Vim Asciidoctor 改进: 打造文档工具链

目录

1. 介绍

此插件是在 Vim-Asciidoctor 项目基础上 fork 而来，原版介绍看 README_Origin，本项目改进如下：

☑ 新增 asciidoc 向 markdown 转换方法 :Asciidoctor2MARKDOWN。

☑ 新增使用系统默认程序打开转换后的 markdown 方法 :AsciidoctorOpenMARKDOWN。

☑ 新增 UltiSnips 格式的 snippets，辅助进行快速插入各种版式。

☑ 搭配作者本人配置的 asciidoctor-pdf 主题+字体，可直接生成论文与书籍，开箱即用。

☐ 给 pygments 搞一套 asciidoc 语法高亮的配置文件。

2. 面向人群

- 日常工作有大量文本编辑与格式化输出需要的角色，比如产品经理、文档整理员、博客与书籍作者等。
- 掌握 vim/neovim 和终端命令行的基本使用。如进入退出 vim 和编辑文件、使用命令行复制粘贴、clone 代码库等基本操作。

3. 环境安装

3.1. 基础依赖

1. 安装各种包和依赖

```
git nodejs(version >= 12) ruby ruby-dev gcc make
neovim
```

2. 安装 neovim 的 python3 支持

```
$ python3 -m pip install neovim pynvim
```

3. 安装 asciidoctor 与 pdf/docx 转换器、语法高亮依赖、neovim 支持

```
$ gem install asciidoctor asciidoctor-pdf
asciidoctor-diagram pygments.rb neovim
```

3.2. 推荐：基于 SpaceVim

1. 安装 SpaceVim：

```
$ curl -sLf https://spacevim.org/cn/install.sh |
bash
```

2. kbd:[Space + f + v + d] 打开 init.toml 文件，启用以下配置：

```
[options]
autocomplete_method = "coc"
snippet_engine = "neosnippet"
[[layers]]
name = 'autocomplete'
auto_completion_return_key_behavior = "complete"
auto_completion_tab_key_behavior = "cycle"
auto-completion-delay = 100
auto-completion-enable-snippets-in-popup = true
[[custom_plugins]]
repo = "neoclide/coc.nvim"
merged = false
[[custom_plugins]]
repo = "DandiWong/vim-asciidoctor"
merged = false
```

3. 启动 nvim 后将自动安装

3.3. 不推荐：手动安装

- 安装 coc.nvim

```
$ mkdir -p ~/.local/share/nvim/site/pack/coc/start
&& cd ~/.local/share/nvim/site/pack/coc/start && git
clone --branch release
https://github.com/neoclide/coc.nvim.git --depth=1
```

- 安装本插件

```
$ mkdir -p
~/.local/share/nvim/site/pack/DandiWong/start && cd
~/.local/share/nvim/site/pack/DandiWong/start && git
clone https://github.com/DandiWong/vim-
asciidoctor.git --depth=1
```

4. 环境配置

4.1. asciidoctor-pdf 主题配置

完成环境安装部分后，在使用 Vim 编辑 asciidoc 文件时可以通过 :Asciidoctor2PDF/:Asciidoctor2DOCX/:Asciidoctor2MARKDOWN/:Asciidoctor2HTML 这些 vim-asciidoctor 插件构建的命令来调用 asciidoctor-pdf 完成格式的转换。但是默认的主题与字体配置对多语种混排、CJK 字体、Nerd Fonts 的支持都很差，所以作者也自定义了一套配置，你可以直接使用：

```
$ mkdir -p ~/.config && cd ~/.config/ && git clone
https://github.com/DandiWong/asciidoctor-pdf-config.git
```

