An Introduction to Vim

Jack Burke

October 17, 2016

Contents

Chapter 1

What is Vim

Vim is a **modal text editor** designed around efficient text editing and made to be highly configurable.

1.1 What Does That Mean

A **modal text editor** is one that is focused around switching between different modes which all have different key bindings. In Vim, the 4 main modes are Normal, Insert, Visual, and Command Line. These modes allow you to maneuver, modify, add, and select text, or run various scripts and commands, based on the current mode. By having different key bindings for modes, you have to switch between them often but not do elaborate button combinations for commands.

Efficient text editing comes from the idea that you aren't supposed to use your mouse, only the keyboard. By having many commands for editing text, or moving around it, you can be much faster at editing without having to move your hand over to the mouse. Vim also has countless useful commands that speed up the code writing process.

Vim has been around since 1991 and has developed an extensive community for plugins and modifications. Between those and Vim's base levels of configuration, Vim can be a lightweight basic text editor or an elaborate IDE with many bells and whistles.

1.2 What Is It Good For

Vi (which is the base version of Vim) was made for text editing from the command line where you could not use a mouse. Vim developed off of that adding many features such as portability, plugins, and scripts. Vim is now a flexible editor that is ideal for programming, markup languages (such as html and LaTeX), or writing plain text. In comparison to other text editors, Vim offers unparalleled speed at editing and moving through a file, over 20 years of refinement and support, heavy customization, and native integration with the command line.

1.3 Vim vs. Emacs

Vim and Emacs users have a long standing rivalry over which text editor is better. There is no clear cut better one to use; they have differences that you should learn about before you choose between the two.

- Vim is a modal editor where Emacs is based around using button combinations (i.e. shift and ctrl key)
- Emacs in its base form is an all-in-one editor, that you never have to leave. Vim is treated like a tool in your command line belt
- Vim is always installed on a server, where Emacs may not be
- Vim requires less movement of your hands and fingers to manipulate text, but being efficient at Emacs gives comparable speed

In essence, Emacs is a more all-in-one editor that has deeper levels of customization but is treated like a program that you never need to leave.

Vim is a tool used to quickly edit text, but that is all it is at its core.

I advise spending time with both, especially if the modal system of vim seems awkward to you.

1.4 Vim the Editor vs. Vim the Language

Vim as an editor is a lightweight yet customizable editor which you can use for many different text editing tasks. The language it is based off of is what makes using the editor so quick and unique. While you can't use the editor without using the language, the language can be used in other various editors as well.

Most modern IDE's have versions of Vim keybindings that you can use. While these won't work as well as if they were used on the actual Vim editor, they can allow you to utilize the speed of typing with Vim, while using a specific IDE, or editor of your choice.

Editors such as Eclipse, Intellij, Atom, Sublime, and even Emacs all have vim plugins or native support for the keybindings. Even things like web browsers, and bash have Vim modes that you can utilize.

While having keybindings for Vim is a great choice, Vim is much more than just its keybindings, which you will discover with use.

Chapter 2

Getting Started

The purpose of this workshop is an interactive tutorial that allows you to learn Vim through exercises. To begin this, you have to use git to access the workshop.

2.1 Getting the Workshop Repository

In order to get the workshop, you have to clone the repository from my GitHub. If that doesn't make sense to you, that's alright.

First, in the command line, move to your home directory

```
cd ~
```

Then, enter in

```
git clone http://github.com/JackHBurke/vimworkshop.git vimworkshop
```

This is making a copy of the files that I have stored on GitHub, and putting them in a new directory in your home directory, called vimworkshop

2.2 Creating a .vimrc

In order to customize your Vim, you need what is called a VimRC. The RC stands for Run Commands, which are commands that get run each time you open up Vim. Vim at its basic state lacks some significant quality of life improvements, which can easily be added by using a simple vimrc. A vimrc is something that you will make many modifications and additions to, but I have provided you with a basic one which you can use.

(The following steps are also located in the file basicvimrc)

First lets put this vimrc in the proper place for Vim to find it. Navigate to the newly created vimworkshop directory

```
cd ~/vimworkshop
```

Then, you want to move the file basicvimrc into your home directory, as a hidden file called .vimrc

```
cp basicvimrc ~/.vimrc
```

Then we are going to add the ability to easily add and install new plugins for Vim, by getting a commonly used repository from online.

```
git clone https://github.com/gmarik/vundle.git ~/.vim/bundle/vundle
```

Now, once we have that, go into the newly made vimrc. There should be some errors, don't worry. Just hit enter, and they will go away.

```
vim ~/.vimrc
```

You should see some changes now to your Vim. In order to get all of the changes in this VimRC, you need to enter Command Line mode of vim (by hitting :), and run the plugin install

```
:PluginInstall
```

(This is case sensitive)

Now hopefully, you can exit out of the vimrc once the install has completed, and re-enter to see the changes

```
:q
vim ~/.vimrc
```

2.3 Starting The Tutorial

The workshop is all contained in a text file, in the directory that you initially made. To get there, you type in

```
vim ~/vimworkshop/vimworkshop
```

(Yes I shouldn't have named the file the same as the directory)

And from there, you follow the instructions located in the file.

I would recommend having a cheat-sheet for Vim up, while using vim in this workshop and while using Vim in general.

`https://rumorscity.com/wp-content/uploads/2014/08/10-Best-VIM-Cheat-Sheet-03.jpg`

Is a good one, but there are many to choose from, if you google: Vim Cheat Sheet