Fold Time to Navigate Faster

EMMA JANE WESTBY

Emma Jane Westby is an internationally renowned open source software advocate, technical author, and teacher with Drupalize.Me. In January 2010 she was recognized by the Google Diversity Program for her efforts in increasing female participation in software development. Emma has been teaching Internet technology since 2002 and is the author of *Front End Drupal* and *Drupal User's Guide*. She lives in rural Canada where she raises bees, and enjoys sipping on a nice single malt scotch. You can follow her adventures on Twitter @emmajanehw.

Recently a friend of mine was drooling over a new feature in Omni-Outliner: section folding. I was a bit confused as I've been using folding in Vim for nearly a decade. I didn't think it was all that novel. I just thought it was a sane way to quickly review, and rearrange structured documents. The first memory I have of working with folds in Vim was with DocBook. It must have been turned on by default in a `.vimrc` file that I copied and pasted as I'd always thought of it as a DocBook-specific way of working with text.

Of course, the concept of folding isn't tied to DocBook. In this article, I'll show you how to: use folding in any type of file, enable folding by default in some file types, and add a few shortcuts to making working with folds a lot easier.

What Is Folding

Folding allows you to collapse lines of a structured text document into a single line, based on an arbitrary set of rules. Once folded, the range of lines within the fold is hidden, but not removed from the document.

When collapsed, a folded line of text looks like this:

```
### Fold Methods [13 lines]----------------------------------------
```

If an entire document is folded, you will see only these "summary" lines in your document.

You can test it out inside any document with the combination of two commands: fold creation (`zf`) and text object, such as a paragraph (`ap`). The complete command is: `zfap`.

The user help manual in Vim provides an excellent description, with an ASCII art diagram! To read the manual page: `:help user-manual`, then look for the section on folding (for me it's Chapter 28). You can navigate to this chapter by positioning your cursor inside the `|vertical bars|` and pressing `CTRL-]`.

Enabling Folding

Chances are very good that folding is already available in your copy of Vim. The packages for Vim on Ubuntu and via brew on OS X both come with folding compiled into the build. It is far more likely that the document type you're editing has no fold methods available.

Fold Methods

There are a number of different ways that a folding point can be defined: for example, a text object, such as a paragraph, or the number of indents at the beginning of a line. The plugin we'll be exploring in this article uses an expression to define folds.

There are lots of plugins similar to the one I found for other file types. You might want to fold files based on function, for example, or class declarations in PHP, or perhaps patch

files. A little time with your favorite search engine will probably yield at least a few relevant plugins to try out. If you're not feeling very adventurous, but you do want to know more about how Vim calculates different folding points, take a look at the manual page at `:help foldmethod`.

Markdown Folding Plugin

For this article, I was most interested in replicating OmniOutliner's folding functionality, but in Markdown files. To do this I used a plugin with a fold expression specific to Markdown syntax. You can download this plugin from GitHub at https://github.com/nelstrom/vim-markdown-folding.

Seeing as I'm using pathogen, all I needed to do to install the Markdown folding plugin was to download the plugin and place it into the directory `~/.vim/plugins`.

This plugin merely provides the definitions on what can be folded. In the case of a Markdown file, you can fold a document based on the headings. Your Markdown files can use either of the accepted syntaxes for a heading; however, if you use # to identify headings, the outline is much easier to read since there is a clue in the text as to the level of the heading.

Configuring Folding

The author of the plugin for Markdown folding recommends making a few changes to your `.vimrc` file. In addition to his configuration, I add a few of my own:

```
set nocompatible
if has("autocmd")
    filetype plugin indent on
endif
" To start, close all folds
" you will likely want this set to 1 for non-markdown files.
set foldlevelstart=0
" Use the space bar to conveniently open and close folds.
nnoremap <Space> za
vnoremap <Space> za
```

Basic Fold Commands

Assuming you're following along with the article and you've installed the Markdown folding plugin, and set the fold level to zero, you're going to have a big surprise waiting the next time you open a Markdown file in Vim. Instead of seeing your text, you'll be greeted with a list of all of the headings in your document.

The letter "z" represents the folding of a sheet of paper, just as you are folding a block of text. (I personally find my pinkie has a hard time reaching the `z` key, but I'm refraining from remapping everything.) You can think of the "z" as a preface to all fold-related commands.

Open and Fold

To navigate folded sections, you can use your standard navigation keys: `j` to move down a line; `k` to move up a line. When you locate a fold you want to open, use the key combination `zo` or press the space bar. To open all of the folds at once, use the key combination: `zR`. Once all of the folds have been opened, you can jump between headings using the key combinations `zj` and `zk`. To close a fold, use the key combination `zc` or press the space bar again. To close all folds, use the key combination `zM`. To reverse what's open and what's closed, use `zi`. I find this one particularly handy when switching frequently between everything open and everything closed.

Although I find "open" and "close" to be easy to remember, you may prefer using the key combination `za`. This command will do the opposite (or "alternate") of whatever the current state is, but only for the current fold. For convenience' sake, `za` is mapped to the space bar in my `.vimrc` file (and yours too if you've been following along). It will only toggle open/closed the current fold though. If you want to recursively close or open folds, you'll need to use `zc` and `zo`, respectively.

Navigating Open Folds

Assuming you're using the standard navigation keys, `j` and `k`, navigating open folds, is also a breeze. To move to the next heading, use the key combination `zj`; when traveling in the opposite direction, `zk` will take you to the end of the previous fold. Within your current fold, you can navigate to the end of the fold using `]z`, or to the beginning using `[z`.

Customizing the Folds

When I was first learning the folding commands, I found it much easier to have everything closed by default, so that I didn't forget to take advantage of folding. As I worked with files more, however, I found my preferences varied from day-to-day depending on the task I was working on.

For this article, it was best if level one headings were visible by default, but level three headings were invisible. To achieve this I tried customizing the fold start level in my `.vimrc` file (`set foldlevelstart=1`). If you're not sure what fold level is right for you, you can play around by setting `:set foldlevel=X` (where X is a number you'd like to try).

It wasn't quite what I wanted though. What I was looking for was the function `FoldToggle`. You can run the function from within Vim (`:FoldToggle`) or add the following to your `.vimrc` configuration: `let g:markdown_fold_style = 'nested'`.

In Closing

This is just starting to scratch the surface of what you can do with folding in Vim. If you didn't know about folding before, hopefully you'll be motivated to see how else (and for what other types of files) people are using this feature.