4.2. neovim 配置

如果您使用的是 SpaceVim，那么配置文件为 ~/.SpaceVim/init.vim，如果您使用的是原生 neovim，那么配置文件为 ~/.config/nvim/init.vim，木有的话请自行创建。

init.vim

```
let g:username = 'Your name'
let g:email = 'Your email'
let g:current_time = strftime('%Y-%m-%d %H:%M:%S', localtime())

let g:python3_host_prog = '/usr/bin/python3'
let g:asciidoctor_executable = 'asciidoctor'
let g:asciidoctor_extensions = ['asciidoctor-diagram']
let g:asciidoctor_pdf_executable = 'asciidoctor-pdf'
let g:asciidoctor_pdf_extensions = ['asciidoctor-diagram']
" let g:asciidoctor_css_path = '~/docs/AsciiDocThemes'
let g:asciidoctor_pdf_themes_path = '~/.config/asciidoctor-pdf-config/themes'
let g:asciidoctor_pdf_fonts_path = '~/.config/asciidoctor-pdf-config/fonts'
let g:asciidoctor_pandoc_data_dir = '~/.config/asciidoctor-pdf-config/reference'
let g:asciidoctor_pandoc_executable = 'pandoc'
```

```
let g:asciidoctor_pandoc_other_params = '--toc --quiet'
let g:asciidoctor_fenced_languages = ['python', 'c', 'javascript', 'cpp', 'go', 'java', 'ruby', 'sh', 'typescript', 'markdown', 'html', 'css', 'rust', 'arduino', 'asciidoc']
let g:coc_global_extensions = ['coc-snippets']

" Use tab for trigger completion with characters ahead and navigate.
" NOTE: Use command ':verbose imap <tab>' to make sure tab is not mapped by
" other plugin before putting this into your config.
inoremap <silent><expr> <TAB>
      \ pumvisible() ? "\<C-n>" :
      \ <SID>check_back_space() ? "\<TAB>" :
      \ coc#refresh()
inoremap <expr><S-TAB> pumvisible() ? "\<C-p>" : "\<C-h>"

function! s:check_back_space() abort
  let col = col('.') - 1
  return !col || getline('.')[col - 1]  =~# '\s'
endfunction

" Use <c-space> to trigger completion.
if has('nvim')
  inoremap <silent><expr> <c-space> coc#refresh()
else
  inoremap <silent><expr> <c-@> coc#refresh()
endif

" Make <CR> auto-select the first completion item and notify coc.nvim to
```

```
" format on enter, <cr> could be remapped by other vim plugin
inoremap <silent><expr> <cr> pumvisible() ? coc#_select_confirm()
                              \: "\<C-g>u\<CR>\<c-r>=coc#on_enter()\<CR>"
```

