Detailed description of my tex4ht setup

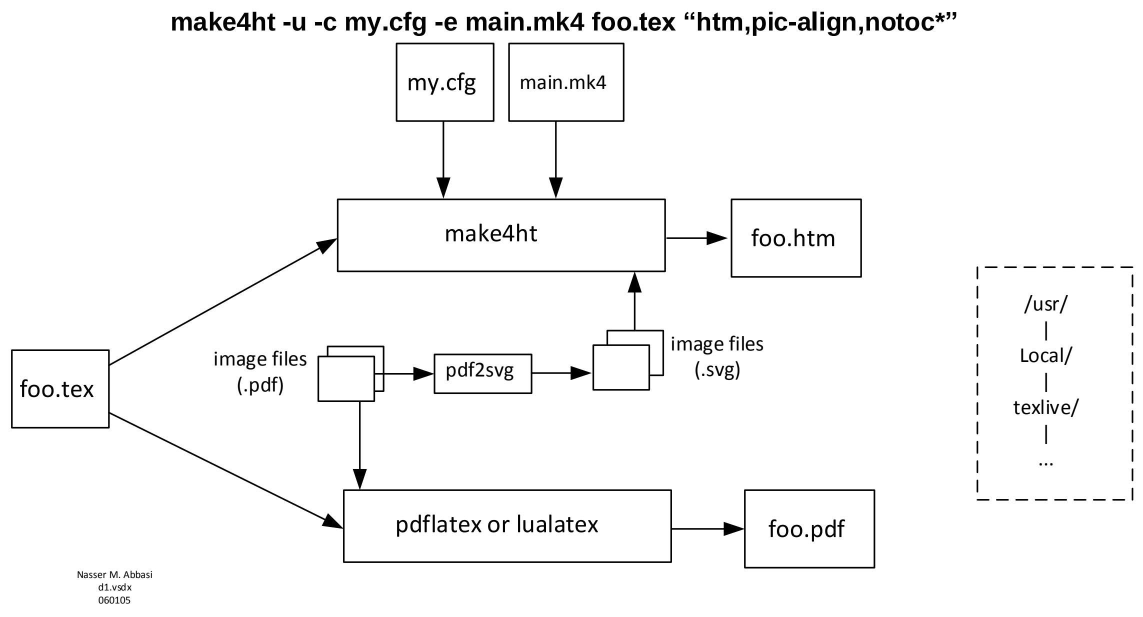
Nasser M. Abbasi

March 29, 2022 Compiled on March 29, 2022 at 5:33pm [public]

Contents

1 Method that uses SVG for images and does not require eps to obtain image size

The general parts of the setup are illustrated in the following diagram, then a detailed description is given for each part



```
make4ht -u -c my.cfg -e main.mk4 foo.tex "htm,pic-align,notoc*"
```

Figure 1: Overall view of the setup

The above diagram shows the main files used. There are basically two important files: The file `my.cfg` and `main.mk4`. Since make4ht is used to build latex to html, the original tex4ht file `tex4ht.env` is no longer used. This is a good thing, as `tex4ht.env` is very hard to manage.

The file `my.cfg` in the diagram above is the configuration file for tex4ht. The file `main.mk4` is used to customize the make4ht build process.

The setup described here uses `.svg` for both math and external images included using `includegraphics` when building the html pages, and uses `.pdf` for external images included using `includegraphics` when building to pdf file with pdflatex of lualatex.

Hence an initial step is to crop and then convert each pdf image file to svg file. This is done for each image file that is to be included using `includegraphics`

A small script is used to do this conversion. Here is the script called `prep`. This scipt will crop the pdf file (remove white spaces around the image) and then convert it to svg

```
#!/bin/bash
set -x
for file in $1; do
    filename=${file%.*}
    pdfcrop --margins 10 --clip "$filename.pdf" "$filename.pdf"
    pdf2svg "$filename.pdf" "$filename.svg"
done
```

I also use scour which is a python script to reduce the size of the svg images. Which does a very good image in reducing the size of SVG file. This is not needed, but if you need to use it, this is how the extended `prep` script above would be now

```
#!/bin/bash
set -x
for file in $1; do
    filename=${file%.*}
    pdfcrop --margins 10 --clip "$filename.pdf" "$filename.pdf"
    pdf2svg "$filename.pdf" "$filename.svg"
    scour -i "$filename.svg" -o x000.svg
    mv x000.svg "$filename.svg"
done
```

