods to compare two files side by side using built-in diff commands

Example: Example (Vim)

```
" Method 1: Split and diff two files
:sp <filename>
:windo diffthis

" Method 2: Vertical split with diff
```

```
:vert diffsplit <filename>
```

Example: Example (Neovim)

```
-- Method 1: Split and diff two files
vim.cmd('sp ' .. filename)
vim.cmd('windo diffthis')

-- Method 2: Vertical split with diff
vim.cmd('vert diffsplit ' .. filename)
```

Source: vim.fandom.com

68.18 Easily Turn Off Diff Mode

Category: window_management
Tags: diff-mode, window-management, buffer-reset

Quickly disable diff mode across all windows without closing buffers

Example: Example (Vim)

```
" Switch off diff mode for all windows
command! DiffOff call DiffOff()
function! DiffOff()
  windo set nodiff
  windo set noscrollbind
  windo set foldmethod=manual
  windo set foldcolumn=0
  windo unlet! b:did_ftplugin | let &filetype = &filetype
endfunction
```

Example: Example (Neovim)

```
-- Switch off diff mode for all windows
vim.api.nvim_create_user_command('DiffOff', function()
  for _, win in ipairs(vim.api.nvim_list_wins()) do
    vim.api.nvim_win_call(win, function()
      vim.wo.diff = false
      vim.wo.scrollbind = false
      vim.wo.foldmethod = 'manual'
      vim.wo.foldcolumn = '0'
    end)
  end
end, {})
```

Source: vim.fandom.com

68.19 Enable Mouse Focus Across Splits

Category: window_management
Tags: mouse, split-windows, interaction

Enable mouse focus to easily interact with multiple window splits

Example: Example (Vim)

```
set mousefocus
```

Example: Example (Neovim)

```
vim.opt.mousefocus = true
```

Source: vim.fandom.com

68.20 Execute SQL Queries in Split Window

Category: window_management
Tags: sql, external-commands, productivity

Execute SQL queries from Vim and display results in a split window, allowing seamless database interaction without leaving the editor

Example: Example (Vim)

```
au FileType sql map <F12> <C-W><C-O>:silent w !isql -SYourServerName
↪  -DYourDatabaseName -UYourUserName -PYourPassword >
↪  tmpsqlresult.txt<CR>:split tmpsqlresult.txt<CR>
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('FileType', {
  pattern = 'sql',
  callback = function()
    vim.keymap.set('n', '<F12>', function()
      vim.cmd('silent w !isql -SYourServerName -DYourDatabaseName
      ↪  -UYourUserName -PYourPassword > tmpsqlresult.txt')
      vim.cmd('split tmpsqlresult.txt')
    end, { buffer = true })
  end
})
```

Source: vim.fandom.com

68.21 Focus mode for writing

Category: Window Management
Tags: focus, writing, distraction, zen

Create a distraction-free environment for writing and focused editing.

Example: Example (Vim)

```
" 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>
```

Source: Community contributed

68.22 Go to Nth-from-Last Window

Category: window_management
Tags: window-navigation, window-commands

Efficiently navigate to the nth window from the last window using winnr() function

Example: Example (Vim)

```
:map <F12> :exe winnr('$') - 1 'wincmd w'<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<F12>', function()
  vim.cmd.wincmd(tostring(vim.fn.winnr('$') - 1))
end)
```

Source: vim.fandom.com

68.23 Keep Quickfix Window at Fixed Height

Category: window_management
Tags: autocmd, quickfix, window-resize

Automatically maintain a consistent quickfix window height across window switches, useful for debugging and error navigation

Example: Example (Vim)

```
function MaximizeAndResizeQuickfix(quickfixHeight)
  set lazyredraw
  set ei=WinEnter
  wincmd _
  if (getbufvar(winbufnr(winnr()), "&buftype") == "quickfix")
    wincmd p
    resize
    wincmd p
    exe "resize " . a:quickfixHeight
  endif
  set ei-=WinEnter
  set nolazyredraw
endfunction

au WinEnter * call MaximizeAndResizeQuickfix(8)
```

Example: Example (Neovim)

```
local function maximize_and_resize_quickfix(quickfix_height)
  vim.o.lazyredraw = true
  vim.o.eventignore = 'WinEnter'
  vim.cmd('wincmd _')

  if vim.bo.buftype == 'quickfix' then
    vim.cmd('wincmd p')
    vim.cmd('resize')
    vim.cmd('wincmd p')
    vim.cmd('resize ' .. quickfix_height)
  end

  vim.o.eventignore = ''
  vim.o.lazyredraw = false
end

vim.api.nvim_create_autocmd('WinEnter', {
  callback = function() maximize_and_resize_quickfix(8) end
})
```

Source: vim.fandom.com

68.24 Keep cursor centered

Category: Window Management
Tags: cursor, center, scroll, display

Keep cursor centered vertically on screen for better visibility while editing.

