

Set up the TikZ in Emacs Org

Eason

December 7, 2019

Contents

1	config the header args	1
2	generate results with different formats	2
3	export the results according to the backend	2
4	the final workflow	3

An introduction to setup the TikZ environment in Emacs Org File. So that in Org file, you can generate either vector graph of **pdf** format or raster graph of **png** format. Furthermore, you can export the vector graph when latex is called and otherwise raster graph.

1 config the header args

You can embed TiKz (One of \LaTeX graphic package) code into a Org file. With org-babel, you can generate ,insert and export the figure generated from Tikz package. At first, you need set up the environment. [This Post](#) serves as a good introduction for beginners. Following it you may have a minimum working example like below:

```
#+name: tutorial
#+header: :file "~/Dropbox/mstemc_hugo/static/img/tikz/tutorial.png"
#+header: :results raw :exports none :fit yes :border 0cm
#+header: :imagemagick t :iminoptions -density 400
#+header: :imoutoptions -geometry 400 -flatten
#+header: :headers '("\usepackage{tikz} \usetikzlibrary{positioning,
    shapes.symbols, calc}")
#+begin_src latex
  \begin{tikzpicture}
    \node [circle, draw, fill=red!20] at (0,0) {1}
    child { node [circle, draw, fill=blue!30] {2}
      child { node [circle, draw, fill=green!30] {3} }
      child { node [circle, draw, fill=yellow!30] {4} } };
  \end{tikzpicture}
#+end_src
#+RESULTS: tutorial
[[file:../../img/tikz/tutorial.png]]
```

The example begins with several lines containing **#+header** which is sort of clutter. we can put them in a file and include it at the beginning of the Org file.

2 generate results with different formats

By changing the extension of **:file** (which is **pdf** for `"../../img/tikz/tutorial.pdf"`), we can generate results with different formats. Now, I need **pdf** and **png**. You can see the result by just press **C-c C-c** in the body of the **tikz** code.

3 export the results according to the backend

You can set **:exports** to control how the results will be exported. Now I set it as **none** which means the result will not be exported to latex or other format. I want to set more options of the exports. So I use:

```
#+ATTR_HTML: :width 800 :align center
#+ATTR_LATEX: :width 0.8\textwidth :align center
{{{if-latex(tutorial.pdf,tutorial.png)}}}
```

if-latex is a Org MACRO whose definition is :

```
#+MACRO: if-latex (eval (if
(org-export-derived-backend-p org-export-current-backend 'latex)
  (concat "[[file:../../img/tikz/" $1 "]]")
  (concat "[[file:../../img/tikz/" $2 "]]") ))
```

The if-latex MACRO let you export different formats by the backend. If the backend is `latex` then, `pdf` format figure will be exported. Otherwise, `png` format figure. Eventually, in the final `pdf` document, you figure can be zoomed in or out without losing any resolution.

4 the final workflow

The minimum working example at the start of this post is simplified as below.

```
#+header: :file "../../img/tikz/tutorial.pdf"
#+begin_src latex
  \begin{tikzpicture}
    \node [circle, draw, fill=red!20] at (0,0) {1}
    child { node [circle, draw, fill=blue!30] {2}
      child { node [circle, draw, fill=green!30] {3} }
      child { node [circle, draw, fill=yellow!30] {4} }};
  \end{tikzpicture}
#+end_src

#+RESULTS:
[[file:../../img/tikz/tutorial.png]]
```

The following is the generated figure.

```
#+ATTR_HTML: :width 800 :align center
#+ATTR_LATEX: :width 0.8\textwidth :align center
{{{if-latex(tutorial.pdf,tutorial.png)}}}
```

Many settings are grouped into a setup file which is included at the top of this post:

```
#+SETUPFILE: ~/.spacemacs.d/org-templates/math-en.org
```

Now, If you set the extension of the target file in the first line either **pdf** or **png** , a corresponding **pdf** or **png** figure will be generated. If execute **M-x org-toggle-inline-images** , you can preview the result. If export the org-file as latex file then the **pdf** file, the image of **pdf** format will be inserted. If export to other format, **png** image will be used.