Using Pandoc to convert LaTeX: Accessibility

Using Pandoc to convert LaTeX

To make LaTeX documents accessible, you can

1. Create a PDF using an engine (e.g., pdflatex, LuaLaTex, XeLaTeX) and then manually tag the PDF using Acrobat.
2. Use the `accessibility` package in LaTeX to add `alt` tags to pdfs. If you need to tag math, you can use the `axessibility` package.
3. Convert LaTeX into HTML. Given the limitations and workload involved in the first two approaches, the recommended approach is to convert LaTeX to HTML.

HTML Conversion: Simple or Complex?

1. If your document is simple (basic math, minimal use of packages, no custom environments/macros), then use **pandoc**
2. If your document is complex, then use the **lwarp** package.

In these slides, I'll discuss `pandoc`.

Pandoc approach

1. Let's assume you have a TeX engine on your machine (e.g., **MikTeX** or **MacTeX**).
2. Install pandoc: **pandoc installation instructions**.
3. If you are not sure if pandoc is the right approach, you can also try pandoc in the browser here: **pandoc try** or use the pandoc VSCode extension (limited to converting from markdown).

How to use pandoc

- Use **pandoc** by running commands in terminal / CLI (e.g., cmd, powershell, bash):
    - Mac: Command(⌘) + Spacebar, type "Terminal"
    - Windows: Searchbar, type "cmd"
    - VSCode: Ctrl+` opens the terminal.

In the terminal

Navigate to the directory that the `.tex` file is located. Then use the following command:

```
pandoc sample.tex -o sample.html
```

- runs pandoc on `sample.tex`
- `-o` says to output a file: `sample.html`
- Works *if* your document is simple (no math, figures, etc) and you want to paste the HTML into an LMS (e.g., CANVAS).

Standalone option

If you need a standalone document (e.g., with a html header, basic styling, javascript, meta tags), then you'll add the `-s` or `standalone` option to make it a standalone document:

```
pandoc sample.tex -s -o sample.html
```

Document strucure

Accessible documents have good organizational structure so they can be easily navigated:

1. Title and Heading tags
2. Numbered headings
3. Sections
4. TOC

Headings

LaTeX documents should be organized with sections and subsections.

```
\section{Section 1}
some content
\subsection{Section 2}
more content
```

Title and headings

The `title` of the document is labeled as a `<title>` tag in the metadata and as an `<h1 class="title">` tag.

```
<h1 class="title">Sample LaTeX Document</h1>
```

`\section` will be `<h1>`, `\subsection` will be `<h2>` and so on.

Setting Metadata Variables

The title is set with metadata variables extracted from the LaTeX or can be set manually using the `-M` metadata variables, e.g., `title`, `author`:

```
pandoc -M title="My Document" -M author="John Doe" input.md -o output.html
```

Headings

The NVDA screenreader will read `\section` will produced a level 1 heading, `\subsection` a level 2 heading, and so on.
Here is the readout from NVDA:

```
Introduction  heading    level 1
Main Content  heading    level 1
Some math  heading    level 2
Using tables  heading    level 2
Links  heading    level 2
Pictures and graphics  heading    level 2
Tikzpicture  heading    level 2
Custom Macro  heading    level 2
Custom Environment using a package  heading    level 2
Conclusion  heading    level 1
no next heading
```

Heading numbering

W3C does not appear have any guidelines about whether headings should be numbered: **W3C page on Headings**. Nevertheless, it helps with organization, so to number sections, add the `-N` option.

```
pandoc sample.tex --mathjax --toc -s -N --section-divs--resource-path=imgs -o sample.html
```

Sections

You can add `section` tags but there is a note that advises against this as it can cause issues with screen readers parsing headers (see **Assistive Technology Notes**)

```
pandoc sample.tex --toc -s -N --section-divs -o sample.html
```

TOC

To add a table of contents, use the `--toc` option (needs to be in the command rather than the YAML file):

```
pandoc sample.tex --mathjax --toc -s -o sample.html
```

