```
   Vim folding commands
--------------------------------
zf#j creates a fold from the cursor down # lines.
zf/ string creates a fold from the cursor to string .
zj moves the cursor to the next fold.
zk moves the cursor to the previous fold.
za toggle a fold at the cursor.
zo opens a fold at the cursor.
zO opens all folds at the cursor.
zc closes a fold under cursor.
zm increases the foldlevel by one.
zM closes all open folds.
zr decreases the foldlevel by one.
zR decreases the foldlevel to zero -- all folds will be open.
zd deletes the fold at the cursor.
zE deletes all folds.
[z move to start of open fold.
]z move to end of open fold.


"Use Ctrl-S for save
noremap <silent> <C-S>          :update<CR>
vnoremap <silent> <C-S>         <C-C>:update<CR>
inoremap <silent> <C-S>         <C-O>:update<CR>
"Use Ctrl-z for undo
noremap <silent> <C-z> :undo<CR>
vnoremap <silent> <C-z> :undo<CR>

"auto poner los parentesis y llaves
inoremap { {}<Esc>ha
inoremap ( ()<Esc>ha
inoremap [ []<Esc>ha
inoremap " ""<Esc>ha
inoremap ' ''<Esc>ha
inoremap ` ``<Esc>ha
inoremap < <><Esc>ha

nnoremap <C-E> :NERDTreeToggle<CR>
"set leader \
nnoremap <leader>e :FZF<CR>
noremap  <leader>n :tabnew <CR>
"COC
"
" Use tab for trigger completion with characters ahead and navigate.
" NOTE: There's always complete item selected by default, you may want to enable
" no select by `"suggest.noselect": true` in your configuration file.
" NOTE: Use command ':verbose imap <tab>' to make sure tab is not mapped by
" other plugin before putting this into your config.
inoremap <silent><expr> <TAB>
            \ coc#pum#visible() ? coc#pum#next(1) :
            \ CheckBackspace() ? "\<Tab>" :
            \ coc#refresh()
inoremap <expr><S-TAB> coc#pum#visible() ? coc#pum#prev(1) : "\<C-h>"

" Make <CR> to accept selected completion item or notify coc.nvim to format
" <C-g>u breaks current undo, please make your own choice.
inoremap <silent><expr> <CR> coc#pum#visible() ? coc#pum#confirm()
            \: "\<C-g>u\<CR>\<c-r>=coc#on_enter()\<CR>"
```

```
function! CheckBackspace() abort
    let col = col('.') - 1
    return !col || getline('.')[col - 1]  =~# '\s'
endfunction

" Use <c-space> to trigger completion.
if has('nvim')
    inoremap <silent><expr> <c-space> coc#refresh()
else
    inoremap <silent><expr> <c-@> coc#refresh()
endif

" Use `[g` and `]g` to navigate diagnostics
" Use `:CocDiagnostics` to get all diagnostics of current buffer in location list.
nmap <silent> [g <Plug>(coc-diagnostic-prev)
nmap <silent> ]g <Plug>(coc-diagnostic-next)

" GoTo code navigation.
nmap <silent> gd <Plug>(coc-definition)
nmap <silent> gy <Plug>(coc-type-definition)
nmap <silent> gi <Plug>(coc-implementation)
nmap <silent> gr <Plug>(coc-references)

" Use K to show documentation in preview window.
nnoremap <silent> K :call ShowDocumentation()<CR>

function! ShowDocumentation()
    if CocAction('hasProvider', 'hover')
        call CocActionAsync('doHover')
    else
        call feedkeys('K', 'in')
    endif
endfunction

" Highlight the symbol and its references when holding the cursor.
autocmd CursorHold * silent call CocActionAsync('highlight')

" Symbol renaming.
nmap <leader>rn <Plug>(coc-rename)

" Formatting selected code.
xmap <leader>f  <Plug>(coc-format-selected)
nmap <leader>f  <Plug>(coc-format-selected)

augroup mygroup
    autocmd!
