Preparing Accessible Math Documents using MathJax v3

Volker Sorge

School of Computer Science
University of Birmingham
cs.bham.ac.uk

Progressive Accessibility Solutions
Birmingham, UK
progressiveaccess.com

MathJax Consortium
mathjax.org

Empower 2019, Delhi, 14 October 2019

Slides and material at:
http://progressiveaccess.com/empower19

Overview

1. Generating Accessible Math documents for the Web
2. Using and customising MathJax v3.0 accessibility extension
3. Accessibility of STEM Diagrams

Part 1:
Generating Accessible Math documents for the Web

A Quick Introduction

- Learn two important open source tools: Pandoc and MathJax
- Transform documents using PanDoc
- Access Mathematics using MathJax's accessibility extension
- Learning outcomes:
  - Know how to convert your own documents
  - Be familiar with MathJax's a11y extension
  - Understand MathJax options to customise the accessibility experience
  - Be able to teach others!

What is MathJax?

- MathJax is a JavaScript library for rendering Mathematics in all browsers
- Can take LaTeX, AsciiMath, and MathML as input
- Generates browser output, e.g. HTML/CSS, SVG
- Standard Maths rendering solution for: stackexchange, wordpress blogs, mediawiki, etc.

MathJax is the de facto rendering solution of (nearly) all Mathematics on the web (in 2016: 35 million unique daily rendering requests via CDN)

`http://www.mathjax.org`

What is Pandoc?

- Pandoc is a flexible multi-format transformer between document formats
- It supports the transformations of most common sources for Maths document:
  LaTeX, Word, Markdown
- It allows for the direct integration of MathJax (v3 soon)

`http://pandoc.org`

- Note
  - There are a number of other document transformers out there
  - Many source dependent, i.e., one-source-to-many
  - Pandoc is really many-to-many, open source and easy to use in a pipeline

Steps in this Part

- Getting your Document to the Web
- Get familiar with PanDoc
- Create or download a math document
- Translate document into HTML format
- Run it locally in a browser

What you should have available

- Pandoc binary
  - Download it from `https://progressiveaccess.com/empower19`
  - Or alternatively from `https://pandoc.org`. This is an older version, though.
  - You can build directly from `https://github.com/jgm/pandoc`. But this takes a long time!
- Shell
  - On Windows: Powershell is best, Command Line will do.
  - On Linux/MacOSX: any shell or terminal is fine
- Firefox or Chrome
- Screen Reader
- ASCII text editor: NotePad, vi, Emacs, jedit, etc.

Getting familiar with Pandoc

- Open your shell/command line.
- Make sure you know your path to the pandoc executable
- We assume that it is at `PATH`
- Run the command below, which should display pandoc's options.

```
${PATH}/pandoc --help
```

Getting started with Pandoc

- Let's convert our first file: We by converting LaTeXinto HTML
- Use your favourite LaTeXfile or download `https://progressiveaccess.com/empower19/examples/example1.tex`

```
pandoc --standalone -t html -o example1.html example1.tex
```

- Inspect the options:
  - `-t` is the target format. In our case HTML.
  - `-o` is followed by the output filename
  - `--standalone` ensures that pandoc produces a fully HTML file not just an HTML fragment.

Converting a Word Document

- Similar to LaTeXwe can also convert word documents containing mathematics
- If you have one, try your own document.
- Alternatively download the example file at `https://progressiveaccess.com/empower19/examples/example2.docx`

```
pandoc --standalone -f docx -o example2.html example2.docx
```

- A note on the options:
  - Here we use the `-f` option which specifies the input format.
  - Neither `-f` or `-t` are strictly necessary if the formats can be deduced from the file extensions

Inspecting the Output

- Both conversions will have produced some noise on the console.
- Let's inspect the output:
  - Load example1.html or example2.html into a browser
  - You will note that some of the Math looks somewhat ugly and some is raw TeX
- pandoc tries to translate Math as much as possible, but often fails

Introducing MathJax

- Let's provide a rendering solution for the math
- We could add it to the html files directly to
- Alternatively we can ask pandoc to do it for us
- Example:

```
pandoc --standalone --mathjax -o example2.html example2.docx
```

```
pandoc --standalone --mathjax -o example1.html example1.tex
```

- Note that we have omitted the from and to options.
- Instead we now have a `--mathjax` option.

Inspecting the Output Again

- pandoc should have produced a lot less noise on the console.
- That's because it has a rendering solution and does not bother to translate the Math itself.
- Let's inspect the output in the browser again.
- All the math should now render nicely
- It's still not accessible, but we will get to that later

Inspecting the Code

- Load the html files into an ASCII editor
- Note, do *not* load them into an office suite like Word, LibreOffice, ...
- Inspecting the code, you can now see the MathJax script tag

```
<script src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/tex-mml-c
        type="text/javascript"></script>
```

Loading MathJax

- The script tag ensures that MathJax is loaded into the page
- It is loaded directly from its Content Distribution Network
- It is loaded directly from the content distribution network
- It uses a default configuration, for input and output
- Specifically it allows
  - LaTeXor MathML input
  - HTML/CSS output
- Other potential output is for example SVG
- For other configurations that can be provided as default see: `http://docs.mathjax.org/en/latest/web/components/combined.html`

Preparing a HTML Directly

- Let's prepare our own HTML document with Maths
- Write a HTMl document
- Add you favourite maths
- Load it into the browser to see if it renders
- **Please do this as an exercise**

Exercise

1. Build a web document for with your favourite maths.
2. If you don't have any then use the quadratic formula:

```
\[
    x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}
\]
```

3. Proceed in three steps
    1. Create a skeleton HTML document
    2. Add the MathJax script tag to the header
    3. Add some LaTeX to the body

Hint: Basic web document

```
<html>
  <head>
   ... MathJax script tag goes here ...
  </head>

  <body>
   ... Quadratic equation goes here ...
  </body>
</html>
```

A Solution

```
<html>
  <head>
    <title>Empower-19 Exercise</title>
    <script src="https://cdn.jsdelivr.net/npm/mathj
  </head>

  <body>
    <h1>Quadratic Equation</h1>
    \[
    x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}
    \]
  </body>
</html>
```