Example: Example (Vim)

```
" Vimscript:
" 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
```

Example: Example (Neovim)

```
-- Lua:
-- Keep cursor centered when scrolling
vim.keymap.set('n', '<C-d>', '<C-d>zz', { desc = 'Scroll down and center' })
vim.keymap.set('n', '<C-u>', '<C-u>zz', { desc = 'Scroll up and center' })
vim.keymap.set('n', 'n', 'nzz', { desc = 'Next search result and center' })
vim.keymap.set('n', 'N', 'Nzz', { desc = 'Previous search result and center'
↪  })

-- Or automatic centering
vim.opt.scrolloff = 999
```

Source: Community contributed

68.25 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: Example (Vim)

```
:bp|bd #        " go to previous buffer, delete current
:enew|bd #      " create new buffer, delete previous
```

Source: Community contributed

68.26 Maximize Current Window Quickly

Category: window_management
Tags: window, mapping, productivity

Provides a quick way to maximize the current window without closing other windows or with full screen

Example: Example (Vim)

```
" Maximize current window keeping other windows open
map <F5> <C-W>_<C-W><Bar>

" Close other windows and maximize current
:only
" or
CTRL-W o
```

Example: Example (Neovim)

```
-- Maximize current window while keeping others
vim.keymap.set('n', '<F5>', '<C-W>_<C-W>|', { desc = 'Maximize current
↪  window' })

-- Close other windows and maximize current
vim.keymap.set('n', '<leader>wo', ':only<CR>', { desc = 'Close other
↪  windows' })
```

Source: vim.fandom.com

68.27 Maximize Vim Window Size in KDE

Category: window_management
Tags: window-size, kde, gui

Set exact window lines and columns to achieve maximization in KDE using .gvimrc

Example: Example (Vim)

```
set lines=999 columns=999
```

Example: Example (Neovim)

```
vim.opt.lines = 999
vim.opt.columns = 999
```

Source: vim.fandom.com

68.28 Maximize Vim Window on Windows

Category: window_management
Tags: windows, startup, configuration

Automatically maximize Vim window when launching on Windows systems

Example: Example (Vim)

```
au GUIEnter * simalt ~x
```

Example: Example (Neovim)

```
vim.api.nvim_create_autocmd('UIEnter', {
  callback = function()
    if vim.fn.has('win32') == 1 then
      vim.cmd('simalt ~x')
    end
  end
})
```

Source: vim.fandom.com

68.29 Minimize and Restore Vim Windows Quickly

Category: window_management
Tags: window-control, system-integration, productivity

Use Ctrl-Z to suspend/minimize Vim in the background, with an optional system-wide shortcut for quick restoration

Example: Example (Vim)

```
" Built-in Vim suspend functionality
" Ctrl-Z suspends Vim in normal/visual mode
```

Example: Example (Neovim)

```
-- Neovim uses the same built-in suspend functionality
-- Use Ctrl-Z to suspend the current Vim instance
-- Restore using fg in terminal or system shortcut
```

Source: vim.fandom.com

68.30 Move Window Between Tabs Dynamically

Category: window_management
Tags: tabs, window-navigation, advanced-movement

Functions to move the current window between tabs, allowing dynamic rearrangement of buffers across tab pages

Example: Example (Vim)

```
function MoveToNextTab()
  "there is only one window
  if tabpagenr('$') == 1 && winnr('$') == 1
    return
  endif
  "preparing new window
  let l:tab_nr = tabpagenr('$')
  let l:cur_buf = bufnr('%')
  if tabpagenr() < tab_nr
    close!
    if l:tab_nr == tabpagenr('$')
      tabnext
    endif
    sp
  else
    close!
    tabnew
  endif
  "opening current buffer in new window
  exe "b".l:cur_buf
endfunc
```

Example: Example (Neovim)

```
function _G.move_to_next_tab()
  local total_tabs = vim.fn.tabpagenr('$')
  local current_tab = vim.fn.tabpagenr()
  local current_buf = vim.fn.bufnr('%')

  if total_tabs == 1 and vim.fn.winnr('$') == 1 then
    return
  end

  if current_tab < total_tabs then
    vim.cmd('close!')
    vim.cmd('tabnext')
    vim.cmd('split')
  else
    vim.cmd('close!')
    vim.cmd('tabnew')
  end

  vim.cmd('buffer ' .. current_buf)
```

```
end

-- Add a keymap
vim.keymap.set('n', '<A-.>', '<cmd>lua move_to_next_tab()<CR>', { desc =
↪  'Move window to next tab' })
```

Source: vim.fandom.com

68.31 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: Example (Vim)

```
Ctrl+w T  " move current window to new tab
```

Source: Community contributed

68.32 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: Example (Vim)

```
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
```

Source: Community contributed

68.33 Multiple Views of Same Buffer in Different Tabs

Category: window_management
Tags: tabs, buffer-management, workflow

Create alternate views of the same file in different tabs, such as a folded summary view and a detailed view with additional settings

