Wrap Your Mind Around Vim 8.0

John Filippone

August 19, 2017

Contents

Chapter 1

Preamble

1.1 Purpose

The object of this book is to provide the key knowledge and frame of mind necessary for the reader to achieve a highly efficient workflow with Vim and to gain command over its vast capabilities.

1.2 Vim at a Glance

As stated on vim.org: Vim is a highly configurable text editor built to make creating and changing any kind of text very efficient. It is included as "vi" with most UNIX systems and with Apple OS X.

Vi was developed for UNIX and first released in 1978. "Vim" is short for Vi Improved. Vim was initially modeled after vi and since the first release of Vim in 1991, Vim development has been highly active with the most recent release to date of this book being Vim 8.0 in 2016.

1.3 Important Historical Perspective

When Vim was created, it was made by Bram for Bram; meaning, the engineer who developed Vim did so for his personal use. He developed the features that he personally wanted to use. As years went on and Bram continued developing, the community asked for other features that Bram did

not necessarily care about. He implemented them anyway as long as he felt enough people would enjoy having them. After decades of development, Vim has become a vast collection of features. There are too many for any one person to really need or want or use them all. The expansive collection of features is meant to accommodate a diverse set of users where each user only needs and/or wants a small subset of all features to achieve their ideal workflow. It is for this reason that this book focuses on understanding Vim rather than a complete learning of the tool. Getting the maximum utility from Vim on your particular workflow does not require you to know how to use all the features but rather (usually) only 5-10 percent of them.

1.4 This Book's Unique Approach

This is not a 'HOW to use Vim' book. Instead, this book focuses on teaching WHAT Vim can do. The approach of this book is based on the premise that learning HOW to use a particular feature in Vim is easy; the hard part is figuring out the WHAT feature you want to learn. Because Vim is such an extensively customizable tool, the HOW of a feature is not particularly important when it comes to achieving efficient workflow. If the default invocation of a feature does not work for you, there is almost definitely a way to change the invocation to something that does work. Any how-to elements of this book will be minimal for the sake of brevity. This book will instead into detail on the WHAT is possible in Vim and WHAT the proper terminology is. Once you know what the thing you want to do is and what it is called, all it takes some internet searching and some practice. If you try to dive right into learning all of Vim without this type of understanding you will waste a lot of time. You will never use most of what you learn. It is most effective to understand Vim first and understand what it has to offer your personal workflow, then only learn those parts of Vim. After reading this book you will have effectively wrapped your mind around Vim. You will have a strong understanding of the majority of the moving parts of Vim and the functionality that they offer. Armed with that understanding, you can quickly determine the set of features you need for your particular workflow. More importantly, the next time your workflow changes, your knowledge of Vim will allow you to quickly assess whether or not Vim is the tool for the job (more than likely it is) and you will know what to search for when learning how to invoke the functionality you want.

1.5 How Most People Learn Vim

Most people are thrust into Vi and/or Vim and are forced to frantically learn the basic editing commands. Even very smart users get stuck in Vim because they do not understand the capabilities of the tool or the technical colloquialisms associated with it. When struggling with a problem, the new user has two main issues. They do not know that Vim is capable of doing the thing they want or they know what they want and suspect that Vim can do it but have no idea what the feature is called. Unfortunately, most books for Vim are recipe books. They provide tutorial on specific features or describe great ways to solve common problems with Vim. These are great for picking up tricks but you will brush through hundreds of pages before learning the key set of Vim features that work best for you and the task you are applying them to. Most will try these 'HOW to' approaches, then quickly give up on them and default to being satisfied with a cumbersome workflow. The internet can be very helpful if you know what you are looking for as long as you don't fall into a wormhole of mystifying power-user language. The weird dialog about buffers and Ex-mode and autocommands may be easy enough for you to sift through but the real trouble is the former. When you start using Vim, you have no idea what you need to learn! Most have a vague idea of what functionality they want, but do not know the proper Vim terminology, making internet searching for a how-to very difficult. Most users will pick up a trick here and there over the years from their coworkers or on the internet after getting fed up with a difficult workflow, but never fully feel comfortable working with Vim.

1.6 How You Ought To Learn Vim

1. Understand what Vim has to offer. (Wrap Your Mind Around Vim)
2. Pick a subset of key features you need to learn in order to have an efficient workflow for your specific task.
3. Learn each of these features through the Vim help files, books, and/or the internet.
4. Practice using them enough to commit them to muscle memory for a truly efficient workflow.

