Accessible LaTeX Mathematics

Don Massie*, Andrey Sarantsev†

June 26, 2019

Abstract

In this document, we explain how to make math content produced from a LaTeX document accessible for visually impaired students. We outline some possible solutions, in particular converion of LaTeX documents to an accessible HTML-MathJax format.

1 Preliminaries

For the purposes of this discussion, a document is considered *accessible* if it satisfies two requirements: (a) it can adequately be read by assistive technologies, such as screen readers, used by visually impaired people; (b) it is *tagged:* all headings, tables, lists, pictures, graphs, and other special objects must be identified as such, and all pictures or graphs must have captions or alternative text.

Accessibility can be measured by automated tools which rate the degree of tagging and other usability supports. One of them is Ally, pre-installed in Canvas, which is the University of Nevada, Reno (UNR) learning management system. Another is WAVE, available as an extension of browsers. The most important way to test, however, is to use an actual screen reader: JAWS (commercial), NVDA (free open-source), or others.

There are tools to make Word, PDF, or PowerPoint documents accessible. For example, in (commerical) Adobe Acrobat Pro there are tools called `Accessibility` and the `Action Wizard`, which facilitate making many types of content accessible. These tools are not in the free Adobe Acrobat Reader DC, but they are freely available for UNR faculty.

However, mathematics (and other disciplines that use many formulas) presents special problems. Indeed, most math content is created in LaTeX. This is a publishing system based on TeX markup language, which is used for typesetting mathematics formulas. It is widely used in Mathematics, Computer Science, Statistics, Physics, Chemistry, Biology, Economics, Finance, and other disciplines with complicated formulas. LaTeX is not WYSIWYG (what you see is what you get): insteadm one types the text with commands (all starting from backslash \) which generate math symbols and formatting, in a special document `.tex`, which is then compiled. The output is PDF.

*Teaching and Learning Technologies, donmassie@unr.edu

†Department of Mathematics and Statistics, asarantsev@unr.edu

TEX language was created by Donald Knuth in 1978 and is still very widely used. Mathematicians and other aforementioned disciplines rarely use Word, OpenOffice, or other WYSIWYG editors, because LATEX has much higher quality of typesetting.

LATEX is a free, open-source software that works equally well on common operating systems. It is available on multiple web sites, including `https://www.latex-project.org/`. One can also use the online compilator Overleaf to convert from `.tex` to `.pdf`. There are other publishing systems based on TEX, but LATEX is used overwhelmingly. In this review, we shall use TEX and LATEX interchangeably. Below is an example of a very short LATEX file, let us call it `test.tex`:

LATEX Code

```
\documentclass{article}
\usepackage{amsfonts}
\begin{document}
Hello, world!
$x + y = \alpha$
$$(x-y)\cdot (x+y) = x^2 - y^2$$
\end{document}
```

Output

Hello, world!
$x + y = \alpha$

$$(x - y) \cdot (x + y) = x^2 - y^2$$

The output is in `test.pdf`. The dollar signs create a *math mode* between them. In it, one can use math commands, such as the one which produces the greek letter α. The text of `.tex` document outside of the math mode is called the *text mode.* The `usepackage` command adds special packages which are not part of the core LATEX but are often useful; the `amsfonts` package adds special fonts created by the AMS (American Mathematical Society). These two web sites: CTAN and TUG, are repositories of TEX and LATEX online. They serve as reference places for the TEX global community.

2 Main Problem

This language cannot, however, create an accessible PDF. The PDF output is not tagged, and therefore not accessible. To learn more about this, search for the question: *How can I create math and science documents that are accessible to students with visual impairments?* on the web page `https://www.washington.edu/doit/` (University of Washington, Seattle).

Our goal is to find a way to make mathematics typed in TEX, as a PDF or otherwise, accessible. Our target audience is an average professor, researcher, or graduate student.