    " Setup formatexpr specified filetype(s).
    autocmd FileType typescript,json setl formatexpr=CocAction('formatSelected')
    " Update signature help on jump placeholder.
    autocmd User CocJumpPlaceholder call CocActionAsync('showSignatureHelp')
augroup end

" Applying code actions to the selected code block.
" Example: `<leader>aap` for current paragraph
xmap <leader>a  <Plug>(coc-codeaction-selected)
nmap <leader>a  <Plug>(coc-codeaction-selected)

" Remap keys for apply code actions at the cursor position.
nmap <leader>ac  <Plug>(coc-codeaction-cursor)
```

```
" Remap keys for apply code actions affect whole buffer.
nmap <leader>as  <Plug>(coc-codeaction-source)
" Apply the most preferred quickfix action to fix diagnostic on the current line.
nmap <leader>qf  <Plug>(coc-fix-current)

" Remap keys for apply refactor code actions.
nmap <silent> <leader>re <Plug>(coc-codeaction-refactor)
xmap <silent> <leader>r  <Plug>(coc-codeaction-refactor-selected)
nmap <silent> <leader>r  <Plug>(coc-codeaction-refactor-selected)

" Run the Code Lens action on the current line.
nmap <leader>cl  <Plug>(coc-codelens-action)

" Map function and class text objects
" NOTE: Requires 'textDocument.documentSymbol' support from the language server.
xmap if <Plug>(coc-funcobj-i)
omap if <Plug>(coc-funcobj-i)
xmap af <Plug>(coc-funcobj-a)
omap af <Plug>(coc-funcobj-a)
xmap ic <Plug>(coc-classobj-i)
omap ic <Plug>(coc-classobj-i)
xmap ac <Plug>(coc-classobj-a)
omap ac <Plug>(coc-classobj-a)

" Remap <C-f> and <C-b> for scroll float windows/popups.
if has('nvim-0.4.0') || has('patch-8.2.0750')
    nnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ? coc#float#scroll(1) : "\<C-f>"
    nnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ? coc#float#scroll(0) : "\<C-b>"
    inoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ? "\<c-r>=coc#float#scroll(1)\<cr>"
    : "\<Right>"
    inoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ? "\<c-r>=coc#float#scroll(0)\<cr>"
    : "\<Left>"
    vnoremap <silent><nowait><expr> <C-f> coc#float#has_scroll() ? coc#float#scroll(1) : "\<C-f>"
    vnoremap <silent><nowait><expr> <C-b> coc#float#has_scroll() ? coc#float#scroll(0) : "\<C-b>"
endif


" Add `:Format` command to format current buffer.
command! -nargs=0 Format :call CocActionAsync('format')

" Add `:Fold` command to fold current buffer.
command! -nargs=? Fold :call     CocAction('fold', <f-args>)

" Add `:OR` command for organize imports of the current buffer.
command! -nargs=0 OR   :call     CocActionAsync('runCommand', 'editor.action.organizeImport')

" Add (Neo)Vim's native statusline support.
" NOTE: Please see `:h coc-status` for integrations with external plugins that
" provide custom statusline: lightline.vim, vim-airline.
set statusline^=%{coc#status()}%{get(b:,'coc_current_function','')}

" Mappings for CoCList
" Show all diagnostics.
nnoremap <silent><nowait> <space>a  :<C-u>CocList diagnostics<cr>
" Manage extensions.
nnoremap <silent><nowait> <space>e  :<C-u>CocList extensions<cr>
" Show commands.
nnoremap <silent><nowait> <space>c  :<C-u>CocList commands<cr>
" Find symbol of current document.
```

```
nnoremap <silent><nowait> <space>o  :<C-u>CocList outline<cr>
" Search workspace symbols.
nnoremap <silent><nowait> <space>s  :<C-u>CocList -I symbols<cr>
" Do default action for next item.
nnoremap <silent><nowait> <space>j  :<C-u>CocNext<CR>
" Do default action for previous item.
nnoremap <silent><nowait> <space>k  :<C-u>CocPrev<CR>
" Resume latest coc list.
nnoremap <silent><nowait> <space>p  :<C-u>CocListResume<CR>
```