Example: Example (Vim)

```
" Open a new tab with folded view
:tab split
:setlocal foldenable
```

Example: Example (Neovim)

```
-- Open a new tab with folded view
vim.cmd('tab split')
vim.wo.foldenable = true
```

Source: vim.fandom.com

68.34 Open GVim Matching Console Window Position

Category: window_management
Tags: window-positioning, shell-integration, x11

Dynamically position GVim window to match the current console window's geometry using X11 window information

Example: Example (Vim)

```
function gv
  xwi=$(xwininfo -id $WINDOWID)
  # Extract window geometry details
  # Position and resize gvim to match console window
  gvim -geometry "${w}x${h}+${x}+${y}" "$@"
endfunction
```

Example: Example (Neovim)

```
function _G.open_gvim_matching_console()
  local windowid = vim.env.WINDOWID
  if not windowid then return end

  -- Use vim.fn.system() to get xwininfo details
  -- Implement geometry extraction and gvim positioning
  vim.cmd('!gvim -geometry "'..geometry..'"')
end
```

Source: vim.fandom.com

68.35 Open Multiple Files in New Windows

Category: window_management
Tags: file-operations, custom-commands, window-splitting

Create custom commands to open multiple files in new windows, supporting glob patterns and flexible file selection

Example: Example (Vim)

```
command! -nargs=* -complete=file New call Foreach( "new %% ", <f-args> )
command! -nargs=* -complete=file Vnew call Foreach( "vnew %% ", <f-args> )

function! Foreach( ... )
  let cmd = a:1
  let fnames = []
  for i in range( 1, a:0 - 1 )
    let l = split( glob( a:000[i] ), "\n" )
    if len( l ) < 1
      let l = [ a:000[i] ]
    endif
    call extend( fnames, l )
  endfor
  for fname in fnames
    let cmd1 = substitute( cmd, '%%', fname, 'g' )
    exe cmd1
  endfor
endfunction
```

Example: Example (Neovim)

```
local function foreach(cmd, ...)
  local fnames = {}
  for _, pattern in ipairs({...}) do
    local files = vim.fn.glob(pattern, false, true)
    if #files == 0 then
      table.insert(files, pattern)
    end
    vim.list_extend(fnames, files)
  end

  for _, fname in ipairs(fnames) do
    local exec_cmd = cmd:gsub('%%', fname)
    vim.cmd(exec_cmd)
  end
end

vim.api.nvim_create_user_command('New', function(opts)
  foreach('new ' .. opts.args, unpack(opts.fargs))
end, { nargs = '*', complete = 'file' })

vim.api.nvim_create_user_command('Vnew', function(opts)
  foreach('vnew ' .. opts.args, unpack(opts.fargs))
end, { nargs = '*', complete = 'file' })
```

Source: vim.fandom.com

68.36 Open Neovim Matching Console Window Geometry

Category: window_management
Tags: window-geometry, terminal-integration, x11

Dynamically position Neovim window to match the current terminal's size and position using X11 window information

Example: Example (Vim)

```
# Korn shell function to get current window geometry
function gv
{
  xwi=$(xwininfo -id $WINDOWID)
  # Extract window position and size
  x=${xyposn%+[0-9]*}
  y=${xyposn##*+}
  w=${wh%x*}
  h=${wh#*x}

  # Adjust for borders and menus
  h=h-4
  x=x-4
  y=y-18

  # Open gvim with matching geometry
  gvim -geometry "${w}x${h}+${x}+${y}" "$@"
}
```

Example: Example (Neovim)

```
-- Lua function to get window geometry and open Neovim
local function open_matching_window()
  -- Use vim.fn.system to run xwininfo and parse output
  local xwi = vim.fn.system('xwininfo -id ' .. os.getenv('WINDOWID'))

  -- Extract window dimensions (simplified example)
  local x, y, w, h = xwi:match('Corners: (%d+)%+(%d+).*geometry
  ↪  (%d+)x(%d+)')

  -- Adjust geometry
  h = tonumber(h) - 4
  x = tonumber(x) - 4
  y = tonumber(y) - 18

  -- Open Neovim with matching geometry
  vim.cmd('!nvim -c "set lines=' .. h .. ' columns=' .. w .. '"')
end
```

Source: vim.fandom.com

68.37 Open New Buffers Below Current Window

Category: window_management
Tags: splits, buffer-management, configuration

Configure Vim/Neovim to open new split windows below the current window by default

Example: Example (Vim)

```
set splitbelow
```

Example: Example (Neovim)

```
vim.opt.splitbelow = true
```

Source: vim.fandom.com

68.38 Open New Vertical Buffers to the Right

Category: window_management
Tags: splits, buffer-management, configuration

Configure Vim/Neovim to open new vertical split windows to the right of the current window

