Converting LaTeX notes to HTML

Written by Ian Price <ian.price2@abdn.ac.uk>

The purpose of this document is to explain how to convert a set of lecture notes, or similar LaTeX document, into HTML using the converter LaTeXML. This is important for accessibility, since PDF is not handled well by screen-reader programs, whereas lots of work has been put into html and web accessibility. It will also cover a number of the issues that can arise when converting and how to overcome them.

In this document all filenames and commands will be marked in a monospace font, e.g., `ca1.tex`, and all commands will be on a separate line.

Installation

I will not provide detailed instructions on installation, except to point out that LaTeXML is available in many package managers for Linux, Mac OS, or through Perl's CPAN. You can find out more on the official installation page, or find more detailed instructions for Windows on StackExchange.

From now on, I will assume LaTeXML is already installed onto your system and you have some basic command line familiarity on your system.

Running LaTeXML

The Basics

We will assume that there is a file `notes.tex` and that any file that includes, e.g. images, are in the same folder. We will also assume that the current directory of the terminal has been switched to that folder with `cd` or similar command. Throughout this example, I will be using the notes for "MA1005: Calculus 1" which produces 100 pages of PDF output, and ran the example on a somewhat old 2017 AMD® A6-7310 processor with 3GB of RAM.

Turning a TeX file into a HTML file with LaTeXML is a two stage process: TeX files are processed into an internal XML representation, and then into HTML for output[1]. You should not need to edit the xml directly, although this may be necessary in extreme cases.

[1] It is possible to convert directly from LaTeX to HTML using the command `latexmlc`, but we will not do this here. One reason it is useful to have the XML around is to allow you to play around with the options to `latexmlpost` without having to rerun the lengthy `latexml` command.

The first step is handled by the program `latexml`, which we can run to produce the xml file

```
latexml notes.tex --dest=notes.xml
```

This command produces a lot of textual output, and may take some time if your document is large or uses a lot of macros or packages, e.g. processing the MA1005 notes takes 20 minutes. This will create a new file in the same folder with the name `notes.xml`.

Next we produce the HTML file with the command

```
latexmlpost notes.xml --dest=notes-folder/notes.html --split
--splitat=section --navigationtoc=context
--javascript="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-chtml
.js?config=MML_HTMLorMML" --urlstyle=file --timestamp=0
```

This is a single command that should be typed in one line, and will run much faster than the previous one. The full options will be explained later, but suffice to say that we are instructing `latexmlpost` to create a new folder called `notes-folder` which will contain the file `notes.html` and any files it depends on. We are also instructing it to include MathJax for rendering mathematics in a screen reader friendly way. The entire contents of `notes-folder` should be accessible when you make this available on the internet[2].

The Details

While the above instructions should be enough for basic usage, you may find you need to customise certain behaviours. There is no substitute for reading the documentation, but we will summarise some of the most common ones below.

Latexml

- `--dest=notes.xml`
  Like most Unix programs, `latexml` defaults to printing out the document on “standard output” (that is, to the terminal) rather than to a file. With this option, it outputs the xml to the file `notes.xml`
- `--includestyles`
  This allows latexml to load raw `.sty` files, which can be useful for loading custom class files.

Latexmlpost

- `--dest=notes.html`
  As above, this tells `latexmlpost` to output the html to the file `notes.html`. Any additional files necessary for rendering the HTML will be put in the same folder, so it is

[2] In the simplest case, deployment of these files is just copy/pasting to an accessible location on a web server, but for more detailed deployment you will need to talk to your system administrator.

often sensible to specify a folder as well as filename, e.g., `--dest=notes-folder/notes.html`

- `--split`
  This tells latexmlpost to split the output into several HTML pages, as opposed to one document. For shorter documents this is unnecessary, but desirable for large documents to cut down html loading times.
- `--splitat=subsection`
  Used in combination with the above option, this tells `latexmlpost` which level of LaTeX heading it splits into a new page, e.g. `section` or `subsection`.
- `--navigationtoc=context`
  This generates a table of contents in a navigation bar on each page.
- `--javascript=js-file`
  This includes a link to the javascript file `js-file`, which may be a local file or a link to another web page[3]. We are primarily using it to include MathJax for rendering mathematics.
- `--css=css-file`
  As above, but for CSS.
- `--urlstyle=file`
  This determines the style of link used by LaTeXML.
- `--timestamp=tstamp`
  Includes a specific timestamp for when the file is generated in the comments of the HTML file. By default `latexmlpost` uses current date and time, or you can use 0 to omit the timestamp.