These people are familiar with their research area, with at least basics of scientific publishing, and have a basic knowledge of LaTeX. Many in our target audience know basic HTML. This is the markup language used by all browsers to create Web pages; see more in Section 3.2. We assume knowledge of HTML is good enough to write simple personal pages. We do not assume knowledge of CSS (Cascading Style Sheets), a more sophisticated technique on top of HTML which enables different *styles.*

Such users often do not know *command line* interface `cmd`. They do not want to learn a new language (programming or markup), or a new piece of software. In general, they are busy with research and teaching, and therefore do not wish to invest a lot of time and effort.

It is surprisingly hard to find a way to make formulae, equations, and special characters accessible. It seems that in the TeX community, this is still an active research and development topic. The technology is relatively unfinished, and not standardized. One needs to look on online forums, most importantly `TeX Stack Exchange` (a Q&A service developed specifically for TeX users), and `Stack Overflow` (a general Q&A forum for information technology and computer science).

We found that there is no universally good solution for an accessible output from LaTeX. However, in Section 3 we describe in detail one method which we found the best for making mathematical content work with screen readers. In Section 4, we describe issues arising from such conversion, and whether they require manual intervention. Finally, in Section 6, we mention several potential paths which we tested and did not find satisfactory. However, they might be useful for more sophisticated users, or for subsequent research on this topic.

3 Our Solution: LaTeX to HTML-MathJax

In this section, we focus on the process that seems the most promising: converting LaTeX documents to HTML (Hyper Text Markup Language) format, which can be read by browsers such as Chrome or Firefox. There is a MathJax plugin for HTML, developed by AMS (American Mathematical Society), in part to make mathematics accessible. MathJax can read LaTeX commands and formulas in 'math mode' as if it were LaTeX itself. In the rest of Section 3, we describe in more detail the HTML format and MathJax plugin. We choose the Pandoc online converter for its simplicity. Then we describe which features of LaTeX can and which cannot be translated to the HTML-MathJax.

3.1 Industry motivation

An additional motivation for converting LaTeX to HTML-MathJax arises from reviewing documents published by Elsevier, Springer, Taylor & Francis (leading publishing houses). They use it to make mathematics accessible in academic monographies, textbooks and journals. Given that they publish hundreds of journals, it is likely that they have a professional converting program. We do not have such resources. However, later we describe some free online tools, and how to overcome their limitations.

Other publishing houses and online journal repositories: Project Euclid and Jstor, do not use this dual system. However, the fact that at least some leading publishers use this technology encourages us.

3.2 HyperText Markup Language (HTML)

HTML is the markup language used by all browsers to create Web pages. The newest version is called HTML5; by default we refer to this version. Its commands are called *tags*. These tags correspond to the tags in PDF documents mentioned in the beginning of Section 1, which are crucial for accessibility.

HTML can include scripts, which add functionality and roughly correspond to using additional packages in LaTeX. An example:

HTML Code

```
<html>
<head>
<script type = ‘‘[type]’’ [link]> </script>
<title> Example </title>
</head>
<body>
<font face = ‘‘arial’’ size = ‘‘+1’’>
<h1> Main Title </h1>
<h2> Details </h2>
<p> Hello, world!  </p>
</body>
```

Output

Example[1]

Main Title

Details

Hello, world!

We used the font `Arial` because it is sans-serif; the default font for HTML is Times New Roman, which has serifs. Sans-serif fonts are recommended to ease the cognitive burden for people with certain learning and/or visual impairments.

Remark 1. Usually, a heading of the first order coincides with the title. In this case, instead of ‘Example’, we shall have ‘Main Title’.

HTML has *hyperlinks:* `<a href = ‘‘https://www.unr.edu/’’> UNR </a>` is the link to the UNR web site. To make content within your web page linkable, create an *anchor:* `<a href = ‘‘[URL of this link]’’> [descriptive name of your content] </a>`, then refer to this anchor as `[web-page-address]# [URL of this link]`.

[1] This is the text shown on the tab of the web page.