Example: Example (Vim)

```
set splitright
```

Example: Example (Neovim)

```
vim.opt.splitright = true
```

Source: vim.fandom.com

68.39 Open Source Navigator Files in Vim Tabs

Category: window_management
Tags: tab-management, external-tool, integration

Use –remote-tab-silent to open files in tabs of the same Vim window

Example: Example (Vim)

```
gvim --remote-tab-silent +"call cursor(%l,%c)" %f
```

Example: Example (Neovim)

```
vim.fn.system('gvim --remote-tab-silent +"call cursor(' .. line .. ',' ..
↪  col .. ')" ' .. filename)
```

Source: vim.fandom.com

68.40 Open Tag in Split Window

Category: window_management
Tags: navigation, tags, window-split

Quickly open a tag definition in a new split window without losing the current context

Example: Example (Vim)

```
:map <C-\> :sp<CR><C-]><C-w>_
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-\>', function()
  vim.cmd('sp')
  vim.cmd('tag')
  vim.cmd('wincmd _')
end, { desc = 'Open tag in new split window' })
```

Source: vim.fandom.com

68.41 Prevent Accidental Window Closure

Category: window_management
Tags: key-mapping, window-navigation, safety

Disable the default <C-W>o command that closes all other windows to prevent accidentally destroying your window layout

Example: Example (Vim)

```
nnoremap <C-W>O :echo "sucker"<CR>
nnoremap <C-W>o :echo "sucker"<CR>
nnoremap <C-W><C-O> :echo "sucker"<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<C-W>o', function() vim.notify('Window closure
↪  prevented', vim.log.levels.WARN) end)
```

```
vim.keymap.set('n', '<C-W>O', function() vim.notify('Window closure
 ↪  prevented', vim.log.levels.WARN) end)
vim.keymap.set('n', '<C-W><C-O>', function() vim.notify('Window closure
 ↪  prevented', vim.log.levels.WARN) end)
```

Source: vim.fandom.com

68.42 Prevent Cursor Movement When Switching Windows

Category: window_management
Tags: mouse-handling, focus, window-interaction

Temporarily disable mouse interactions when switching windows to prevent cursor repositioning

Example: Example (Vim)

```
au!
  au FocusLost * let g:oldmouse=&mouse | set mouse=
  au FocusGained * if exists('g:oldmouse') | let &mouse=g:oldmouse | unlet
   ↪  g:oldmouse | endif
augroup END
```

Example: Example (Neovim)

```
local function setup_mouse_focus()
  local augroup = vim.api.nvim_create_augroup('MouseFocusPreserve', { clear
   ↪  = true })

  vim.api.nvim_create_autocmd('FocusLost', {
    group = augroup,
    callback = function()
      vim.g.oldmouse = vim.o.mouse
      vim.o.mouse = ''
    end
  })

  vim.api.nvim_create_autocmd('FocusGained', {
    group = augroup,
    callback = function()
      if vim.g.oldmouse then
        vim.o.mouse = vim.g.oldmouse
        vim.g.oldmouse = nil
      end
    end
  })
end

setup_mouse_focus()
```

Source: vim.fandom.com

68.43 Quick Tab Management for Editing Multiple Files

Category: window_management
Tags: tabs, file-navigation, buffer-management

Simple mappings to open current buffer in a new tab and quickly close it, improving multi-file editing workflow

Example: Example (Vim)

```
nmap t% :tabedit %<CR>
nmap td :tabclose<CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', 't%', ':tabedit %<CR>', { desc = 'Open current buffer in
↪  new tab' })
vim.keymap.set('n', 'td', ':tabclose<CR>', { desc = 'Close current tab' })
```

Source: vim.fandom.com

68.44 Quick Window Navigation with Spacebar

Category: window_management
Tags: key-mapping, window-navigation, productivity

Use spacebar to quickly switch between and maximize split windows, improving navigation efficiency

Example: Example (Vim)

```
"Jump between windows
map <Space> <c-W>w
"Open window wide
map <Space><Space> :call OpenSplittedWindowWide()<CR>

function OpenSplittedWindowWide()
  normal ^W|
  normal ^W20+
endfunction
```

Example: Example (Neovim)

```
-- Jump between windows
vim.keymap.set('n', '<Space>', '<C-W>w', { desc = 'Switch to next window' })

-- Open window wide
vim.keymap.set('n', '<Space><Space>', function()
  vim.cmd('wincmd |')  -- Maximize width
```

```
  vim.cmd('wincmd +')  -- Maximize height
end, { desc = 'Maximize current window' })
```

Source: vim.fandom.com

68.45 Quick Window Resizing with Keyboard Shortcuts

Category: window_management
Tags: key-mapping, window-resize, productivity

Adds intuitive keyboard shortcuts to quickly resize Vim/Neovim windows horizontally and vertically without complex chording