5. When you change your workflow, use your understanding from step one and revert to step 2.

This is an efficient method of learning Vim. Only learn the features you actually care to use. Know what the tool can do so that later on down the road when you need some other functionality, you know what to look for. This book aids you with the most important step in the process: understanding Vim.

1.7 How to Use This Book

You can read this straight through or you can skip to the parts you really care about. It will be valuable to at least skim all of the material so that you come away with an understanding of the full scope of Vim capabilities. If you are mystified by the terminology in this book, refer to the Vim Terminology chapter.

Chapter 2

Vim Terminology

These are words that you may come accross in this book or on the internet in a Vim context. Many of these can be very mystifying if you don't understand their special Vim meaning. Throughout this book, these words are in bold when used in the context detailed in this section. For example if this book uses the word "buffer" but it is not bold, think in terms of a normal dictionary definition. If it is bold, use the meaning detailed in this section.

abbreviations alternate file (saved in the hashtag register) or alternate buffer argument list autocommands bottom-line command (includes searches and anything on bottom line that is not formally a command) buffers (active, inactive, hidden, unlisted, modifiable, read-only, read errors) commands and colon commands ctags dotfiles edit (as a unit used in undo) events ex editor instance of Vim jump and the distinction between and jump and a motion (if it goes into the jump list it is a jump; motions are always within the file; if it goes to another buffer it is a jump; some motions within file are classified as jumps and go in the jump list) macros mapping modes (normal, command, ex, insert, visual, select, operator pending, insert normal mode, replace mode) motions movement command NERDTree operator options Pathogen pattern in context of searching plugins power user put v paste registers scripts session special buffers (quickfix, help, directory, scratch) status line swp or swap files tab pages or tabs tag stacking or tag stacks text objects (block and non block text objects) toggle options (note the set option—nooption—option?—option!—option(ampersand) Syntax) transparent editing vi vimrc VIMRUNTIME directory Vundle windows word WORD yank v copy row column modelines buffer list current selection and real clipboard from user manual page 73

Chapter 3

Mechanisms of The Vim Universe

3.1 Ways to Access Functionality in Vim

Most editors are quite similar in the way that functionality is accessed. They usually involve point and click menues and toolbars. Vim offers a lot more ways to do things. There are a lot of moving parts and it can get confusing. This section will help you wrap your mind around all the Vim constructs used to invoke and alter functionality.

3.1.1 Command Line Arguments

When you start Vim from the command line you can invoke functionality with the arguments and options. Generally, you will run Vim with a single filename being the only argument.

3.1.2 Commands

Colon Commands

Colon commands, Command-line commands, and Ex Commands are all the same.

Normal Mode Commands

Other Mode Commands

3.1.3 Options

3.1.4 Dotfiles

3.1.5 Autocommands

3.1.6 GUI

Gvim Menus

Gvim Toolbars

3.1.7 Plugins and Scripts

3.1.8 Mouse

3.1.9 Modelines

3.1.10 Shell and External Programs

comments that define options automatially p268 Oualline

3.2 Vim Command Grammars

3.3 Modes

Vim is a modal editor. That simply means that Vim is at all times in only one mode and your keystrokes will mean something different to Vim depending on the modal context in which they are typed. There are two main Modes and several others

3.4 Concept of an Edit

3.5 Buffers

3.6 Registers

3.7 Tags

3.8 Marks

3.9 Regex

Chapter 4

Capabilities

Vim is a text editor with an incredible amount of features. This section will allow you to wrap your mind around the vast set of Vim capabilities. This section will not dive very deep into how to invoke functionality. To do so would be far more information than is necessary to wrap your mind around Vim capabilities.

4.1 Navigating

4.1.1 Moving Around a File

4.1.2 Search

4.2 Editing

4.2.1 Inserting

Basic Inserts and appending

The most basic way to insert in Vim is to activate Insert Mode and start typing. The most basic method for activating Insert Mode is the Normal Mode command, i. Normal Mode is restored by pressing the ESC key. For added efficiency there are a variety of Normal Mode commands that activate Insert Mode and reposition the cursor for typing. These can be pretty nifty. Here are a few: It is possible to enable the point and click insertion that most editors use in Vim as well. See the section on Mouse Compatability.

Autocompletion

Special Characters and Foreign Language

4.2.2 Deleting

4.2.3 Replacing

4.2.4 Cut Copy Paste

4.2.5 Indenting

4.2.6 Toggling Case

4.3 Work Environment

4.3.1 Graphical User Interface (GUI)

4.3.2 File Management

4.3.3 Mouse Compatability