3.3 MathJax

MathJax is a script in HTML that can visualize math formulas in a browser as LaTeX would. It was created by the American Mathematical Society (AMS) for screen-readers and accessibility, see `https://www.mathjax.org/`. It can be enabled by adding to the preamble the following text:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/ 2.7.5/
MathJax.js?config=TeX-MML-AM_CHTML"> </script>
```

To use $ for formulas in MathJax, one should add to the preamble the following script:

```
<script type = 'text/x-mathjax-config' MathJax.Hub.Config({
tex2jax: {inlineMath: [['$','$'], ['\\(','\\)']]}});}> </script>
```

We do not need this latter preamble if we use Pandoc.

3.4 Converters

The markup languages LaTeX and HTML are different; they format tables, sections, lists, etc. differently. One needs a translator between them, and they are available: `tex4ht`, `htlatex`, LaTeX2HTML, Hevea, and others. See `https://texfaq.org/FAQ-LaTeX2HTML` for more information. They seem to do a good job, but require using the command line `cmd`, since these commands are not built in the usual TeX interface such as TeXworks. We did not assume audience familiarity with this, though it could greatly simplify this process.

One commercial converter which seems to be good is `MicroPress TeXpider`. It is marketed as *A tool that really works*, which is a big claim in this area. Unlike other programs mentioned above except for Adobe Acrobat Pro, this converter is not free, but costs $100 for individuals. We did not test this tool. It remains to be seen how accessible and screen-readable these outputs are.

3.5 Pandoc: Our Choice

One online tool that we found satisfactory is Pandoc: `https://pandoc.org`. Go to 'Demos', then 'Try Pandoc online'. In the original window (visual left side), choose LaTeX format. In the target window (visual right side), choose HTML5. Then copy the body of your LaTeX document to the left-side window, and click 'convert'. You might need to copy this bit by bit, because there is a limit on the document size. The resulting math mode in HTML is compatible with MathJax.

This converter creates `<span class="math inline">*</span>` around the inline math mode (where the asterisk stands for LaTeX math content), converting from single dollar signs in LaTeX; and `<span class="math display">*</span>` around the display math mode, converting from double dollar signs in LaTeX.

With the second preamble above, one need not worry about single and double dollar signs, as well as `\[ *\]`, since MathJaX will automatically recognize them as inline or display math mode. For a LaTeX document with plain text outside of dollar signs, simply copy-paste this LaTeX in HTML, without Pandoc, and put the two preambles above it.

3.6 Summary

Create HTML-MathJax documents by putting scripts mentioned above in the preamble, and (if necessary) converting the body of LaTeX to HTML via Pandoc.

This produces a good-looking, screen-readable, accessible document. In our tests, JAWS (a common screen reader) was able to read much of text and math.

However, our job is not finished. This conversion process can do most things, but not all. Below, we describe some of the possible issues, and whether they are actually solved by Pandoc or need to be edited manually. Some of them apply more to research papers rather than short lecture notes or homework/exam files.

4 Pandoc Issues

4.1 Sections and Headings

LaTeXuses commands `section`, `subsection`, and `subsubsection`. This corresponds to the document class `article`; in the class `book` we also have command `chapter`. HTML uses *headings:* `<h1>` for main topics, `<h2>` for subtopics, up to `<h6>` for sequential categories. Often, `<h1>` is the title of the page. Pandoc converts them well, and no manual work is required (unless the author wants to change the size of headings).

4.2 Lists

Both markup languages have commands for numbered lists and bulleted lists. Pandoc works well here, too. No manual work is required, except for the following case: LaTeX package `enumitem` allows one to produce lettered lists, with [a], [b], [c] items. Since this requires a special package, we doubt that Pandoc will covert this to HTML format. In HTML, there is an option for bulleted lists with bullet points having special shapes, such as squares. In our tests, Pandoc failed to adequately convert similar LaTeX lists, where changing the shape of this bullet is more difficult. It can be done manually in HTML, if desired. However, this level of control is often optional and not essential in practice.

4.3 Metadata

LaTeX has special commands for authors, title, keywords, abstract, and AMS subject classification. Pandoc does not translate them, but simply removes them. The author must reproduce these in HTML-MathJax manually. However, these are not required for screen reader accessibility (except the title).

4.4 Images

For external images attached to LaTeX such as `.pdf`, `.png`, `.jpg`, Pandoc transfers LaTeX pictures commands to the HTML commands: the LaTeXenvironment `figure` is converted into HTML tag `<figure>`, and the LaTeXenvironment `figure*` the (star means no automatic numbering) is converted into `<pic>`. Unfortunately, there is no way to convert automatic

LaTeX numbering to HTML. One should do it manually, as for theorems. Two- or many-part pictures in LaTeX are converted into separate pictures in HTML. One can draw simple pictures in LaTeX itself. We select two well-known LaTeX packages.

1. **Xy-pic**. This package is good for diagrams. MathJax can be made Xy-pic friendly with XyJax, `http://sonoisa.github.io/xyjax/xyjax.html`. But this software is still under development. More importantly, it requires the use of command prompt, which we assumed the audience does not know.

2. **TikZ**. This is a powerful package that allows the user to draw the following types of pictures in LaTeX: arrows, circles, nodes, other geometric figures. However there is no way to make TikZ MathJax-compatible. We would recommend creating separate pictures in JPG or PDF from TikZ or Xy-pic, and then include them in the HTML document.

4.5 Equation Numbering

As discussed before, Pandoc does a great job converting math to a MathJax-readable form inside an HTML document. However, a user needs to do some manual work to convert references to equations. In LaTeX, we have:

LaTeX Code

