Zed Cheat Sheet For Vim Users

Created by @jackkinsella (Semicolon & Sons)

General

- Everything (e.g., commands/symbols for file) is fuzzy searched - so to go to `get_read_only_methods` just type `gro` and you are likely there.
- Type `:` and you can browse random features and discover more about the editor.

Pane Management

- Toggle left: `Command B`
- Toggle right: `Command R`
- Toggle bottom (e.g., terminal): `Command J`
- Full screen: `fn shift f`
- Make bigger vertically: `control w +`
  - Smaller: `control w -`
- Make bigger horizontally: `control w >`
  - Smaller: `control w <`

Text transformations

- Camel case / snake case / upper / lower - via `:`

Git

- Open with `:G[it]`
- Expand diff hunk: `do` (mnemonic - diff open)
- View diff of file under cursor: `gf`
    - Unstage hunk
        - Vim: `dO`
        - Zed: `Command Alt Y`
    - Stage hunk
        - Vim: `dO`
    - Restore hunk
        - Zed: `Command Alt Z`
        - Vim: `dp`
- Stage all: `command control y` ("yes")
    - Stage all in current file: `Command Y`
    - Unstage all: `command control Y` (capital Y)
- Get rid of all changes: `Control G` then `backspace`
- Stage file under cursor: `x`
    - Unstage: same
- Restore file under cursor (or delete if new): `Backspace`
- Bring up commit window: `Command Enter`
    - Actually commit: `Command Enter`
- Auto-generate commit message: `Alt Tab`
- Fetch latest code: `Control G` (twice in a row)
- Diff of everything: `Control G d`
- Push code: `Control G up`
- Regular files
    - `do` (normal mode) - diff under cursor - decide to stage or not
    - move to next git change: `]c` (chunk)
- Link to current LOC on GH - `:cpp(link)`
- Select branch - `Command Alt B`
- Create branch - `Command Alt B` and start typing
- Blame - `Command Alt G` - then `B`

File Explorer

- Open with `:E[xplore]`
- Open file under cursor: `o`
- New file: `%`
- New directory: `d`

- Delete entry: `D`
- Go to parent directory: `-`
- Toggle if folder is opened: `enter`
- Rename: `R`

Selection/Tree sitter/Text objects

- Functions: `af`, `if`
- Classes: `ac`, `ic`
- Next bigger/smaller syntax: `]x` or `[x`
    - Note that this feels counterintuitive to the direction. It's about abstraction.
- Current indent level: `ii`
    - Plus one line before: `ai` (e.g., fun defs in Python)
    - Plus one line after: `aI` (e.g., function defs in JS)
- HTML tag: `at`, `it`
- A func argument
    - Including comma: `aa` ("around argument")
    - Without comma: `ia` ("inside argument")
- Moving
    - Next method start: `[m`
        - End: `[M`
    - Next section start: `[[` or `]]`
        - End: `[]`
    - Next comment: `[*`

Editor Settings

- See what a keyboard shortcut is doing: `:key_context_view`
- `Command ,` - open up settings.json file

Quality of life

- Get absolute path of current file: `:copy_path`
    - Relative path: `:copy_relative_path`

- Save without formatting the file (e.g., if auto formatter would introduce a big diff): `:save_without_formatting` or `Command K s`
- Get a link to the current line on GitHub to send to colleagues: `:cpplink`

LSP

- Next diagnostic: `]d`
- Show error for symbol: `gh`
- Hover definition: `K`
- Go to definition: `gd`
- See all uses: `gA`
- See list of all symbols (e.g., methods, vars) in file: `gs/`
    - Fuzzy search
    - Filter for classes by typing "class"
- See list of all symbols (e.g., methods, vars) in project: `gS`
- Change name: `cd` (can be called when not on definition)
- Code actions: `g.`
    - `Control n` / `Control p` to navigate... or arrow keys
    - `Enter` to select
        - Lint errors - can fix or add ignore automatically!

Editing

- Multi-cursor
    - Next instance of current word: `Command D`
    - Multiple lines
        - Vim mode: `Control V` for visual block then `I` or `A`
        - Zed mode: `Command Alt up/down`
    - `Esc` to leave
- Exchange
    - `cx` - then text obj (e.g. `cxiw`) then do the same on the item to exchange position
- Add line above/under cursor and start text (from vim but I never used much): `o` or `O`
- Jump to nearest enclosing bracket: `%` (this can be within an enclosed area)

Navigation

- Search project files: `Command P`
- Re-open closed tab: `Command Shift T`
- `Control Tab` - switch tabs
- Project wide search:
  - Trigger
    - Vim mode: `g/`
    - Zed: `Command Shift F`
  - Open file under: `g space`
    - In split: `Control W space`
- Buffer left / right: `[b` and `]b` or alternatively `command alt left/right`
- Close buffer
  - `Command W`
  - Vim: `:q`
- Search all current tabs by name: `:ls`
  - Remove one of them in the list: `Control Backspace`
- Open a new file in a split: `:new`
  - Vertical split: `:vne`
- Open a new file in a new tab: `:tabnew`
  - or `Command N`... probably easiest
- Close all open buffers: `:qa`
- Close all open buffers and write: `:wqa`
- Save all open buffers: `:wa`
- Outline panel: `Command Shift B`

Find and Replace

- Trigger menu: `Command Shift F`
  - Vim mode: `g/`
- Toggle case-sensitive: `Command alt c` (c for "case")
- Toggle match whole word: `Command alt w` (w for "word")
- Toggle using regex: `Command alt x` (x for "expression")
- Toggle filter: `Command alt f` (f for "filters")
- Toggle replace menu: `Command shift H` (h for "hotswap" - yea I know this mnemonic is more of a stretch)
  - Replace next: `Enter`
  - Replace all: `Command Enter`

- Browse through these
    - `n` or `N` as in Vim

Terminal

- Open with `:T[erm]`
- Open new terminal tab: `Control Shift \`
- Move between terminals: `Command Alt Arrow`
- Move to last used one: `Control Tab`
- Search output of terminal: `Command f`
- Clear output: `Command K`
- Switch bottom panel to show terminal: `Control ~`

AI

- Open up
    - Separate AI window with `:AI`
    - In-line AI box from a particular line of the file being edited: `Control X Control A`
    - `Esc` to leave
- Switch agent profile (e.g., write vs ask): `Command I`
- New AI thread (clear context): `Command N`
- Add context (files, functions, etc.) menu: `Command Shift A` ("add")
    - Then you can select current file, for example
- Remove context: `Command alt e` (e for "end")
- Insert selected text as context: `Command >` (i.e., `Command shift` and `.`)
- Cancel an AI request in progress: `Esc`
- Open the rules library: `Command Alt p`
- Show AI settings: `Command Alt c`
    - Go back from settings to AI area again: `Control -`

Code Folding

- Fold under cursor: `za` ("away")

    - Unfold: `zA` (opposite of "away")
- Fold all in file: `zM` ("minimize")
- Open all in file: `zR` ("reveal")

Debugger

- Trigger the pane with `F4`
- Set a breakpoint on the relevant LOC: `F9`
- Step over: `F7`
- Terminate all threads: `Shift F5`
- Switch over to debug panel if you have multiple bottom panels going: `Command Shift D`

Tasks

- Add new task: `zed: open_project_tasks`
    - Or look up `tasks.json`
- Select task to run: `Command Shift R`
- Re-run previous one: `command alt r`
- Variables
    - Currently selected text: `$ZED_SELECTED_TEXT`