Multiple output formats from the same LaTeX source*

Gareth Walker

February 19, 2025

Contents

1 Introduction

There are times when it can be useful to use the same LaTeX source code and produce multiple output formats. One such case is where you are preparing materials for students and you want them to have, for example, a PDF 'reference' version but would also like them to have the same materials in HTML format and in formats they can use in LibreOffice (ODT) and in Microsoft Word (DOCX). This document gives you some tips on how to achieve that. If you are currently looking at an output file in PDF, HTML, ODT, or DOCX format **these all originated from the same single LaTeX source file**. So as well as documenting the process of conversion, this file functions as an example of the output which can be produced.

*The information in this document is offered in good faith and I hope that others find it helpful. Much of it has come, directly or indirectly, from the author of the make4ht tool, Michal Hoftich. There are absolutely no guarantees as to whether, and how well, these tools will work in your case. I am not the author of any of the tools referred to in this document. Corrections and improvements to this document are welcome.

2 The main tool: make4ht

The main tool in the conversion process described here is Michal Hoftich's excellent make4ht tool (and the related tex4ht package). So, install those things if you don't already have them. There is quite extensive documentation if you need it, but this document should help you get started.

2.1 Getting HTML output

This is as simple as doing this from the command line:

```
make4ht multiple-formats.tex
```

2.2 Getting LibreOffice (ODT) output

You can do this from the command line:

```
make4ht -f odt multiple-formats.tex
```

2.3 Getting Microsoft Word (DOCX) output

Once you have the ODT file, you can open the file in LibreOffice and use "Save as..." to save the file in DOCX format. If you are able to call LibreOffice from the command line (terminal) on your system, you may be able to do the conversion that way, something like this:

```
libreoffice24.2 --convert-to docx multiple-formats.odt
```

2.4 Putting it all together

Depending on your system, you may be able to put all these steps together in a shell script so that running that shell script on a file generates some or all of the output formats (PDF, HTML, ODT, DOCX). On my system (Ubuntu Linux) I can have a bash script like this:

```
#!/bin/bash
fileName=$(basename "$1" | cut -d. -f1)
latexmk $fileName
make4ht $fileName
make4ht -f odt $fileName
libreoffice7.5 --convert-to docx $fileName.odt
```

If I save this as `makeMultipleFormats`, make that file executable, and possibly put it somewhere in my system's path, I can then run this command:

```
makeMultipleFormats multiple-formats.tex
```

That single command gets me PDF, HTML, ODT and DOCX versions of the same LaTeX source, all at the same time. Nice!

3 Other things

This section contains some suggestions on how to deal with some of the situations which might crop up, as well as showing you some of the things which can be converted.

3.1 Tables

Tables are converted.

a	b	c
1	2	3
2	3	4

Table 1: A table

3.2 Graphics

You can include graphics like this, optionally specifying a size and alt-text to describe the image:

```
\includegraphics[alt={a colour picture of a tiger},width=3cm]{tiger}
```

The alt-text will be preserved in the HTML and ODT versions, but seems to be lost when converting to DOCX from LibreOffice.

To get the image size you want in the ODT and DOCX file, you may need to do a bit of work. One thing you can do is use the `ebb` command from the command line to create a special file which contains the dimensions of the graphics, e.g.

```
ebb -x tiger.pdf
```

(This works on other types of graphics as well, not just PDF.) There is more detail on this on StackExchange.

3.3 Math

Numbered equations such as

```
\begin{equation}
4x^2+\frac{3}{2}x-15=0
\end{equation}
```

can be converted to graphics:

$$4x^2 + \frac{3}{2}x - 15 = 0 \tag{1}$$

The graphics will have alt-text generated automatically.

Inline math (e.g. `$sin^2x+cos^2x\equiv1$`) can appear in the text: $sin^2x + cos^2x \equiv 1$. There are options to convert math into other accessible formats, including MathML and MathJax: see the documentation.

3.4 Phonetic symbols

Phonetic symbols seem to convert fine when done something like this:

```
\documentclass{article}
\usepackage{fontspec}
\newfontfamily\ipafont{Doulos SIL}
\newcommand\ipa[1]{{\ipafont #1}}
\begin{document}
This is a test
\ipa{[(input unicode IPA here)]}
This is a test
\end{document}
```

Since you need XeLaTeX or LuaTeX to use the `fontspec` package, to do the conversion to conversion you need e.g.

```
make4ht -l file.tex
```

and for ODT output e.g.

```
make4ht -l -f odt file.tex
```

A DOCX file can be created from the ODT file using LibreOffice in the same way as described in section 2.3.

3.5 Syntax trees

This is a document showing a syntax tree using the qtree package, from the qtree documentation:

```
\documentclass{article}
\usepackage{qtree}
\usepackage{ulem}
\begin{document}
This is a tree.

\Tree
[.IP [ Roses ].NP_i [.I\1 [ are ].I\0
[.VP t_i [ [ going ].V\0 \qroof{out of style}.PP ].V\1 ].VP
].I\1 ]
\end{document}
```

The conversion to HTML seems robust, e.g.

```
make4ht tree.tex "svg"
```

The "svg" switch to get graphics in the SVG format seems to be required in order to preserve the lines in the graphic representing branches.

To create a ODT or DOCX version, you can open the HTML file in LibreOffice, then save it in those other formats. The size of the graphics in the final files may need to be adjusted.

3.6 Heading levels

You can change the style formatting to start at e.g. "Heading 1" by using a configuration file like this:

```
\Preamble{xhtml}
\Configure{Heading-2}{Heading 1}
\Configure{Heading-3}{Heading 2}
\Configure{Heading-4}{Heading 3}
\begin{document}
\EndPreamble
```

If you save the configuration file as `heading.cfg` you can then do e.g.

```
make4ht -f odt -c heading.cfg file.tex
```

3.7 Creating ZIP files of HTML versions

There is some information about this on StackExchange.

3.8 Putting HTML versions of documents on Blackboard

Make a ZIP file of all the files you need to make available (e.g. the HTML file, CSS file, any graphics). Upload your ZIP files to the content collection as a package, so that it unzips once uploaded. You then have to give all the files the correct permissions. In the content collection, find the folder you want to make available. Click on the 'permissions' icon for that folder. That will take you to a list of who can access the contents of the folder, and what they can do with the (e.g. read, write, modify, manage). Click "Select Specific Users By Place", then "Course". Choose the course (module) taken by the students to whom you want to give access, check the box(es), scroll to the very bottom to "select roles" and select e.g. "Student" or "All course users", check the permissions you will give (probably just "Read"), then click "Submit".