4.3. coc.nvim 配置

安装 coc.nvim 后，启动 neovim 并在命令模式下 :CocConfig 打开配置文件，将下方配置项复制进去。

coc-settings.json

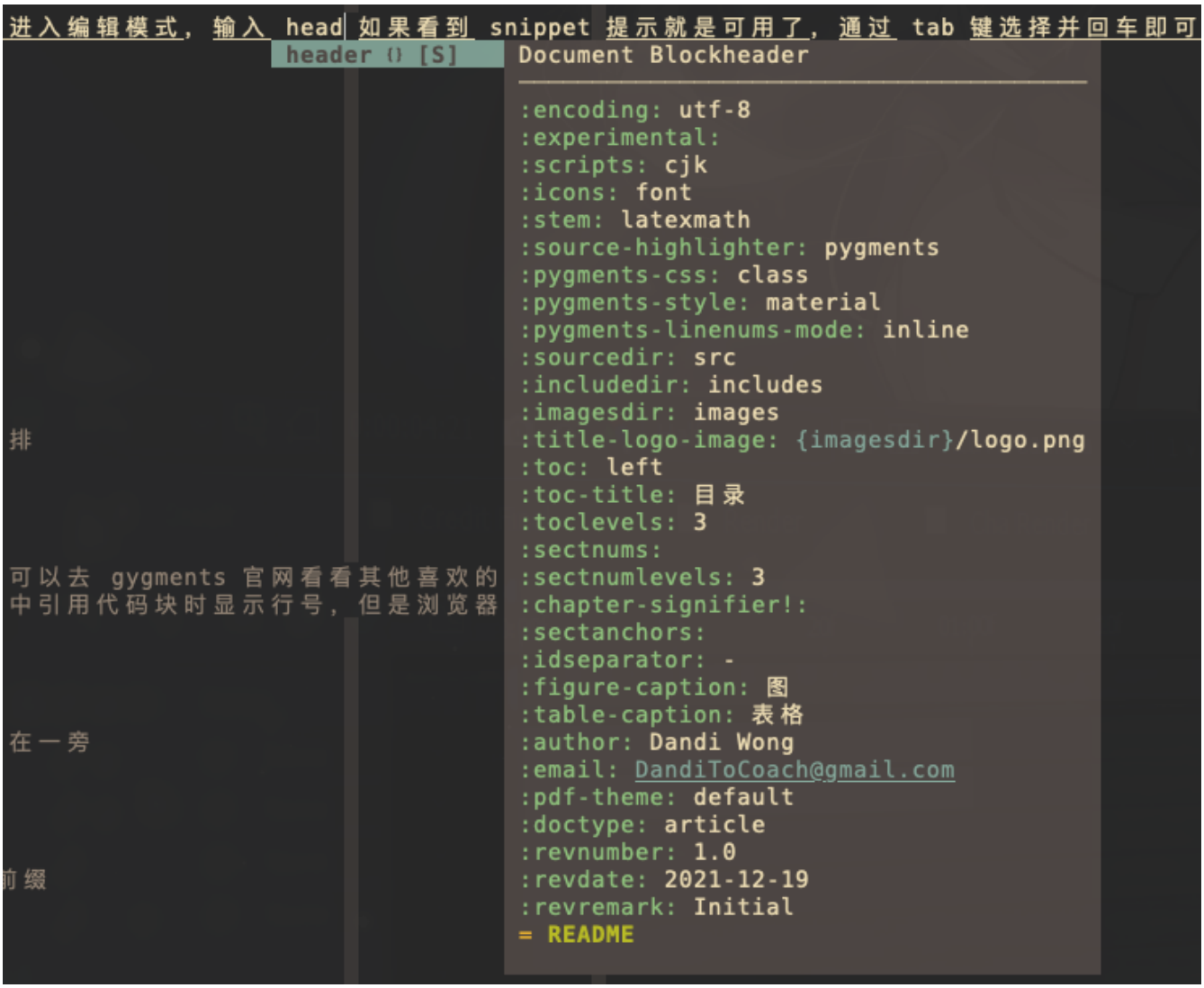
```
{
  "snippets.ultisnips.enable": true,
  "snippets.priority": 90,
  "suggest.snippetsSupport": true,
  "snippets.enableStatusItem": true,
  "snippets.autoTrigger": true,
  "snippets.loadFromExtensions": true,
  "snippets.trace": "error",
  "snippets.ultisnips.directories": [
      "UltiSnips"
  ]
}
```

5. 用起来吧！

5.1. 快速输入格式

nvim test.adoc 创建一个 asciidoc 文件进入编辑模式，输入 head 如果看到 snippet

提示就是可用了，通过 tab 键选择并回车即可，如下图:



6. snippets 介绍

6.1. 标准文档头

作用见注释

snippets-header

```
:encoding: utf-8 // 使用标准编码
:experimental:
:scripts: cjk // 支持中日韩文字与西文混排
:icons: font
```

```
:stem: latexmath
:source-highlighter: pygments
:pygments-css: class
:pygments-style: material // 高亮样式，可以去 gygments 官网看看其他喜欢的样式
:pygments-linenums-mode: inline // 文档中引用代码块时显示行号，但是浏览器预览时没展示，需要再调整
:sourcedir: src // 代码目录
:includedir:  // 引用文件目录
:imagesdir: images // 图片目录
:title-logo-image: logo.png
:toc: left  // 浏览器插件查看效果时目录在一旁
:toc-title: 目录
:toclevels: 3
:sectnums: // 启用章节编号
:sectnumlevels: 3
:chapter-signifier!: // 关闭章节自定义前缀
:sectanchors:
:idseparator: -
:figure-caption: 图 // 图片名称前缀
:table-caption: 表格 // 表格名称前缀
:author: `!v g:username` // 使用 init.vim 中自定义的作者名
:email: `!v g:email` // 同上
:pdf-theme: custom // 自定义主题
:doctype: book // book 比 article 多一些版式
:revnumber: 1.0
:revdate: `!v g:current_time` // 文档创建时间
:revremark: Initial
= ${1:`!v expand('%:r')`} // 使用去除路径和扩展的文件名作为标题
```

6.2. CSV 格式表格

表格 1. csv 表格展示

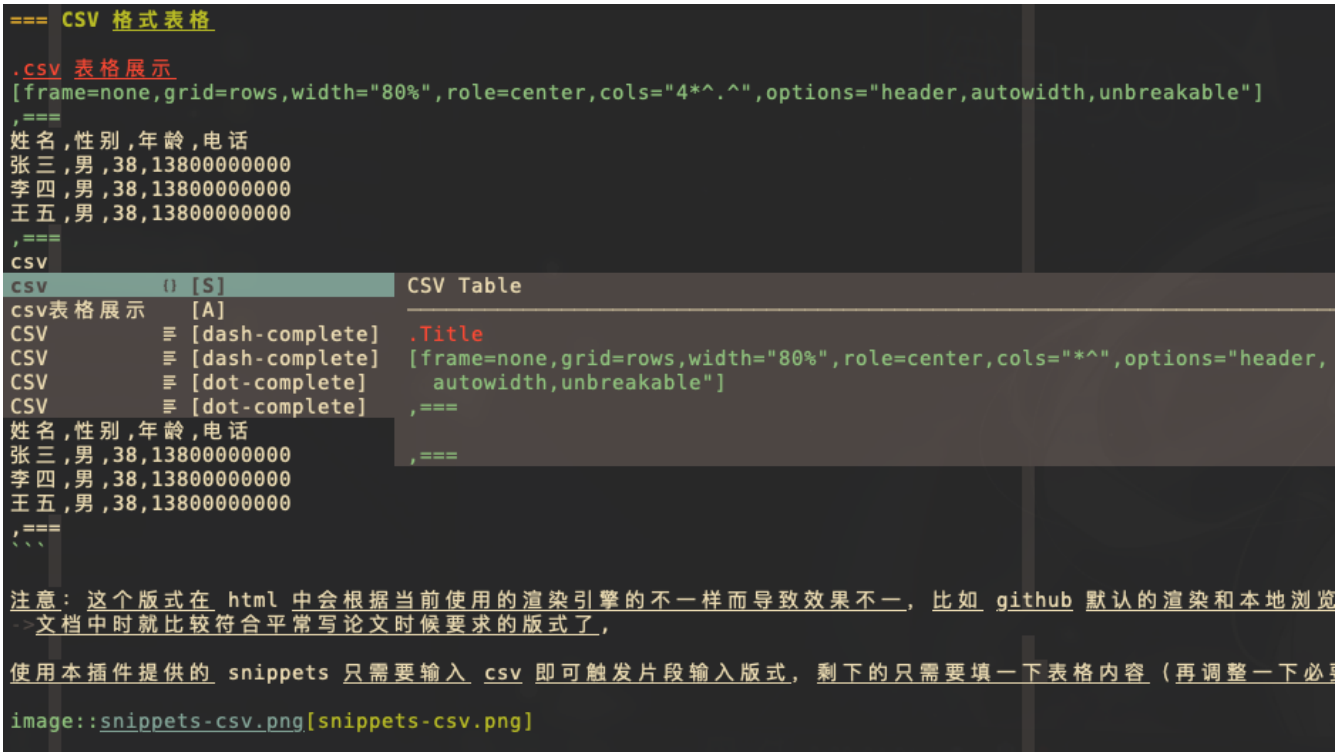
姓名	性别	年龄	电话
张三	男	38	13800000000
李四	男	38	13800000000
王五	男	38	13800000000

