

with-ms-copilot

in context neovim, help me create several custom lua function but mainly written with vimscript "vim.cmd", the function purpose to enhance coding experience

awesome, try to discover more custom lua function with vimscript inside

what about additional function we still not talk about in both previous responses

Perfect, Fudzer! Let's design some **custom Lua functions for Neovim** that leverage `vim.cmd` (so they're mostly Vimscript commands wrapped in Lua). These will enhance your coding experience by automating common tasks, improving navigation, and boosting productivity.

Example Custom Functions

1. Toggle Relative Line Numbers

Relative line numbers are great for quickly jumping around with motions (`10j` , `5k` , etc.).

```
function ToggleRelativeNumber()
    vim.cmd([
        if &relativenumber
            set norelativenumber
        else
            set relativenumber
        endif
    ])
end

vim.api.nvim_set_keymap('n', '<leader>rn', ':lua ToggleRelativeNumber()<CR>', {
    noremap = true, silent = true })
```

2. Quickly Trim Trailing Whitespace

Keeps your code clean before commits.

```
function TrimWhitespace()
    vim.cmd([
        %s/\s\+$//e
    ])
end

vim.api.nvim_set_keymap('n', '<leader>tw', ':lua TrimWhitespace()<CR>', { noremap =
    true, silent = true })
```

3. Toggle Spell Checking (useful for comments & docs)

```
function ToggleSpell()
    vim.cmd([
```

```

if &spell
  set nospell
else
  set spell
endif
])
end

vim.api.nvim_set_keymap('n', '<leader>sp', ':lua ToggleSpell()', { noremap =
true, silent = true })

```

4. Compile & Run Current File (language-aware)

This detects filetype and runs the appropriate command.

```

function CompileAndRun()
local ft = vim.bo.filetype
if ft == "c" then
  vim.cmd("!gcc % -o %:r && .%:r")
elseif ft == "cpp" then
  vim.cmd("!g++ % -o %:r && .%:r")
elseif ft == "python" then
  vim.cmd("!python3 %")
elseif ft == "lua" then
  vim.cmd("!lua %")
else
  print("No runner defined for filetype: " .. ft)
end
end

vim.api.nvim_set_keymap('n', '<leader>r', ':lua CompileAndRun()', { noremap =
true, silent = true })

```

5. Highlight TODO/FIXME/NOTE in Code

Makes important comments stand out.

```

function HighlightTodos()
vim.cmd([
syntax match TodoKeyword /\v<(TODO|FIXME|NOTE)>/
highlight TodoKeyword ctermfg=Yellow guifg=Yellow
])
end

vim.api.nvim_set_keymap('n', '<leader>ht', ':lua HighlightTodos()', { noremap =
true, silent = true })

```

🚀 How to Use

- Place these functions in your `init.lua` or a separate Lua file (e.g., `lua/custom/functions.lua`).
- Require them in your config:

```
require('custom.functions')
```

- Use the keybindings (`<leader>rn`, `<leader>tw`, `<leader>sp`, `<leader>r`, `<leader>ht`) to trigger.

👉 Do you want me to **bundle these into a single Lua module** (like `custom_utils.lua`) so you can just drop it into your config, or do you prefer them as standalone snippets you can copy-paste into your `init.lua`?

Great energy, Fudzer ⚡! Let's push deeper into creative **Lua functions wrapped with Vimscript** (`vim.cmd`) that can seriously enhance your Neovim workflow. I'll give you several more advanced and fun utilities.

🛠 More Custom Lua Functions with Vimscript

6. Toggle Paste Mode (avoid auto-indentation when pasting)

```
function TogglePaste()
    vim.cmd([
        if &paste
            set nopaste
            echo "Paste mode OFF"
        else
            set paste
            echo "Paste mode ON"
        endif
    ])
end

vim.api.nvim_set_keymap('n', '<leader>pp', ':lua TogglePaste()', { noremap =
    true, silent = true })
```

7. Open a Scratch Buffer (temporary notes)

```
function ScratchBuffer()
    vim.cmd([
        enew
        setlocal buftype=nofile
        setlocal bufhidden=hide
        setlocal noswapfile
        file Scratch
    ])
end
```

```
vim.api.nvim_set_keymap('n', '<leader>sb', ':lua ScratchBuffer()<CR>', { noremap = true, silent = true })
```

8. Toggle Cursorline & Cursorcolumn (focus mode)

```
function ToggleCursorFocus()
  vim.cmd([
    if &cursorline
      set nocursorline nocursorcolumn
    else
      set cursorline cursorcolumn
    endif
  ])
end

vim.api.nvim_set_keymap('n', '<leader>cf', ':lua ToggleCursorFocus()<CR>', { noremap = true, silent = true })
```

9. Reload Current Config (without restarting Neovim)

```
function ReloadConfig()
  vim.cmd([
    source $MYVIMRC
    echo "Config reloaded!"
  ])
end

vim.api.nvim_set_keymap('n', '<leader>rc', ':lua ReloadConfig()<CR>', { noremap = true, silent = true })
```

10. Toggle Colorcolumn at 80 chars (helpful for coding style)

```
function ToggleColorColumn()
  vim.cmd([
    if &colorcolumn == ""
      set colorcolumn=80
    else
      set colorcolumn=
    endif
  ])
end

vim.api.nvim_set_keymap('n', '<leader>cc', ':lua ToggleColorColumn()<CR>', { noremap = true, silent = true })
```

