LaTeX to Web publishing

Michal Hoftich
May 3, 2019

Charles University, Faculty of Education, Library
Popular \TeX{} to HTML convertors

- tex4ht
- Pandoc
- LaTeXML
- LaTeX2HTML
- Lwarp
tex4ht overview

- https://www.tug.org/tex4ht/
- created in the mid nineties
- original author Eitan Gurari (1947–2009)
- current team Michal Hoftich and Karl Berry
- updates goes directly to TeX Live
Basic features

- it uses $\TeX$ for the compilation of the document (all formats in theory, mainly $\LaTeX$ in practice)
- it can convert parts of the document to images
- it supports multiple output formats
Overview of the compilation

LaTeX \rightarrow x.dvi \rightarrow \text{tex4ht} \rightarrow x.idv

html files

images & CSS files
Compilation by \TeX with the \texttt{tex4ht.sty} package auto-loaded

- loading configuration \texttt{.4ht} files for the supported packages
- patching commands with hooks
- hooks configuration according to the output format
DVI processing using the \texttt{tex4ht} command

- generation of the output files
- font handling
  - based on the \texttt{.htf} files, they contain mappings between the font characters and Unicode
  - information about the font style
  - character encoding conversion
- prepare \texttt{.lg} and \texttt{.idv} files
.lg file processing using the t4ht command

- CSS file
- picture generation
- external commands calling (xslt, tidy, xmllint, xtpipes)
How does `tex4ht.sty` work?

- `tex4ht.sty` package is called before the document is loaded by TeX.
- It modifies the document processing.
- It detects all used packages and loads configuration `.4ht` files at the `\begin{document}`.
- The configurable hooks are inserted into redefined commands.
- Another `.4ht` file with tags is included after the package configurations. It contains all configurations for the current output format.
How does \texttt{tex4ht.sty} work?

- \texttt{tex4ht.sty} package is called before the document is loaded by \TeX{}
- it modifies the document processing
- it detects all used packages and loads configuration \texttt{.4ht} files at the \texttt{\begin{document}}
  the configurable hooks are inserted into redefined commands
- another \texttt{.4ht} file with tags is included after the package configurations. It contains all configurations for the current output format

Alert!
The commands used in the document preamble are not patched by \texttt{tex4ht} by default
When is it necessary to insert the configurable hooks?

- when we want to keep the logical structure of the document (sectioning, tables, lists, etc.)
- in the case of a clash between existing tex4ht commands and commands provided by a package
How are hooks configured

- hooks are configured using the `\Configure` command
- either in the output format file (html4.4ht, html5.4ht, ooffice.4ht)
- or in the private configuration file
- the output format can define options that are passed to the `tex4ht.sty` package
they are used by the output format for a conditional configuration

- they can be passed on the command line
- or in a private configuration file

Example

$ make4ht filename.tex "mathml,mathjax"
Some available options

- fn-in
- pic-m, pic-align
- svg
- info
- mathml
- mathjax
Private configuration file

- Basic structure

  \% \texttt{\textbackslash RequirePackage is possible to use here}  
  \texttt{\textbackslash Preamble\{xhtml, options\}}  
  \texttt{\textbackslash Configure\{foo\}{}{}}  
  \texttt{\textbackslash Css\{body{...}\}}  

  ...  

  \texttt{\textbackslash begin\{document\}}  
  ...  

  \texttt{\textbackslash EndPreamble}
Some available commands

- \Configure, \ConfigureEnv, \ConfigureList
- \HCode, \Css, \Hnewline
- \EndP, \IgnorePar
- \Picture+, \Picture*
- \NoFonts, \EndNoFonts
Example for the `\Configure` command

Example

`\Configure{textit}
    {\HCode{<em>}{\NoFonts}
     {\EndNoFonts}{\HCode{</em>}}
    }`
Issues with paragraphs

Example

\ConfigureEnv{rightaligned}
   \HCode{<section class="right">}
   \HCode{</section>}}{}

The generated HTML code is invalid

Example

<p class="indent" >&lt;section class="right"&gt;
A correct solution

Example

\ConfigureEnv{rightaligned}
  {{\ifvmode\IgnorePar\fi\EndP%
  \HCode{<section class="right">}\par
  {{\ifvmode\IgnorePar\fi\EndP%
  \HCode{</section>}}}}}}

Result

<section class="right">
<!--l. 9--><p class="indent" >
</p><!--l. 9-->
Example

\ConfigureEnv{topicture}
  {\Picture*{}}
  {\EndPicture}
}{}}
Complete configuration file

\Preamble{xhtml}
\Configure{textit}
  {\HCode{<em>}}\NoFonts
  {\EndNoFonts\HCode{</em>}}
\ConfigureEnv{rightaligned}
  {\ifvmode\IgnorePar\fi\EndP
    \HCode{<section class="right">}\par
  }\ifvmode\IgnorePar\fi\EndP
  \HCode{</section>}}{}{}
\ConfigureEnv{topicture}
  {\Picture*{}{}}{\EndPicture}{}}{}
\Css{.right{text-align:right;display:block;}}
\begin{document}
\EndPreamble
How to add support for a new package?