asciidoc 源格式为：

```
.csv表格展示
[frame=none,grid=rows,width="80%",role=center,cols="4*^.
^",options="header,autowidth,unbreakable"]
,===
姓名,性别,年龄,电话
张三,男,38,13800000000
李四,男,38,13800000000
王五,男,38,13800000000
,===
```

注意：这个版式在 html 中会根据当前使用的渲染引擎的不一样而导致效果不一，比如 github 默认的渲染和本地浏览器插件查看效果就是不一样的，而输出到 pdf 与 word 文档中时就比较符合平常写论文时候要求的版式了，

使用本插件提供的 snippets 只需要输入 csv 即可触发片段输入版式，剩下的只需要填一下表格内容（再调整一下必要的参数如 `4*^` 这里的数字应当为你表格的真实列数）就好了，很方便吧。



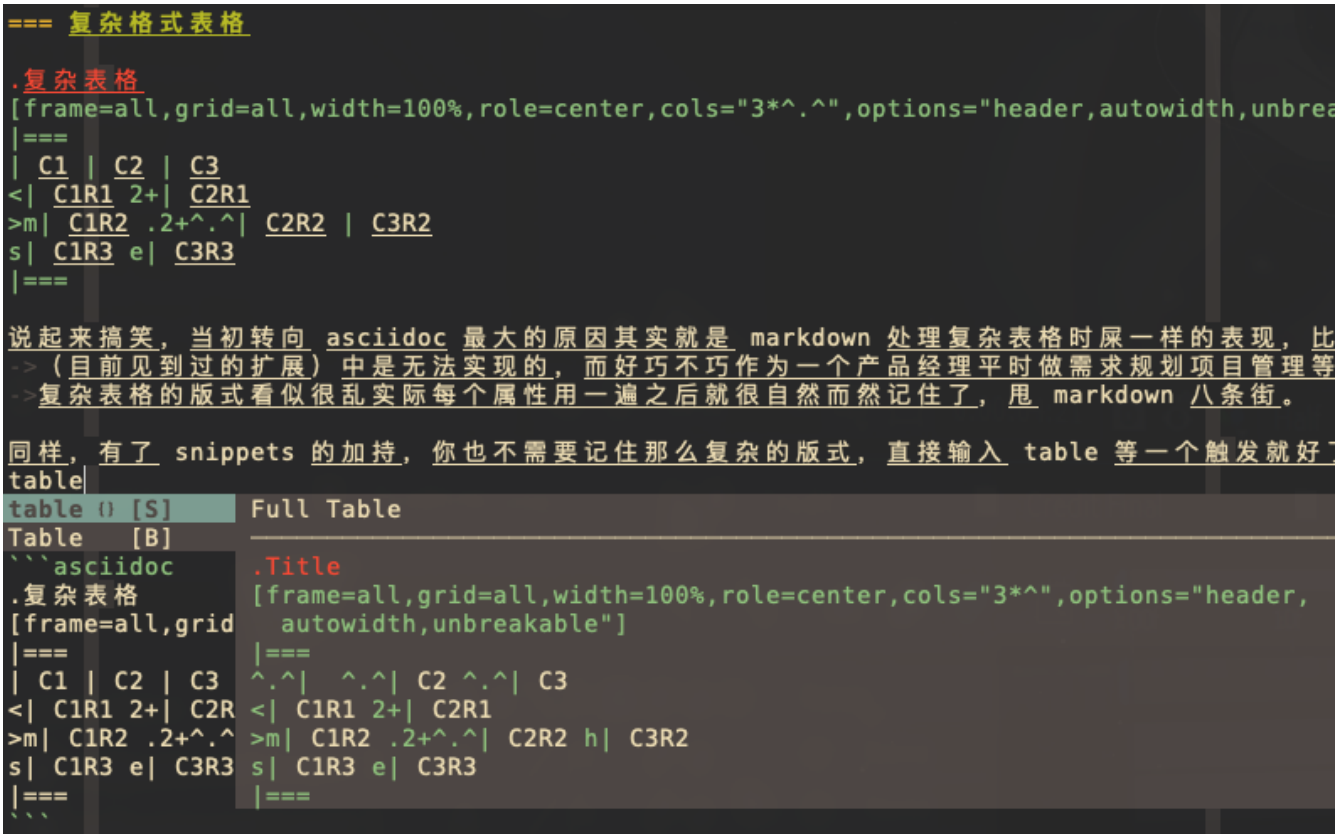
6.3. 复杂格式表格

表格 2. 复杂表格

C1	C2	C3
C1R1	C2R1	
C1R2	C2R2	C3R2
C1R3		C3R3

说起来搞笑，当初转向 asciidoc 最大的原因其实就是 markdown 处理复杂表格时屎一样的表现，比如上面这种输出样式在 markdown （目前见到过的扩展）中是无法实现的，而好巧不巧作为一个产品经理平时做需求规划项目管理等等时的文档就经常遇到各种合并单元格的情况，asciidoc 复杂表格的版式看似很乱实际每个属性用一遍之后就很自然而然记住了，甩 markdown 八条街。

同样，有了 snippets 的加持，你也不需要记住那么复杂的版式，直接输入 table 等一个触发就好了。




```
.复杂表格
[frame=all,grid=all,width=100%,role=center,cols="3*^.^",
options="header,autowidth,unbreakable"]
|===
| C1 | C2 | C3
<| C1R1 2+| C2R1
>m| C1R2 .2+^.^| C2R2 | C3R2
s| C1R3 e| C3R3