```
\begin{equation}
\label{this}
(a + b)^2 = a^2 + 2ab + b^2
\end{equation}
We refer to equation \eqref{this}.
```

Output

$$(a + b)^2 = a^2 + 2ab + b^2 \tag{1}$$

We refer to equation (1).

Pandoc HTML output

```
<p>
<span class="math display">\[\label{this}(a + b)^2 = a^2 + 2ab + b^2\]</span>
We refer to equation
<a href="#this" data-reference-type="eqref" data-reference="this">[this]</a>.
</p>
```

The number (1) would not appear. We need to manually change `[this]` to `$\eqref{this}$`. Then references will correctly work in HTML-MathJax. There is a way in LaTeX to make references dependent on a section; for example, the second equation in the third section will be numbered as (3.2). We did not test this in Pandoc; this is left for future research.

4.6 Tables

LaTeX and HTML have different commands for tables, but Pandoc seems to convert them very well. No manual work is required.

4.7 Theorems, Definitions, and Other Environments

In mathematics, theorems, lemmas, definitions etc. are formatted as separate text blocks, and their names are in boldface. In addition, theorems and lemmas (but not definitions) have statements which are italicized. In LaTeX, there are built-in *environments:* pieces of text which have special formatting inside. We already saw the environment `equation` which creates math mode in a separate line. Here is an example of the environment `theorem`. The LaTeX command `label` has an argument, which is the label assigned to this theorem. This is very similar to the concept of an *anchor* in HTML. We refer to this label via the LaTeX command `ref`.

LaTeX Code

```
\begin{theorem}
\label{that}
We have:  $x + y = z$.
\end{theorem}
We refer to Theorem \ref{that}.
```

Output

Theorem 1. *We have:* $x + y = z$.

We refer to Theorem 1.

Remark 2. The command `ref`, unlike `eqref`, does not provide brackets around the number.

In the LaTeX preamble, put `\newtheorem{theorem}[Theorem]` to use this environment. The process is similar for `lemma`, `proposition`, `definition`, `example`, `remark`, and other commonly used *statement environments* (environments used to format statements). Note that text is commonly italicized inside the *theorem-type environments:* `theorem`, `lemma`, `proposition`, but not inside other environments, mentioned above.

Theorems and other types of environments are automatically numbered: the second environment `theorem` produces **Theorem 2.** One can number environments of different types together, and include the section number in the theorem number (**Theorem 3.2**).

Unfortunately, Pandoc simply removes these theorem environment commands, because it cannot process them. The only way is to manually create a special heading for the theorem, and to manually italicize the internal text in HTML.

Remark 3. One can use CSS to automate formatting theorems, definitions, and other environments, see `http://felix11h.github.io/blog/mathjax-theorems`. However, we assumed that our audience does not know CSS. Regardless, using CSS does not support automatic numbering and references: One must do references manually, using anchors described in subsection 3.2 `<a link = ‘‘[text]’’>` and `<a href = ‘‘[text]’’>`

4.8 Macros

LaTeX allows users to define short versions of commands and even new commands: Instead of typing `\Rightarrow` for the symbol $\Rightarrow$, we write `\newcommand{\Ra}{\Rightarrow}` in the preamble, and then continue typing `\Ra` in the main text. To replicate this in HTML-MathJax, please type