Example: Example (Vim)

```
if bufwinnr(1)
  map <kPlus> <C-W>+
  map <kMinus> <C-W>-
  map <kDivide> <c-w><
  map <kMultiply> <c-w>>
endif
```

Example: Example (Neovim)

```
if vim.fn.bufwinnr(1) ~= -1 then
  vim.keymap.set('n', '<kPlus>', '<C-W>+', { noremap = true, silent = true
  ↪ })
  vim.keymap.set('n', '<kMinus>', '<C-W>-', { noremap = true, silent = true
  ↪ })
  vim.keymap.set('n', '<kDivide>', '<C-W><', { noremap = true, silent = true
  ↪ })
  vim.keymap.set('n', '<kMultiply>', '<C-W>>', { noremap = true, silent =
  ↪ true })
end
```

Source: vim.fandom.com

68.46 Quick Window Resizing with Keymaps

Category: window_management
Tags: window-resize, key-mapping, ui

Easily resize Vim/Neovim windows using function keys with different modifier behaviors

Example: Example (Vim)

```
" Resize windows using F8 and F9
:map <F8> :set columns+=10<CR>
:map <S-F8> :set columns-=10<CR>
:map <C-F8> :set columns=132<CR>
:map <M-F8> :set columns=80<CR>

:map <F9> :set lines+=5<CR>
:map <S-F9> :set lines-=5<CR>
:map <C-F9> :set lines=60<CR>
:map <M-F9> :set lines=30<CR>
```

Example: Example (Neovim)

```
-- Resize windows using function keys
vim.keymap.set('n', '<F8>', ':set columns+=10<CR>', { desc = 'Increase
 ↪  window width' })
vim.keymap.set('n', '<S-F8>', ':set columns-=10<CR>', { desc = 'Decrease
 ↪  window width' })
vim.keymap.set('n', '<C-F8>', ':set columns=132<CR>', { desc = 'Set standard
 ↪  wide width' })
vim.keymap.set('n', '<M-F8>', ':set columns=80<CR>', { desc = 'Set standard
 ↪  narrow width' })

vim.keymap.set('n', '<F9>', ':set lines+=5<CR>', { desc = 'Increase window
 ↪  height' })
vim.keymap.set('n', '<S-F9>', ':set lines-=5<CR>', { desc = 'Decrease window
 ↪  height' })
vim.keymap.set('n', '<C-F9>', ':set lines=60<CR>', { desc = 'Set standard
 ↪  tall height' })
vim.keymap.set('n', '<M-F9>', ':set lines=30<CR>', { desc = 'Set standard
 ↪  short height' })
```

Source: vim.fandom.com

68.47 Quick file explorer

Category: Window Management
Tags: explorer, netrw, files, browse

Use built-in file explorer (netrw) for quick file navigation and management.

Example: Example (Vim)

```
: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
```

Source: Community contributed

68.48 Quickly Resize Windows in Neovim

Category: window_management
Tags: window-resize, split-management, key-mapping

Provides multiple methods to resize split windows quickly, with keyboard shortcuts and command-line commands

Example: Example (Vim)

```
" Resize window vertically
Ctrl-w +  " Increase height by 1 line
Ctrl-w -  " Decrease height by 1 line
Ctrl-w >  " Increase width by 1 column
Ctrl-w <  " Decrease width by 1 column

" Max out window height/width
Ctrl-w _   " Maximize window height
Ctrl-w |   " Maximize window width

" Proportional resizing
nnoremap <Leader>+ :exe "resize " . (winheight(0) * 3/2)<CR>
nnoremap <Leader>- :exe "resize " . (winheight(0) * 2/3)<CR>
```

Example: Example (Neovim)

```
-- Resize window functions
vim.keymap.set('n', '<C-w>+', '<C-w>+', { desc = 'Increase window height' })
vim.keymap.set('n', '<C-w>-', '<C-w>-', { desc = 'Decrease window height' })
vim.keymap.set('n', '<C-w>>', '<C-w>>', { desc = 'Increase window width' })
vim.keymap.set('n', '<C-w><', '<C-w><', { desc = 'Decrease window width' })

-- Proportional resizing
vim.keymap.set('n', '<Leader>+', function()
  local current_height = vim.api.nvim_win_get_height(0)
  vim.api.nvim_win_set_height(0, math.floor(current_height * 1.5))
end, { desc = 'Increase window height by 1.5x' })

vim.keymap.set('n', '<Leader>-', function()
  local current_height = vim.api.nvim_win_get_height(0)
  vim.api.nvim_win_set_height(0, math.floor(current_height * 2/3))
end, { desc = 'Decrease window height by 0.67x' })
```

Source: vim.fandom.com

68.49 Resize Terminal for Diff View

Category: window_management
Tags: terminal, xterm, window-resize

Automatically resize terminal window when running vimdiff to provide more horizontal space for comparing files