• create file named *package name + .4ht*, redefine commands and insert configurable hooks here
• configure the hooks in the output format *.4ht* file
Sample package

custom.sty:
\ProvidesPackage{custom}
\newcommand\custom[1]{\bgroup\itshape#1\egroup}
\endinput

custom.4ht:
\NewConfigure{custom}{2}
\pend:defI\custom{\a:custom}
\append:defI\custom{\b:custom}
\Hinput{custom}
Configure the hooks for the HTML output

\Configure{custom}
{\HCode{<span class="custom">}}\NoFonts
{\EndNoFonts\HCode{</span>}}
• high number of compilation scripts
• the basic script was **htlatex**
• the difference between the scripts is just in used options
• superseded by **make4ht**
Traditional compilation scripts

- bash scripts for UNIX, batch scripts for Windows
- parameters can be passed for each command used in the tex4ht compilation
htlatex issues

- difficult way of passing the arguments to `htlatex`
- fixed compilation sequence
  - `\TeX` is always executed three times
  - it is not possible to use Bib\TeX{} or similar tools
- it is hard to modify the image conversion process
- copying of files to an output directory doesn’t work correctly
- post-processing of the generated files
• support for e-books
• written in Lua
• simplified interface, use of command line switches
• Lua build file support
  • call external commands
  • picture generation process simplified
  • post-processing of the generated files
• extensions
• it keeps the correct directory structure with the \texttt{--output-dir} option
• evolved from tex4ebook
• supports all tex4ht output formats
htlatex versus make4ht

$ htlatex filename.tex \\ "tex4ht.sty options" "tex4ht options" \\ "t4ht options" "TeX options"

versus

$ make4ht [make4ht switches] filename.tex \\ "tex4ht.sty options" "tex4ht options" \\ "t4ht options" "TeX options"
How to get the UTF-8 encoded document?

$ \texttt{htlatex filename.tex "xhtml,charset=utf-8"} \\
\hspace{1em} \texttt{-cmozhtf -utf8} \\

versus

$ \texttt{make4ht -u filename.tex}$
make4ht switches

--utf8 (-u)
--mode (-m)
--lua (-l)
--config (-c)
--build-file (-e)
--output-dir (-d)
--shell-escape (-s)
--xetex (-x)
--format (-f)
Supported formats

make4ht

- html5
- xhtml
- odt
- TEI
- DocBook
- etc.

tex4ebook

- ePub
- ePub3
- mobi
Example

$ make4ht -f html5+tidy simple-example.tex
Available extensions

latexmk_build

tidy
dvisvgm_hashes

common_filters and common_domfilters

mathjaxnode – example https://www.kodymirus.cz/samples/mathjaxnode/maths.html

staticsite
Example

\documentclass{article}
\begin{document}
Test {\itshape háčků}
\end{document}
Example

<!--l. 4--><p class="noindent">Test <span class="rm-lmri-10">h</span><span class="rm-lmri-10">á</span><span class="rm-lmri-10">čk</span><span class="rm-lmri-10">ů</span> </p>
Example

```lua
local domfilter = require("make4ht-domfilter")
local function domsample(dom)
  for _, par in ipairs(dom:query_selector("p")) do
    par:set_attribute("class", "mypar")
  end
  return dom
end
local process = domfilter({
  "joincharacters",
  domsample
})
Make:match("html$", process)
```
Example

<!-- l. 3 -->
<p class='mypar'>
Test <span class='rm-lmri-10'>háčků</span></p>
Build file with external commands

sample.mk4

Make:add("biber","biber ${input}" )
Make:htlatex {}
Make:biber {}
Make:htlatex {}
Make:image("png$",
"dvipng -bg Transparent -T tight -o ${output}" ..
"-pp ${page} ${source}" )

$ make4ht -e sample.mk4 filename.tex
That’s all

Thanks for your attention.

Questions?