Editors (Vim)

Writing English words and code are distinct tasks. For programming, efficient navigation and editing are key, hence specialized editors like Vim are essential.

Learning a New Editor

- **Tutorial**: Begin with guided learning.
- **Consistency**: Use the editor for all text editing tasks.
- **Curiosity**: Look up better ways to perform tasks.

Mastering an Editor

- **Hour 1-2**: Learn basic functions.
- **Hour 20**: Match previous editor speed.
- **Beyond**: Save time with advanced knowledge and muscle memory.

Which editor to learn?

Programmers have strong opinions about editors.

- **Popular Editors**:
  - Visual Studio Code
  - Vim

Vim

Vim, stemming from Vi (1976), is a powerful command-line-based editor, still evolving and supported by many tools through Vim emulation modes.

Philosophy of Vim

Vim is a *modal* editor designed for efficiency:

- **Modes**: Inserting vs. Manipulating text.
- **Programmability**: With Vimscript and languages like Python.
- **Keystroke Commands**: Mnemonic keystroke commands are composable.
- **Mouseless Operation**: Prioritizes keyboard use over the mouse.

Modal Editing in Vim

Vim's modes optimize for less writing and more editing:

- **Normal**: Navigate and edit.
- **Insert**: Insert text.
- **Replace**: Replace text.
- **Visual**: Select text.
- **Command-line**: Execute commands.

Mode-switching is done using keystrokes like `<ESC>`, `i`, `R`, `v`, `V`, `<C-v>`, and `:`.

Basics of Vim

Inserting Text

Switch to Insert mode with `i` and return to Normal with `<ESC>`.

Buffers, Tabs, and Windows

- **Buffers**: Open files.
- **Tabs/Windows**: Organize views and splits.

Basics of Vim (cont.)

Command-line Mode

Commands like `:q`, `:w`, `:wq`, `:e {file}`, `:ls`, and `:help {topic}` control file operations and more.

Vim’s Interface as a Programming Language

Vim's keystrokes, the "nouns" and "verbs", are commands that are composable for efficient editing.

Movement Commands

- **Basic**: `hjkl`
- **Words**: `w`, `b`, `e`
- **Lines**: `0`, `^`, `$`
- **Screen**: `H`, `M`, `L`
- **Scroll**: `Ctrl-u`, `Ctrl-d`
- **File**: `gg`, `G`
- **Line numbers**: `:{number}<CR>`, `{number}G`
- **Find**: `f{char}`, `t{char}`, `F{char}`, `T{char}`, `,`, `;`
- **Search**: `/{regex}`, `n`, `N`

Visual Selection and Editing

- **Visual Modes**: `v`, `V`, `Ctrl-v`
- **Editing Commands**: `i`, `o`, `O`, `d{motion}`, `c{motion}`, `x`, `s`, `u`, `<C-r>`, `y`, `p`

Counts and Modifiers

- **Counts**: Repeat actions with a number (e.g., `3w`, `5j`).
- **Modifiers**: Modify the scope of actions (e.g., `ci(`, `da'`).

Demo: Fixing Fizz Buzz

A broken fizz buzz implementation is fixed using Vim, demonstrating the editor's efficiency and command composition.

```
def fizz_buzz(limit):
    for i in range(limit):
        if i % 3 == 0:
            print('fizz')
        if i % 5 == 0:
            print('fizz')
        if i % 3 and i % 5:
            print(i)

def main():
    fizz_buzz(10)
```

Customizing Vim

- Use `~/.vimrc` for a personalized setup.
- Start with a basic config from here.

Extending Vim with Plugins

No need for a plugin manager; use Vim's built-in package management system.
Favorite plugins include:

- ctrlp.vim
- ack.vim
- nerdtree
- vim-easymotion

Vim-mode in Other Programs

Vim emulation is available in many tools, enhancing productivity across various environments.

- **Shell**: `set -o vi`, `bindkey -v`, `fish_vi_key_bindings`
- **Readline**: `set editing-mode vi` in `~/.inputrc`
- **Browsers**: Vimium for Chrome, Tridactyl for Firefox
- **Jupyter notebooks**: Vim bindings available

Advanced Vim Techniques

Advanced features like search and replace, window management, and macros showcase Vim's power.

- **Search and Replace**: `:s/foo/bar/g`
- **Windows**: `:sp`, `:vsp`
- **Macros**: Record with `q{char}`, replay with `@{char}`

Resources for Learning Vim

- `vimtutor`
- Vim Adventures
- Vim Tips Wiki
- Vimways
- Vim Golf
- Vi/Vim Stack Exchange
- Vimcasts
- Practical Vim (book)