Example: Example (Vim)

```
if &diff
  set columns=140
endif
```

Example: Example (Neovim)

```
if vim.o.diff then
  vim.o.columns = 140
end
```

Source: vim.fandom.com

68.50 Resize Windows with Alt+Arrow Keys

Category: window_management
Tags: key-mapping, window-resize

Quickly resize windows vertically using Alt+Left/Right arrow keys

Example: Example (Vim)

```
map <M-right> <Esc>:resize +2 <CR>
map <M-left> <Esc>:resize -2 <CR>
```

Example: Example (Neovim)

```
vim.keymap.set('n', '<M-Right>', ':resize +2<CR>', { silent = true })
vim.keymap.set('n', '<M-Left>', ':resize -2<CR>', { silent = true })
```

Source: vim.fandom.com

68.51 Resize Windows with Ctrl + Cursor Keys

Category: window_management
Tags: window-resize, key-mapping

Resize split windows relative to neighboring windows using Ctrl and cursor keys

Example: Example (Vim)

```
" Resize split windows
nmap <C-Left> <C-W>-<C-W>-
nmap <C-Right> <C-W>+<C-W>+
nmap <C-Up> <C-W>><C-W>>
nmap <C-Down> <C-W><<C-W><
```

Example: Example (Neovim)

```
-- Resize split windows
vim.keymap.set('n', '<C-Left>', '<C-W>-<C-W>-', { desc = 'Decrease
 ↪  horizontal split size' })
vim.keymap.set('n', '<C-Right>', '<C-W>+<C-W>+', { desc = 'Increase
 ↪  horizontal split size' })
vim.keymap.set('n', '<C-Up>', '<C-W>><C-W>>', { desc = 'Increase vertical
 ↪  split size' })
vim.keymap.set('n', '<C-Down>', '<C-W><<C-W><', { desc = 'Decrease vertical
 ↪  split size' })
```

Source: vim.fandom.com

68.52 Restore Current Window/Buffer After Global Command

Category: window_management
Tags: buffer-navigation, window-management, automation

Create custom commands that perform actions across windows/buffers while preserving the original window or buffer context

Example: Example (Vim)

```
function! BufDo(command)
  let currBuff=bufnr("%")
  execute 'bufdo ' . a:command
  execute 'buffer ' . currBuff
endfunction
com! -nargs=+ -complete=command Bufdo call BufDo(<q-args>)
```

Example: Example (Neovim)

```
function _G.buf_do(command)
  local current_buf = vim.api.nvim_get_current_buf()
  for _, buf in ipairs(vim.api.nvim_list_bufs()) do
    vim.api.nvim_set_current_buf(buf)
    vim.cmd(command)
  end
  vim.api.nvim_set_current_buf(current_buf)
end
vim.api.nvim_create_user_command('BufDo', function(opts)
  _G.buf_do(opts.args)
end, { nargs = '+' })
```

Source: vim.fandom.com

68.53 Rolodex Window Mode

Category: window_management
Tags: window-layout, productivity, display

Create a dynamic window layout where the current window is full height and others are minimized

Example: Example (Vim)

```
set noequalalways winminheight=0 winheight=9999 helpheight=9999
```

Example: Example (Neovim)

```
vim.opt.equalalways = false
vim.opt.winminheight = 0
vim.opt.winheight = 9999
vim.opt.helpheight = 9999
```

Source: vim.fandom.com

68.54 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: Example (Neovim)

```
-- 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'
↪  })
```

Source: Community contributed

68.55 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: Example (Vim)

```
Ctrl+w f  " split and open file under cursor
Ctrl+w ]  " split and jump to tag
Ctrl+w x  " exchange current window with another
```

Source: Community contributed

68.56 Synchronous Scrolling in Multiple Windows

Category: window_management
Tags: split-windows, scrolling, window-navigation

Enable synchronized scrolling across multiple Vim/Neovim windows, allowing simultaneous vertical scrolling

Example: Example (Vim)

```
:set scrollbind
:set scb! (to toggle)
```

Example: Example (Neovim)

```
-- Set scrollbind for current window
vim.wo.scrollbind = true

-- Toggle scrollbind
vim.cmd('set scb!')
```

Source: vim.fandom.com

68.57 Tab management

Category: Window Management
Tags: tabs, navigation, workspace, organize

Use tabs as workspaces to organize different projects or contexts.

