What's new in TeX4ht: 2022

Michal Hoftich

Abstract

This article provides an overview of the recent development of TeX4ht, LaTeX to XML converter, and make4ht, the build system that carries out this conversion.

1 Introduction

Richard Koch wrote an article on interactive documents produced using TeX4ht in this issue of *TUGboat*. He and Karl Berry asked if I would be able to provide additional tips on the usage of TeX4ht and also to summarize recent changes in the system.

You can find the basic summary of the basic features of TeX4ht in my previous article [1]. I will focus on new features and changes in this article.

2 Changes and new features in make4ht

There are some substantial changes in the make4ht build system. These are the most important:

2.1 Terminal output

Originally, make4ht showed the full terminal output of TeX and all the commands it called during the conversion process. This resulted in a huge amount of information printed on the terminal. It also used the default behavior of LaTeX, so the compilation was stopped for every error, waiting for the user action.

The new default behavior is to run the compilation in `\nonstopmode`, with most terminal output suppressed. Only errors and warnings are shown.

You can change the output method using a new command line option `--loglevel`, or `-a` in the short form. Each log level prints messages of the current level and all higher levels. It supports the following levels:

error print only error messages.

warning show make4ht warnings, for example, from HTML postprocessing filters.

status this is the default level.

info print all make4ht messages, but suppress the output from commands.

debug this level is the original default, printing all output from TeX and all other executed programs, and it also stops on compilation errors.

2.2 Input redirection

make4ht now supports shell input redirection, which means that it can process the output of other commands without the need to use temporary files. You need to pass `-` as the filename, and also set the output filename using the `--jobname` or `-j` option:

```
$ python generatetex.py | make4ht -j foo -
```

2.3 Conversion of additional markup languages

In addition to LaTeX and plain TeX, make4ht supports some additional markup languages, thanks to the preprocess_input extension. It detects the markup used using the file extension, so it is necessary to name the file accordingly. It preprocesses the input using Pandoc or R with the Knitr library, which needs to be installed on your system.

Here's the list of supported file extensions:

.rtex	LaTeX with R code chunks
.rnw	LaTeX with Sweave code chunks
.rmd	RMarkdown
.rrst	R + reStructuredText
.md	Markdown
.rst	reStructuredText

For example, the following LaTeX document contains R commands, so we name it (say) `x.rtex`:

```
\documentclass{article}
\begin{document}
You can have R commands in your \LaTeX{}
document. They will be processed and
their output will be typeset:
<<>>=
# Create a sequence of numbers
X = 2:10
# Summary of basic statistical measures
summary(X)
@
\end{document}
```

You can compile it with the following command, which loads the preprocess_input extension:

```
$ make4ht -f html5+preprocess_input x.rtex
```

2.4 New commands available in build files

With make4ht, you can use Lua build files to call additional commands, such as indexing and bibliography processors. Built-in commands are provided for Biber, BibTeX, Makeindex, Xindy, Xindex and PythonTeX. They take care of the special settings necessary to work correctly with TeX4ht.

As an example, the following document produces an index with links that point to the places where `\index` is used:

```
\documentclass{article}
\usepackage{makeidx}
\makeindex
\begin{document}
Hello\index{hello} world\index{world}
\printindex
\end{document}
```

doi.org/10.47397/tb/43-3/tb135hoftich-make4ht

A build file, say `build.lua`, that uses Makeindex as an indexing processor could look like this:

```
Make:htlatex {}
Make:makeindex {}
Make:htlatex{}
```

The command `Make:htlatex` compiles the document using LaTeX with the TeX4ht package automatically loaded, `Make:makeindex` calls Makeindex and the final `Make:htlatex` compiles the document with the index included. Note that instead of page numbers, the numbers in the index are numbered consecutively for each `\index` command. Due to that, we can point every index entry back to the original location.

To use a build file, use the `-e` command line option:

```
$ make4ht -e build.lua foo.tex
```

3 Documentation and server side compilation

We have made progress in writing new TeX4ht documentation. It contains chapters on available configuration commands and command line options, and also a how-to guide with common tasks. Developer information for package writers is also included. It is available here:

`www.kodymirus.cz/tex4ht-doc`

It also describes an important development, the usage of server-side compilation. Thanks to Github Actions, documentation is automatically generated from LaTeX sources every time we update them. We don't need to upload generated HTML files to a web server, everything is handled automatically by Github Actions. In the background, the Docker container for TeX Live is used. It enables us to call any command available in TeX Live, including `make4ht`. A similar service is also provided by GitLab and other source code hosting platforms.

This method has also been used for the conversion of Overleaf projects linked to Github repositories, the HTML version of `make4ht` documentation, and even a simple blog:

`www.kodymirus.cz/testblog/`

4 JATS format support

We recently added support for the JATS XML format, which is intended for scientific article authoring. This is an important development, as this format is required by many publishers for article archiving or further processing.

It is also the first output format for TeX4ht that I personally created. The specification is quite strict on the structure of the document, which is often inconsistent with the free document structure used in LaTeX. `make4ht` postprocessing using LuaXML is heavily used to produce the correct structure.

The support is still fairly basic, so user feedback and bug reports are appreciated. The basic invocation:

```
$ make4ht -f jats foo.tex
```

5 MathJax configuration

We continue to extend support for MathJax, which can be used to render math in converted documents. The resulting document typically looks much better than documents converted using the default TeX4ht method, which uses a mix of images and HTML formatting. It is also better for accessibility, as MathJax can support screen readers, for example.

One pitfall is that MathJax does not support custom commands out of the box. It needs to be given special configuration that declares these commands. Here is an example of such a configuration file for a hypothetical macro `\foo`:

```
\Preamble{xhtml,mathjax}
\Configure{MathJaxConfig}{{
tex: {
 \detokenize{%
  macros: {
    foo: "\\mathrm{foo}",
  }
 }
}
}}
\begin{document}
\EndPreamble
```

The `\Configure{MathJaxConfig}` command here is given JavaScript code that configures MathJax. The `macros` table, which needs to be located inside the table `tex`, can contain user macros. The `\detokenize` (ε-TeX) command is used to prevent problems with backslash characters, which need to be doubled. In the example, we define the `\foo` macro, which prints the word "foo" in roman font.

References

[1] M. Hoftich. TeX4ht: LaTeX to Web publishing. *TUGboat* 40(1):76–81, 2019. `tug.org/TUGboat/tb40-1/tb124hoftich-make4ht.pdf`

◇ Michal Hoftich
`michal dot h21 (at) gmail dot com`
`https://tug.org/tex4ht/`