General Advice

Before going on to explain how to fix common issues that occur during the conversion process, here is a short list of things that might make your life easier.

[3] This is not the time and place for a Javascript/CSS Tutorial, but to get an example of what can be done, you should look at andy-navbar.js and andy-navbar.css in this tutorial.

Firstly, I recommend that if you have a large document that you split off different sections, say a week's worth of notes, into separate files and take advantage of the `/input` and `/include` LaTeX commands. Aside from being good practice, it makes it easier to selectively include / exclude pieces of the notes while you debug any formatting issues and will have faster compile times.

If you want things to be done a certain way depending on whether generating PDF or HTML, you can take advantage of the command `\iflatexml \else \fi`.

You should try to be somewhat conservative when using packages. Ask yourself if you really need to bring in an additional library just for one teeny improvement. LaTeXML does not support every LaTeX package, and there is no (and can't be) a comprehensive list of those supported vs not. This includes some core packages, like mdframed, which is used for coloured boxes, and imakeidx for creating indexes. Also LaTeXML points out that some low-level TeX primitives are not mimicked faithfully. We will have more to say about this in the next section.

Since the purpose of this document is to create accessible documents, you should ensure that you are doing other necessary steps, e.g., checking if your images are colour-blind friendly[4], and providing accurate and useful alt-text for your images using the `\description` command[5].

Similarly, to take full advantage of HTML's ability to reflow text, you should make limited use of commands that specify precise spacing. Also some commands like `\hfill` no longer make sense in a web context where there is no inherent notion of page width.

Finally, we have limited our discussion to notes and other documents that use standard classes like `article`. While I'm not saying it won't work for other classes, one major sticking point is Beamer, a package used for creating presentations. At present, there is no good solution for converting beamer to HTML[6]. If you need to create HTML accessible presentations, you might consider something like Pandoc to create HTML presentations, but an alternative is just to provide a second document containing, or summarising, the same information as the presentation.

Formatting Issues

Assuming there were no significant errors in the `latexml` or `latexmlpost` phases, you should have got one or more HTML files. However, unless you were quite conservative in which LaTeX

---

[4] This is easier said than done, but a summary of the issues and some suggestions can be found at https://knightlab.northwestern.edu/2016/07/18/three-tools-to-help-you-make-colorblind-friendly-graphics/.

[5] The command `\Description[<short description>]{<long description>}` can, and should, be used inside every figure and table. It differs from `\caption` in that it is meant to be used instead of the image, rather than in combination with it.

[6] There has been recent work in Tex4ht, another LaTeX to HTML converter, trying to get beamer supported again, but at the time of writing I cannot recommend it.

packages and features you used, there are likely to be some formatting issues to correct. It is essentially impossible to be comprehensive here, but here are some I have found and how I resolved them.
I'd also like to point you to another tutorial which has found other bugs, but I will not repeat them here.

[NOFIX] First row of table bolded in tabular

```
\begin{examplesol}\hfill
\begin{center}
\begin{tabular}{llll} (a) $3$\hspace{20 pt}    &    (b)
1\hspace{20 pt}    &  (c) The limit does not exist.\hspace{20 pt}
&    (d) $g(2)$ is not defined
    \\  \\ (e) 2  &  (f) 2 &   (g) 2  &  (h) 1
\end{tabular}
```

In this example, `examplesol` is a type of theorem environment used for solutions to example exercises, and appear with the word "Solution" in bold. However, in this example, the first line of the table is also in bold. It is not clear why `examplesol` has this issue and other `newtheorem*` environments do not.