I had to make a temporary file `x000.svg` in the above, since `scour` does not allow one to overwrite the same file. To use `scour`, these are the commands to install it on Linux

```
sudo apt-get install python-pip
sudo pip install scour
```

Script `prep` is used as follows. To convert one pdf image file say `image.pdf` to `image.svg"`, the command is `prep image.pdf`

To convert all pdf image files in one folder to svg image files type `prep "*.pdf"`

After this prepration step, the latex file is compiled to html using the command

```
make4ht --lua -u -c my.cfg -e main.mk4 foo.tex "htm,pic-align,notoc*"
```

make4ht contains documentation on all the options it supports in addition to the ones shown above. For example to send all the output which includes the html file generated and all the images to a separate folder, the following command is used

```
make4ht --lua -u -c my.cfg -e main.mk4 foo.tex "htm,pic-align,notoc*" -d my_output_folder
```

A template latex file is given below. The `\includegraphics` command below does not include the image file extension. This is taken care of by the use of the configuration file described below. When compiling using `pdflatex` the pdf image file is read and when compiling using tex4ht, the svg file is read automatically.

```
\documentclass[11pt]{scrartcl}

\usepackage{graphicx}
\begin{document}
%....
\includegraphics[width=.8\textwidth]{image1}
%.....
\end{document}
```

The `\includegraphics` command must use a size specification in this setup, such as `width=.8\textwidth` or `width=.8\linewidth` or another exact value. It can not be without some size specification such as using scale or just `\includegraphics[]{image1}` as this will not work.

This is to remove the need to generate `.xbb` file for each svg. The setup here uses the `width=scale\textwidth` for all images.

1.1 Template tex4ht configuration file

The tex4ht configuration file `my.cfg` is described below. This file tells texh4t to use SVG instead of png for images, as svg is vector graphics and it scales better at different resolution. This file can also be used to customize the HTML output more by adding direct `CSS` command in the section starting after `\begin{document}` in this configuration file. There is a `\begin{document}` in the configuration file, which is not to be confused with the `\begin{document}` in the latex file itself. There are not the same.

Here is the template for the file `my.cfg` used in the above command. (Thanks to Post by Michal Hoftich on Tex stackexchange

```
\Preamble{ext=htm,charset="utf-8",p-width,pic-align}
\DeclareGraphicsExtensions{.svg,.png,.jpg}
\Configure{Picture}{.svg}

\Configure{VERSION}{}
\Configure{DOCTYPE}{\HCode{<!DOCTYPE html>\Hnewline}}
\Configure{HTML}{\HCode{<html>\Hnewline}}{\HCode{\Hnewline</html>}}
\Configure{@HEAD}{}
\Configure{@HEAD}{\HCode{<meta charset="utf-8"/>\Hnewline}}

\Configure{@HEAD}{\HCode{<meta name="generator" content="TeX4ht
(http://www.cse.ohio-state.edu/\string~gurari/TeX4ht/)" />\Hnewline}}
\Configure{@HEAD}{\HCode{<link rel="stylesheet" type="text/css"
```

```
             href="\expandafter\csname aa:CssFile\endcsname" />\Hnewline}}

\DeclareGraphicsExtensions{.svg,.png}

\Configure{graphics*}
        {png}
        {%the special command below so it will copy the png over
           \special{t4ht+@File: \csname Gin@base\endcsname.png}
           \Picture[pict]{\csname Gin@base\endcsname .png
              \space width="\expandafter\the\csname Gin@req@width\endcsname"
}%
}%

%need this below for MATH.
\Configure{Picture}{.svg}
%this below to make it resize the SVG image, if it is there, to
%what is in the includegraphics. but I am going back to .png for
%images (not math)thanks to @Michael.h21 for this trick
\makeatletter
\newcommand\emwidth{10}
\newcommand\CalcRem[1]{\strip@pt\dimexpr(#1)/\emwidth}
\Configure{graphics*}
           {svg}
            {\Picture[pict]{\csname Gin@base\endcsname.svg
            \space style="width:\CalcRem{\Gin@req@width}em;"
            }%
            \special{t4ht+@File: \csname Gin@base\endcsname.svg}
           }
\makeatother

%see http://tex.stackexchange.com/questions/308309/problem-breaking-url-when-using-tex4ht-only
%by michal.h21

\begin{document}
%add any custome \Css or HTML commands here as needed.

%These to tell tex4ht to make inline math images also
\Configure{$}{\PicMath}{\EndPicMath}{}
\Configure{PicMath}{}{}{}{class="math";align="absmiddle"}

\EndPreamble
```