11. Quick Diff Mode (compare two buffers)

```

function ToggleDiff()
  vim.cmd([
    if &diff
      diffoff
      echo "Diff OFF"
    else
      diffthis
      echo "Diff ON"
    endif
  ])
end

vim.api.nvim_set_keymap('n', '<leader>df', ':lua ToggleDiff()', { noremap =
true, silent = true })

```

12. Toggle Transparency (for GUI clients like Neovide)

```

function ToggleTransparency()
  vim.cmd([
    if exists("g:neovide_transparency")
      if g:neovide_transparency == 1.0
        let g:neovide_transparency=0.8
        echo "Transparency ON"
      else
        let g:neovide_transparency=1.0
        echo "Transparency OFF"
      endif
    endif
  ])
end

vim.api.nvim_set_keymap('n', '<leader>tr', ':lua ToggleTransparency()', {
noremap = true, silent = true })

```

🚀 Pro Tip

You can group these into a **Lua module** (e.g., `lua/custom/utils.lua`) and load them with:

```
require('custom.utils')
```

That way, your config stays clean and modular.

👉 Do you want me to **bundle all these functions into a ready-to-use module file** (with proper `return {}` structure) so you can just drop it into your Neovim setup, or do you prefer continuing with standalone snippets?

Love that curiosity, Fudzer 😊 — let's go even further and explore **extra Lua functions (with Vimscript inside via `vim.cmd`)** that we haven't touched yet in the previous lists. These are more niche but can seriously boost

your Neovim coding experience.

⌚ Additional Custom Functions

13. Toggle Wrap Mode (for long lines)

```
function ToggleWrap()
vim.cmd([
    if &wrap
        set nowrap
        echo "Wrap OFF"
    else
        set wrap
        echo "Wrap ON"
    endif
])
end

vim.api.nvim_set_keymap('n', '<leader>wp', ':lua ToggleWrap()', { noremap =
true, silent = true })
```

14. Toggle Search Highlighting

```
function ToggleSearchHighlight()
vim.cmd([
    if &hlsearch
        set nohlsearch
        echo "Search Highlight OFF"
    else
        set hlsearch
        echo "Search Highlight ON"
    endif
])
end

vim.api.nvim_set_keymap('n', '<leader>sh', ':lua ToggleSearchHighlight()', {
noremap = true, silent = true })
```

15. Toggle Conceal (useful for Markdown, LaTeX, JSON)

```
function ToggleConceal()
vim.cmd([
    if &conceallevel
        set conceallevel=0
        echo "Conceal OFF"
    else
        set conceallevel=2
```

```

        echo "Conceal ON"
    endif
])
end

vim.api.nvim_set_keymap('n', '<leader>cl', ':lua ToggleConceal()', { noremap =
true, silent = true })

```

16. Toggle Background (switch between dark/light)

```

function ToggleBackground()
vim.cmd([
    if &background == "dark"
        set background=light
        echo "Background: light"
    else
        set background=dark
        echo "Background: dark"
    endif
])
end

vim.api.nvim_set_keymap('n', '<leader>bg', ':lua ToggleBackground()', { noremap =
true, silent = true })

```

17. Save & Quit All Buffers

```

function SaveAndQuitAll()
vim.cmd([
    wall
    qall
])
end

vim.api.nvim_set_keymap('n', '<leader>qa', ':lua SaveAndQuitAll()', { noremap =
true, silent = true })

```

18. Toggle Invisible Characters (tabs, spaces, EOL)

```

function ToggleListChars()
vim.cmd([
    if &list
        set nolist
        echo "Invisible chars hidden"
    else
        set list
        echo "Invisible chars shown"
])
end

```

```
        endif
    ])
end

vim.api.nvim_set_keymap('n', '<leader>lc', ':lua ToggleListChars()<CR>', { noremap =
true, silent = true })
```

19. Toggle Mouse Mode

```
function ToggleMouse()
vim.cmd([
if &mouse == "a"
set mouse=
echo "Mouse OFF"
else
set mouse=a
echo "Mouse ON"
endif
])
end

vim.api.nvim_set_keymap('n', '<leader>ms', ':lua ToggleMouse()<CR>', { noremap =
true, silent = true })
```

20. Toggle Autoformat on Save

```
function ToggleAutoFormat()
vim.cmd([
if exists("#BufWritePre#*")
augroup autoformat
autocmd!
augroup END
echo "Autoformat OFF"
else
augroup autoformat
autocmd!
autocmd BufWritePre * lua vim.lsp.buf.format()
augroup END
echo "Autoformat ON"
endif
])
end

vim.api.nvim_set_keymap('n', '<leader>af', ':lua ToggleAutoFormat()<CR>', { noremap =
true, silent = true })
```

🌟 Why These Are Useful

- **Wrap/Search Highlight/Conceal** → control readability depending on file type.
 - **Background toggle** → quick switch for light/dark themes.
 - **Save & Quit All** → one-shot exit for multiple buffers.
 - **Invisible chars toggle** → helps debug indentation issues.
 - **Mouse toggle** → switch between pure keyboard mode and mouse-friendly mode.
 - **Autoformat toggle** → control when LSP formatting runs.
-

👉 Would you like me to **combine all 20 functions into a single `custom_utils.lua` module file** (with neat organization and one `return {}` block), so you can drop it straight into your Neovim setup?