Example: Example (Vim)

```
: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
```

Source: Community contributed

68.58 Toggle Line Numbers Across All Windows

Category: window_management
Tags: key-mapping, toggle, multi-window

Toggle line numbers simultaneously in all open windows, maintaining current window focus

Example: Example (Vim)

```
func! s:NumToggle()
  let s:current_winnr = winnr()
  windo set invnumber
  exec s:current_winnr."winc w"
endfunc
map <F12> :call <SID>NumToggle()<CR>
```

Example: Example (Neovim)

```
local function num_toggle()
  local current_win = vim.api.nvim_get_current_win()
  vim.cmd('windo set invnumber')
  vim.api.nvim_set_current_win(current_win)
end

vim.keymap.set('n', '<F12>', num_toggle, { desc = 'Toggle line numbers in
↪  all windows' })
```

Source: vim.fandom.com

68.59 Toggle Window Maximization Without Losing Splits

Category: window_management
Tags: window-layout, toggle-maximize, session-preservation

Provides a function to maximize the current window and easily restore the previous split layout without losing window configuration

Example: Example (Vim)

```
function! MaximizeToggle()
  if exists("s:maximize_session")
    exec "source " . s:maximize_session
    call delete(s:maximize_session)
    unlet s:maximize_session
    let &hidden=s:maximize_hidden_save
    unlet s:maximize_hidden_save
  else
    let s:maximize_hidden_save = &hidden
    let s:maximize_session = tempname()
    set hidden
    exec "mksession! " . s:maximize_session
    only
  endif
endfunction

nnoremap <C-W>O :call MaximizeToggle()<CR>
nnoremap <C-W>o :call MaximizeToggle()<CR>
nnoremap <C-W><C-O> :call MaximizeToggle()<CR>
```

Example: Example (Neovim)

```
local function maximize_toggle()
  if vim.g.maximize_session then
    vim.cmd('source ' .. vim.g.maximize_session)
    os.remove(vim.g.maximize_session)
    vim.g.maximize_session = nil
    vim.o.hidden = vim.g.maximize_hidden_save
  else
    vim.g.maximize_hidden_save = vim.o.hidden
    vim.g.maximize_session = vim.fn.tempname()
    vim.o.hidden = true
    vim.cmd('mksession! ' .. vim.g.maximize_session)
    vim.cmd('only')
  end
end

vim.keymap.set('n', '<C-W>O', maximize_toggle)
vim.keymap.set('n', '<C-W>o', maximize_toggle)
vim.keymap.set('n', '<C-W><C-O>', maximize_toggle)
```

Source: vim.fandom.com

68.60 Toggle Window Maximize/Restore

Category: window_management
Tags: window, toggle, maximize

Creates a function to toggle between maximizing and restoring a window, useful for quickly changing window size

Example: Example (Vim)

```
let w:windowmaximized = 0
function! MaxRestoreWindow()
  if w:windowmaximized == 1
    let w:windowmaximized = 0
    :simalt ~r
  else
    let w:windowmaximized = 1
    :simalt ~x
  endif
endfunction
map <F5> :call MaxRestoreWindow()<CR>
```

Example: Example (Neovim)

```
local windowMaximized = false

local function maxRestoreWindow()
  windowMaximized = not windowMaximized
  if windowMaximized then
```

```
    vim.cmd('simalt ~x')  -- Maximize
  else
    vim.cmd('simalt ~r')  -- Restore
  end
end

vim.keymap.set('n', '<F5>', maxRestoreWindow, { desc = 'Toggle window
↪  maximize' })
```

Source: vim.fandom.com

68.61 Use Window-Local Variables for Initialization

Category: window_management
Tags: variables, window-local, initialization

Demonstrates how to use window-local variables to track and manage window-specific settings

Example: Example (Vim)

```
" Create an augroup for window initialization
augroup WindowInit
  au!
  autocmd VimEnter * let w:created=1
augroup END
```

Example: Example (Neovim)

```
-- Create an augroup for window initialization
local window_init_group = vim.api.nvim_create_augroup('WindowInit', { clear
↪  = true })
vim.api.nvim_create_autocmd('VimEnter', {
  group = window_init_group,
  callback = function()
    vim.w.created = true
  end
})
```

Source: vim.fandom.com

68.62 Window Splitting and Sizing Control

Category: window_management
Tags: window-layout, ui-configuration

Control window sizing and option inheritance when splitting windows

Example: Example (Vim)

```
set equalalways
set eadirection=both
" Local options are copied to new windows
```

Example: Example (Neovim)

```
vim.opt.equalalways = true
vim.opt.eadirection = 'both'
-- Neovim inherits local options when splitting windows
```

Source: vim.fandom.com

68.63 Window Zooming Toggle

Category: window_management
Tags: window-manipulation, productivity, toggle

Easily maximize the current window while minimizing others, with a toggle functionality

Example: Example (Vim)

```
function! ToggleMaxWins()
  if exists('g:windowMax')
    au! maxCurrWin
    wincmd =
    unlet g:windowMax
  else
    augroup maxCurrWin
        au! WinEnter * wincmd _
    augroup END
    do maxCurrWin WinEnter
    let g:windowMax=1
  endif
endfunction
nnoremap <Leader>max :call ToggleMaxWins()<CR>
```

