

# make STEAM clear

Eason

December 6, 2019

## Contents

<b>1</b>	<b>Videos</b>	<b>1</b>
<b>2</b>	<b>Posts</b>	<b>1</b>
2.1	Set up the TikZ in Emacs Org File . . . . .	1
2.1.1	config the header args . . . . .	1
2.1.2	generate results with different formats . . . . .	2
2.1.3	export the results according to the backend . . . . .	2
2.1.4	the final workflow . . . . .	3
<b>3</b>	<b>Projects</b>	<b>3</b>
3.1	my workflow of creating a video . . . . .	3
3.1.1	Emacs . . . . .	4
3.1.2	Hugo . . . . .	4
3.1.3	manim . . . . .	5
3.1.4	blender . . . . .	5
<b>4</b>	<b>Courses</b>	<b>5</b>

## 1 Videos

## 2 Posts

### 2.1 Set up the TikZ in Emacs Org File

Why is this in green?

#### 2.1.1 config the header args

You can embed TiKz (One of  $\text{\LaTeX}$  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 :imoutoptions -geometry 400 -flatten
#+header: :headers '("\usepackage{tikz} \usetikzlibrary{positioning, shapes.symbols, c
#+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.

### 2.1.2 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.

### 2.1.3 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.

### 2.1.4 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.

## 3 Projects

### 3.1 my workflow of creating a video

During the creation of one video, several tools are tuned to fit my workflow. In this project several videos and posts will be created to make my workflow clear.

Two targets of this project:

1. remind me of the configuration of several tools;
2. show somebody else who may be interested in how I work.

I am not ready to detail each tool. If you are interested in any one of them, please refer to tutorials written by experts or given by the official document.

### 3.1.1 Emacs

Emacs lies in the center of my workflow. It behaves like a super-IDE which assists me finish almost all the steps. Using Emacs, I program and debug, GTD, write the blogs, export and publish them. Many of the Emacs extensions are awesome. In particular, Emacs Org is the killer app.

How to config Emacs? After more Than ten years of using Emacs, I choose spacemacs. Most beginners can start their work using spacemacs with only a little modifications.

1. program and debug

I program in python, C/C++, Matlab and some other languages. Emacs acts as a super-IDE for me.

2. Org mode - journalize your work

Org mode definitely need a standalone subsection.

3. Org mode - get things done

4. Org mode - write a diary

5. Org mode - export your Org file

6. program and debug

7. writing latex

8. org-journal

I write many notes during a post or a project. Most of my journal is maintained by [org-journal: A simple org-mode based journaling mode](#) . And I set the `org-journal-file-type` to `weekly` So that I have around 52 journal file each year.

The journals can be searched conveniently. Also they can be integrated into Org agenda.

9. ox-hugo

[kaushalmodi/ox-hugo: A carefully crafted Org exporter back-end for Hugo](#)

Because hugo supports markdown at first (it also supports org file later), I use Emacs Org from the very beginning. I am quite familiar with org-mode, so have no motivation to dig into markdown. Fortunately, ox-hugo meets all my requirement and definitely worth a try when you want to stay in Org.

Actually, Emacs have extensions to support markdown quite well. However, using Org, I can integrate the org file into my agenda.

10. preview tikz picture in Org

[TikZ and PGF](#) are  $\text{\TeX}$  packages for creating graphics programmatically. TikZ is build on top of PGF and allows you to create sophisticated graphics in a rather intuitive and easy manner.

You can integrated TikZ code into the Org mode as a source block. I have a not-so-detailed post on how to set up the tikz environment in Org.

### 3.1.2 Hugo

As the fastest framework for building websites, Hugo shocks me by its speed and flexibility. It is enough to prettify your site using the more than 300+ beautiful themes,

from which I am in the mode for Academic theme.

Hugo supports Org file but not so good. Ox-hugo, another extension of Emacs, has my attention. Serving as a bridge between Emacs and Hugo, ox-hugo helps me staying in Emacs Org. There is no need for me to write markdown file if Org is available.

I use ox-hugo as a bridge between Emacs and Hugo. So I do not have to care much about how to leverage Hugo. Ox-hugo helps me do most of the work.

### **3.1.3 manim**

### **3.1.4 blender**

## **4 Courses**