[NOFIX] Align environment not handling empty lines

`\align*` environments are often used to provide sequences of equalities alongside a "running commentary" explaining the proof. While this does work correctly, when a line has only the final comment (see below), it is not being aligned correctly.

```
\begin{align*}
&& \text{\small{\color{red} \emph{now} subtract}}
\\
&&&\text{\small{\color{red} finally, don't expand the bottom}}
\end{align*}
```

This is likely a bug in LaTeXML, and I was unable to find a workaround, however in many cases it will be straightforward to append the final comment to the previous line, or add it after the `align*` environment.

Colour leaking to other parts of document

One issue in the calculus was the command `\book` as defined as follows

```
\newcommand{\book}[1]{\marginpar{\tiny\color{brown} Stewart #1}}
```

This caused everything in the document after the first instance of `\book` to be tiny and brown, indicating that these commands were not constrained to the `\marginpar` environment. This seems to be a bug, but is easily fixed by using a second pair of braces to delimit the effect, e.g.,

`\newcommand{\book}[1]{\marginpar{{\tiny\color{brown} Stewart #1}}}`
It is likely that any other “state leakage” can be contained in the same way.

I do not recommend the use of \marginpar in HTML anyway because it produces somewhat ugly output, and the text is only available when you hover over the cross (†) that indicates it.

Set Builder Notation

It is common to typeset sets using so-called “set builder notation”, e.g., the set of all reals satisfying a predicate P might be typeset as

`\left\{x \in \mathbb{R} \ \middle|\ P(x)\right\}`.

However the use of `\middle` was not handled correctly. Fortunately, in this case you can replace it with `\mid` instead giving

`\left\{x \in \mathbb{R} \mid P(x)\right\}`.

Bracketing issues

Not all formatting issues are, in fact, the fault of LaTeXML, but may indicate mistakes in your code, e.g., the following code has an additional right bracket

`\left[\ln(\cos(x))]\right]`.

With PDF output the brackets looked all the same, but with LaTeXML the outermost bracket was of a larger size, and only then did I notice the brackets were unbalanced.

[NOFIX] () in blank sections

When a `\section` command is followed by no text before the first `\subsection`, it is being rendered as a pair of brackets. I currently know of no workaround. This does not seem to exist in the xml file, so is being produced by `latexmlpost` and is likely a bug.

Tikz

The MA1005 notes make heavy use of Tikz, a popular library for creating vector graphics within LaTeX. While many diagrams were produced correctly, due to Tikz’s size and complexity, it ended up being responsible for the lion’s share of the issues in the formatting of the notes. Unfortunately, this was not limited to producing incorrect diagrams, but also sometimes caused bits of the document to be missing, or out of place. In general, if you are using Tikz and a formatting issue arises near a Tikz image, try commenting it out and see if the image was causing it.

While I will list the fixes I know for some of the issues I found, I recommend converting your image descriptions into image files, e.g. in SVG format, and including them as you would a regular image. A brief tutorial for this is provided at the end of this section.

Multiple Images side by side

A single math mode expression can contain multiple tikzpicture environments side by side, e.g., `\begin{tikzpicture} … \end{tikzpicture} … \begin{tikzpicture} …`
This gets displayed as an error message containing “Math Input Error”. To fix this you can take advantage of the subcaption package to create a figure containing several subfigures and have a single tikz expression in each subfigure. An example can be found on stackexchange.

Converting TikZ Code to Images

If you are having significant issues with TikZ, you might consider converting all your TikZ images to standalone images. This can be accomplished by creating standalone pdfs with your image[7] and then converting with a variety of programs including ImageMagick, pdf2cairo, pdf2image, pdf2svg, or some web converter (you’ll find tons if you google).

For example[8], if you have the following latex in `example.tex`

```
\documentclass[tikz, border=1mm]{standalone}
\begin{document}
\begin{tikzpicture}
\draw (0,0) node [] {My text};
\end{tikzpicture}
\end{document}
```

Compiling the latex with `pdflatex example.tex` will produce the file `example.pdf` which you can convert to, e.g., SVG, with `pdf2svg example.pdf example.svg`. Finally, you would include the image in your LaTeX code as normal with `\includegraphics{example.svg}`.

[7] This is strictly speaking unnecessary since you can include PDFs as images, but it is better practice since we want the browser to render it as an image rather than including a PDF.
[8] This example was taken from https://techoverflow.net/2019/11/02/how-to-export-tikz-graphics-as-svg/