```
<div style="display:none">
\(
\newcommand{\Ra}{\Rightarrow}
\)
</div>
```

This HTML code will not be displayed on the web page, but will be treated as a macro by MathJax. The same can be done for more complicated commands with arguments, such as `\norm{a}` with an argument `a`, defined as `\newcommand{\norm}[1]{\lVert#1\rVert}`. This gives us $\|a\|$. This method works for the command `\DeclareMathOperator`, which is used when we wish to type for example ‘exp’, not ‘exp’, in math mode. This last exp looks like italics exp, but actually these are letters `exp` in math mode.

4.9 Bibliography

LaTeX can automatically create bibliographies in two ways.

1. **Inside the main document.** `thebibliography` allows to create citations: `\cite{Book}` `[citation text]` and then refer to them as `\ref{Book}`. The output will look like [23] or [KS91]. Note the square brackets here, as opposed to round brackets for equation references, and no brackets for theorem references. Pandoc converts citations, but does not allow referencing. We introduce referring tags manually, as described above.

2. **From separate reference `.bib` file.** This requires a separate command `BibTeX` which is usually built in the TeX interface. However, this does not seem to be compatible with HTML-MathJax, at least without special packages and the command line.

4.10 Accents

LaTeX can create accents for names of non-English origin: `Poincar\'e` for Poincaré, `It\^o` for Itô, and others. HTML can do this, see `http://sites.psu.edu/symbolcodes/codehtml/`. Pandoc does not convert this. One would need to do this manually in HTML.

5 MathML and Canvas

MathML is a markup language created for browsers to display math content. However, this is not as simple as transferring the LaTeX code to the HTML source. Rather, this is a completely different markup language, a derivative of *Extensible Markup Language* (XML).

MathML is used within HTML, and is compatible with (more advanced) CSS. In fact, MathJax works by converting LaTeX formulae to MathML on the go: When we use MathJax, we implicitly use MathML. This language is screen-readable and tagged in HTML. See `https://www.w3.org/Math/`, `http://www.csun.edu/ hcmth008/technology.html` (Jacek Polewszak, CSU Northridge), and `http://accessibility.psu.edu/math/`.

Canvas (the UNR learning management system, also known as WebCampus) uses MathML in its math mode. In fact, if you create math using 'Insert Math Equation' button in Edit mode, and then switch to HTML editor mode instead of Rich Text mode, you can see the HTML/MathML script. The 'Advanced Option' mode in Canvas coverts LaTeX into MathML. Such script is displayed correctly in browsers (outside Canvas).

However, to the best of our knowledge, this math content is not screen-readable. This seems strange, because in our tests MathML was correctly read by a screen reader. By the way, HTML/MathJax script inside 'HTML mode' of Canvas does not render properly: Canvas cuts the preamble which includes MathJax.

WebCampus has MathJax installed in its LaTeX, plugin. Thus math typed via this plugin is readable by the screen reader JAWS (not NVDA) in our tests.

There are command line executions for converting LaTeX to MathML. Online converters can be found on `https://www.mathtowebonline.com/` but it failed in our tests. Another converter LaTeXMathML can be installed from `http://math.etsu.edu/LaTeXMathML/` (We did not test this.) We did not find such an option in Pandoc online. Let us also mention the commercial editor `MathType` (we have yet to test it).

Unfortunately, MathML is very primitive: It is hard to type even a simple formula. It requires learning yet another markup language, because it is different from both LaTeX and basic HTML. Thus one should either use MathJax or convert LaTeX into MathML using a converting software (some are mentioned earlier in this section). Also, some users claim that MathML does not work in Google Chrome without MathJax (although we had a different experience).

All in all, MathML does not seem to give us any advantage over MathJax, except that pure MathML is rendered on the screen faster than MathJax.

