#29 Aligning text with Tabular.vim

Jan 17, 2011 Run time: 5:11

There are times when you can improve the readability of your code by lining up the elements on neighbouring lines. In this episode, I demonstrate how this can be achieved using the Tabular plugin.

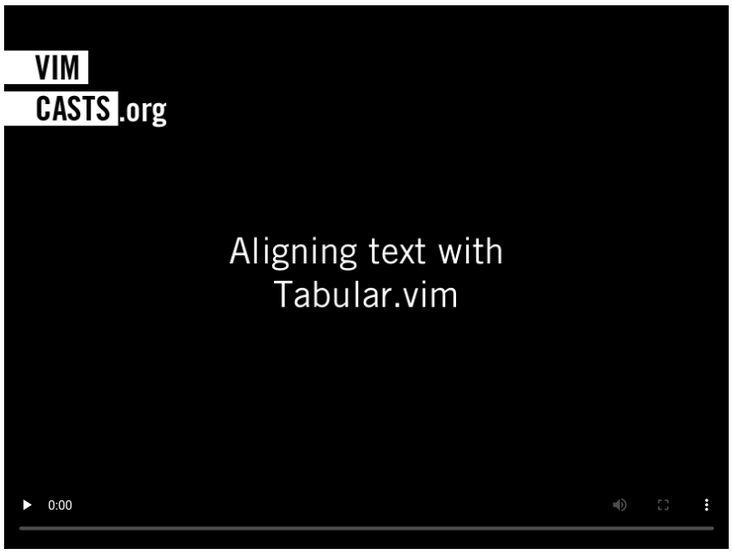
Download

OGG
6.3 MB

Quicktime
8.9 MB

Run time: 5:11

Read transcript



Shownotes

In this episode, I introduce the Tabular.vim plugin, by Matt Wozniski, which makes it easy to align regions of text that match a pattern.

Aligning assignments

Before:

```
one = 1
two = 2
three = 3
four = 4
```

Running **`:Tab /=`** produces:

```
one   = 1
two   = 2
three = 3
four  = 4
```

Colon assignments

There are a couple of different ways that colon assignments could be aligned. If we start with an example that is not aligned:

```
var video = {
    metadata: {
        title: "Aligning assignments"
        h264Src: "/media/alignment.mov",
        oggSrc: "/media/alignment.ogv"
        posterSrc: "/media/alignment.png"
        duration: 320,
    }
}
```

Select the inner block by positioning your cursor inside it and running **`vi}`** (enable *V*isual mode, and select *i*nner *B*race). Then you could run **`:Tab/:`** which would produce this result:

```
var video = {
    metadata: {
        title     : "Aligning assignments"
        h264Src   : "/media/alignment.mov",
        oggSrc    : "/media/alignment.ogv"
        posterSrc : "/media/alignment.png"
        duration  : 320,
    }
}
```

If you don't like stacking the colons in a column, you could use the **\zs** atom to exclude the **:** character from the search match. Running **:Tab /:\zs** produces this result:

```
var video = {
    metadata: {
        title:      "Aligning assignments"
        h264Src:    "/media/alignment.mov",
        oggSrc:     "/media/alignment.ogv"
        posterSrc:  "/media/alignment.png"
        duration:   320,
    }
}
```

Be aware that if you work in a team, there may be a house style that you should follow.

Table markup

Here is a scenario outline for cucumber steps, including a pipe-delimited table of examples:

```
Scenario Outline: eating
  Given there are &lt;start&gt; cucumbers
  When I eat &lt;eat&gt; cucumbers
  Then I should have &lt;left&gt; cucumbers

  Examples:
    |start|eat|left|
    |12|5|7|
    |20|5|15|
```

With the cursor positioned anywhere in the table, running **:Tab/|** produces:

```
Scenario Outline: eating
  Given there are &lt;start&gt; cucumbers
  When I eat &lt;eat&gt; cucumbers
  Then I should have &lt;left&gt; cucumbers

  Examples:
    | start | eat | left |
    | 12    | 5   | 7    |
    | 20    | 5   | 15   |
```

Creating mappings

If you find yourself using a particular token for alignment often, then you might want to save yourself a few keystrokes by creating mappings for normal and visual modes. Here are a few suggestions to get you started:

```
    let mapleader=','
    if exists(":Tabularize")
      nmap <Leader>a= :Tabularize /=<CR>
      vmap <Leader>a= :Tabularize /=<CR>
      nmap <Leader>a: :Tabularize /:\zs<CR>
      vmap <Leader>a: :Tabularize /:\zs<CR>
    endif
```

If you were in normal or visual mode, you could type **,a=** to align equals signs. In visual mode, the alignment would apply to the selected lines, but in normal mode tabular would attempt to guess the range.

You could take it a step further, by creating an insert mode mapping to trigger the **:Tabular** command when you type the character that you want to align. Tim Pope shows us how in this gist:

```
inoremap <silent> <Bar>   <Bar><Esc>:call <SID>align()<CR>a

function! s:align()
  let p = '^\s*|\s.*\s|\s*$'
  if exists(':Tabularize') && getline('.') =~# '^\s*|' && (getline(line('.')-1) =~
    let column = strlen(substitute(getline('.')[0:col('.')],'[^|]','','g'))
    let position = strlen(matchstr(getline('.')[0:col('.')],'.*|\s*\zs.*'))
    Tabularize/|/l1
    normal! 0
    call search(repeat('[^|]*|',column).'\s\{-\}'.repeat('.',position),'ce',line('
  endif
endfunction
```

If you put this in your vimrc file, then it will call the **:Tabularize** command each time you insert a **|** character.

Further reading

- the Tabular plugin
- :help /\zs - Vim's zero-width 'pattern start' search atom
- :help /\@<= - Vim also has a 'positive lookbehind' assertion
- Tim Pope's insert mode cucumber alignment gist
- Align.vim by Charles Campbell - another fine alignment plugin

BROWSE SIMILAR CONTENT

plugins

19 screencasts, 5 articles

Editing text

15 screencasts

Ex commands

8 screencasts

Whitespace

6 screencasts

LEVEL-UP YOUR VIM

Training

Boost your productivity with a Vim training class. Join a public class, or book a private session for your team.

> *Drew hosted a private Vim session for the shopify team that was one of the best workshops I have ever attended.*
>
> *John Duff, Director of Engineering at Shopify*

Publications

Make yourself a faster and more efficient developer with the help of these publications, including Practical Vim (Pragmatic Bookshelf 2012), which has over 50 five-star reviews on Amazon.

> *After reading it, I've switched to vim as my default editor on a daily basis with no regrets.* ★★★★★
>
> *Javier Collado*

Learn to use Vim efficiently in your Ruby projects

In association with thoughtbot, one of the most well respected Rails consultancies in the world, I've produced a series of screencasts on how to make navigating your Ruby projects with Vim ultra-efficient. Along the way, you'll also learn how to make Ruby blocks a first-class text object in Vim. This lets you edit Ruby code at a higher level of abstraction. Available to buy from thoughtbot..