|===
```

6.4. 更多版式

参看作者自定义主题项目的效果 我的文档工具链: 基于 asciidoc 的文档编辑、排版与转换

7. 多种格式输出

asciidoctor 只是一个 asciidoc 的底层运行时，还需要搭配其他工具来进行格式化输出（如 word/pdf/markdown/html 等格式），在这里我们主要用到的是 asciidoctor-pdf 和 Chrome 浏览器插件 Asciidoctor.js Live Preview。

7.1. pdf 输出

在使用 neovim 编辑 asciidoc 文件时，在命令模式下使用 :Asciidoctor2PDF 即可在 asciidoc 文件同目录下生成同名 pdf 文档。根据 header 中的 :pdf-theme: 与 :doctype: 选项来决定版式。默认 :pdf-theme: custom 即作者提供的配置，:doctype: 为 book 时将使用书籍版式进行排版，而为空或者 article 时将使用常规版式进行排版。

7.2. word 输出

命令为 :Asciidoctor2DOCX，注意它的版式也是可以自定义的，版式文件从 asciidoctor-pdf-config/reference 中查找前缀为 :pdf-theme: 参数的文件，如 :pdf-theme: custom 则版式文件为 asciidoctor-pdf-config/reference/custom-reference.docx。

7.3. markdown 输出

命令为 :Asciidoctor2MARKDOWN，由于 asciidoc 支持的版式超过了 markdown，所以转换为 markdown 时会遇到一些版式丢失的情况。

7.4. 浏览器预览及 html 输出

安装 Chrome 插件 Asciidoctor.js Live Preview 并在插件选项中设置为可读取本地文件，将 asciidoc 文件拖入浏览器（Mac 下可以在命令模式中使用 :!Google-Chrome -a %）即可看到实时预览。

使用 :Asciidoctor2HTML 导出为 html 文件。

8. LICENSE

MIT License