Example: Example (Neovim)

```
local function toggle_max_wins()
  if vim.g.window_max then
    vim.cmd('wincmd =')
    vim.g.window_max = nil
  else
    vim.api.nvim_create_augroup('max_curr_win', { clear = true })
    vim.api.nvim_create_autocmd('WinEnter', {
      group = 'max_curr_win',
      callback = function()
        vim.cmd('wincmd _')
      end
```

```
    })
    vim.g.window_max = true
  end
end

vim.keymap.set('n', '<Leader>max', toggle_max_wins, { desc = 'Toggle window
↪  maximization' })
```

Source: vim.fandom.com

68.64 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: Example (Vim)

```
Ctrl+w c  " close current window
Ctrl+w o  " close all other windows
Ctrl+w q  " quit current window
```

Source: Community contributed

68.65 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: Example (Vim)

```
: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)
```

Source: Community contributed

68.66 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: Example (Vim)

```
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
```

Source: Community contributed

68.67 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: Example (Vim)

```
Ctrl+w t  " go to top window
Ctrl+w b  " go to bottom window
```

Source: Community contributed

CHAPTER 69

Workflow patterns

69.1 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: Example (Vim)

```
" 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
```

69.2 Backup and recovery workflow

Category: Workflow Patterns
Tags: backup, recovery, safety, workflow, protection

Systematic backup and recovery workflow patterns for data protection.

Example: Example (Vim)

```
" 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()
```

69.3 Build and deployment workflow

Category: Workflow Patterns
Tags: build, deployment, release, workflow, automation

Integrated build and deployment workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.4 Code quality and standards workflow

Category: Workflow Patterns
Tags: quality, standards, lint, format, workflow

Systematic code quality and standards enforcement workflow patterns.

Example: Example (Vim)

```
" 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
```

69.5 Code review and annotation workflow

Category: Workflow Patterns
Tags: review, annotation, comment, feedback, collaboration

Systematic code review and annotation patterns for collaboration.

Example: Example (Vim)

```
" 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
```

69.6 Code review and collaboration workflow

Category: Workflow Patterns
Tags: collaboration, review, feedback, team, workflow

Systematic collaboration and code review workflow patterns.

Example: Example (Vim)

```
" 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>)
```

69.7 Configuration management workflow

Category: Workflow Patterns
Tags: configuration, dotfiles, settings, management, sync

Systematic configuration and dotfile management patterns.

Example: Example (Vim)

```
" 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>
```

69.8 Documentation workflow

Category: Workflow Patterns
Tags: documentation, writing, markdown, workflow, notes

Efficient documentation and note-taking workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.9 Error handling and debugging patterns

Category: Workflow Patterns
Tags: error, debugging, troubleshoot, workflow, pattern

Systematic error handling and debugging workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.10 Focus and distraction management

Category: Workflow Patterns
Tags: focus, distraction, productivity, workflow, zen

Patterns for maintaining focus and minimizing distractions during coding.

Example: Example (Vim)

```
" 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)
```

69.11 Git workflow integration

Category: Workflow Patterns
Tags: git, workflow, version, control, branch, merge

Comprehensive Git workflow patterns integrated with editing.

Example: Example (Vim)

```
" 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>)
```

69.12 Knowledge management workflow

Category: Workflow Patterns
Tags: knowledge, documentation, notes, learning, workflow

Systematic knowledge capture and documentation workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.13 Learning and experimentation workflow

Category: Workflow Patterns
Tags: learning, experiment, playground, practice, workflow

Structured learning and experimentation workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.14 Multi-file editing workflow

Category: Workflow Patterns
Tags: multi-file, editing, buffer, window, navigation

Efficient patterns for working with multiple files simultaneously.

Example: Example (Vim)

```
" 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
```

69.15 Performance profiling workflow

Category: Workflow Patterns
Tags: performance, profiling, optimization, workflow, analysis

Systematic performance analysis and optimization workflow patterns.

Example: Example (Vim)

```
" 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>
```

69.16 Project workspace initialization

Category: Workflow Patterns
Tags: workspace, project, initialize, setup, session

Establish consistent project workspace patterns for efficient development.

Example: Example (Neovim)

```
-- 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
↪  })
```

69.17 Refactoring workflow patterns

Category: Workflow Patterns
Tags: refactoring, code, restructure, improve, workflow

Systematic code refactoring and improvement workflow patterns.

Example: Example (Vim)

```
" 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
```

69.18 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: Example (Vim)

```
" 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
```

69.19 Search and replace workflow

Category: Workflow Patterns
Tags: search, replace, refactor, pattern, workflow

Systematic approach to search and replace operations across projects.

Example: Example (Vim)

```
" 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
```

69.20 Session and workspace persistence

Category: Workflow Patterns
Tags: session, workspace, persistence, restore, save

Patterns for saving and restoring work sessions and workspace state.

Example: Example (Vim)

```
" 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()
```

69.21 Testing and debugging workflow

Category: Workflow Patterns
Tags: testing, debugging, workflow, development, tdd

Integrated testing and debugging workflow patterns.

Example: Example (Vim)

```
" 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>
```