6 Tried & Failed Methods

6.1 LaTeX Packages and Their Failures

One can include packages in LaTeX by the command `\usepackage{[packagename]}`. There exist packages devoted to accessibility. One is `accessibility` designed by Babett Schaltz as her Ph.D. thesis. See the links on TeX Stack Exchange:

`https://tex.stackexchange.com/questions/261537/`

`https://tex.stackexchange.com/questions/19279/`

`https://tex.stackexchange.com/questions/79947/`

But in our trials, it does not compile some documents well, and the Ally accessibility score is sometimes relatively low (70% or even lower, even for LaTeX documents with only simple commands). There are improvements of this package, see the second answer in the first link above, but they seem hard to use.

Some improvements exist in packages called `accessibility-meta` and `corporate` created by Andrew Clifton: see the intstructions on the following web pages:

`https://github.com/AndyClifton/AccessibleMetaClass`

`https://github.com/AndyClifton/CorporateLaTeX`

But this still does not compile well.

There are other 6 packages on CTAN: `https://www.tug.org/twg/accessibility/` and 5 packages on TUG: `https://ctan.org/topic/accessible`. Among these packages, the most important seem to be `axessibility` and `accsupp`. But they failed our tests.

6.2 Alternative Publishing Systems

As mentioned in Section 1, there exist other publishing systems: XeTeX, XeLaTeX, ConTeXt, and others. Some, see `https://tex.stackexchange.com/questions/418954/`, are proposed as tools for accessibility. However, in our tests, these have failed to compile or create accessible PDF documents, as measured by Ally. Also, these systems are not widely known.

7 Summary and Recommendations

We described a way to make LaTeX accessible for visually impaired students via conversion through Pandoc to HTML-MathJax. We discussed in detail that Pandoc does not convert everything automatically, and we need to do some work manually. This is especially true for research articles, not so much for lecture notes and assignments. In other words, it is easier to deal with materials for undergraduate teaching than it is with research papers.

We state the following recommendations for busy faculty who wish to maximize accessibility while minimizing their time commitment. There are five options, listed from most to least preferable:

1. For your teaching, type math in Web Campus via the LaTeX plugin, instead of uploading prepared PDF files.

2. Instead of converting previosuly created LaTeX content to PDF, convert it to HTML-MathJax via Pandoc as described above. If it is hard to preserve all automatic numbering and references, you are welcome to just skip it. Usually teaching materials do not have much of such content.

3. Published HTML filed can be added and used as files in Canvas. Students can open them in a browser and use assistive technologies, such as screen readers, if desired.

4. For publications in Springer, Elsevier, and other publishing houses which have HTML-MathJax versions of articles on their Web sites together with PDF versions, just refer the student to them.

5. If nothing else works, talk to students individually and refer them to the Disability Resource Center (DRC) and Teaching and Learning Technologies (TLT).

Technology in this area is still new and not universally accepted. An initial focus of scientists and instructors was to create math that will display properly, and now we're learning how to make that math read well with assistive technologies to improve the user experience and to satisfy accessibility compliance guidelines. More cooperation is needed, in the spirit of respect and mutual trust. Another reason that there is not currently in-depth accessibility support is because this technology is created mostly free and open source by professors and graduate students in their spare time. We hope that usage will coalesce around one best model, which becomes universally accepted. This has indeed happened with other aspects of mathematical publishing. Further research is needed, because this technology is still very much emerging.