1.2 Description of make4ht configuration file main.mk4

The above section described one of the main files used which was `my.cfg`. This section describes the other main file `main.mk4` which is used to customize the make4ht build. Below is the listing of the current `main.mk4` used in this setup.

This file comes with the make4ht installation. The first line was modified to make the math little larger. The command

```
dvisvgm -n -c 1.15,1.15 -p 1- ${input}.idv
```

Tells dvisvgm to make the math equations 1.15 larger than the default. This value can be adjusted as needed by editing this file `main.mk4`

The above file is located in the `$HOME` folder, as well as the file `my.cfg`. These files are kept in `$HOME` to make it easier to locate by make4ht. But these can also be in the same folder where the latex file is located.

1.3 Post TexLive configuration

As of Texlive 2015, make4ht is now part of TexLive. There is nothing to do to install it.

The following are the relevant environment variables from the `.bashrc` file used in this setup. Change path to your texlive as needed.

```
export PATH=/usr/local/bin:/usr/local/texlive/2016/bin/i386-linux:$PATH
export TEXINPUTS=.:$HOME/
export TEXMFHOME=$HOME/texmf
export MANPATH=/usr/local/texlive/2016/texmf-dist/doc/man:$MANPATH
export INFOPATH=/usr/local/texlive/2016/texmf-dist/doc/info:$INFOPATH
```

And of course texlive itself has to be installed.

1.4 Using makefile to build latex to HTML and PDF

The setup uses Make to build all latex files. It uses a recursive make that builds the whole tree. But here a small example of Make is given which builds only one directory.

It builds both pdf and HTML from the latex and converts all the pdf image files to svg if needed automatically. This means when the pdf image file is changed (may be due to updating the drawing itself, or running tikz again to generate new pdf image file), then the images that have changed are automatically converted to svg when running make and the pdf and HTML are build. This way the result always contains the correct latest images.

This is illustrated below using a template layout, which consists of one folder that contains the latex file, and a subfolder which contains all the images that are used by the latex file. In this example there are 2 image files. The images are kept in separate folder to make it easier to manage.

The output of the HTML is send to separate folder. Therefore the structure before the make is run is as follows:

```
source/
|-- images
|   |-- d1.pdf
|   |-- d2.pdf
|
|-- index.tex
|-- main.mk4
|-- Makefile
|-- my.cfg
```

After running make, the structure becomes like this

```
source/
|-- images
|   |-- d1.pdf
|   |-- d1.svg
|   |-- d2.pdf
|   |-- d2.svg
|
|-- index.pdf
|-- index.tex
|-- main.mk4
|-- Makefile
|-- my.cfg
|-- output
    |-- images
    |   |-- d1.svg
    |   |-- d2.svg
    |
    |-- index.css
    |-- index.htm
```

Notice that in the output, all the .svg images are copied over as well as the HTML file to the output directory.

The example latex file is.

```
\documentclass[11pt]{scrartcl}
\usepackage{amsmath}
\usepackage{graphicx}

\begin{document}
This is an example Latex file. First image is now loaded

\includegraphics[width=.8\textwidth]{images/d1}

And now the second image is loaded

\includegraphics[width=1\textwidth]{images/d2}

\end{document}
```

The makefile is

```
all: index.htm

IMAGES=$(wildcard images/*.svg)

index.htm: index.tex $(IMAGES)
        pdflatex index.tex
        pdflatex index.tex
        pdflatex index.tex
        make4ht -u -c my.cfg -e main.mk4 index.tex "htm,pic-align,notoc*" -d output
```

```
$(IMAGES): %.svg: %.pdf
        prep $@

.PHONY: clean
clean ::
        -rm -f index.htm
```

From the `source` folder, issuing the following commands will build the index.tex file

```
make clean
make all
```

The script `prep` used in the makefile is shown earlier above.