- This will add a table of contents toward the top of the page.
- It is put in a `<nav>` tag.

```
<nav id="TOC" role="doc-toc">
<ul>
<li><a href="#introduction" id="toc-introduction">Introduction</a></li>
<li><a href="#main-content" id="toc-main-content">Main Content</a>
<ul>
<li><a href="#some-math" id="toc-some-math">Some math</a></li>
<li><a href="#using-tables" id="toc-using-tables">Using tables</a></li>
<li><a href="#links" id="toc-links">Links</a></li>
<li><a href="#pictures-and-graphics"
id="toc-pictures-and-graphics">Pictures and graphics</a></li>
<li><a href="#tikzpicture" id="toc-tikzpicture">Tikzpicture</a></li>
<li><a href="#custom-macro" id="toc-custom-macro">Custom Macro</a></li>
<li><a href="#custom-environment-using-a-package"
id="toc-custom-environment-using-a-package">Custom Environment using a package</a></li>
</ul></li>
<li><a href="#conclusion" id="toc-conclusion">Conclusion</a></li>
</ul>
</nav>
```

TOC depth

If our document has several subsubsections and we don't wish to include all of them, we can specify the depth of the table of contents with `--toc-depth=NUMBER`.

```
pandoc sample.tex --mathjax --toc --toc-depth=1 -s -o sample.html
```

The TOC will only include sections rather than subsections.

Math

Math is another challenge for screenreaders. Ideally, you want to convert to MathML or MathJax rather than an `.svg` that has alt text.

MathML

To make math display correctly, add `--mathjax` or `--mathml`.

MathML

```
pandoc sample.tex --mathml -o sample.html
```

MathML

Here is the Basel problem equation:

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$

```
$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}$$
```

```
<p><math display="block" xmlns="http://www.w3.org/1998/Math/MathML"><semantics><mrow>
<munderover><mo>∑</mo><mrow><mi>n</mi><mo>=</mo><mn>1</mn></mrow><mo
accent="false">∞</mo></munderover><mfrac><mn>1</mn><msup><mi>n</mi><mn>2</mn></msup>
</mfrac><mo>=</mo><mfrac><msup><mi>π</mi><mn>2</mn></msup><mn>6</mn></mfrac></mrow>
<annotation encoding="application/x-tex">\sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}
{6}</annotation></semantics></math></p>
```

MathJax

Alternatively, if using `--mathjax`, make the document standalone (since we need the script tag that links to mathjax):

```
pandoc sample.tex --mathjax -s -o sample.html
```

```
<p><span class="math display">\[\sum_{n=1}^{\infty} \frac{1}{n^2} =
\frac{\pi^2}{6}\]</span></p>
```

MathJax

- **ARIA** labels that provides a spoken description of the math
- Keyboard navigation
- Fallback MathML

Tables

- Tables to HTML (with semantic `<table>`, `<thead>`, and `<th>` tags).
- Let's look at a LaTeX table with three columns, three rows, alignment, a caption, and a label

Table LaTeX

```latex
\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|}
\hline
\textbf{Animal} & \textbf{Sound} & \textbf{Habitat} \\\hline
Cat & Meow & House \\\hline
Dog & Bark & House \\\hline
Rabbit & Thump & Burrow \\\hline
\end{tabular}
\caption{A comparison of pets}
\label{tab:sample}
```

```
\end{table}
```

Table HTML

A comparison of pets

Animal	Sound	Habitat
Cat	Meow	House
Dog	Bark	House
Rabbit	Thump	Burrow

Table HTML

```
<div id="tab:sample">
<table>
<caption>A comparison of pets</caption>
<thead>
<tr>
<th style="text-align: center;"><strong>Animal</strong></th>
<th style="text-align: center;"><strong>Sound</strong></th>
<th style="text-align: center;"><strong>Habitat</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td style="text-align: center;">Cat</td>
<td style="text-align: center;">Meow</td>
<td style="text-align: center;">House</td>
</tr>
<tr>
<td style="text-align: center;">Dog</td>
<td style="text-align: center;">Bark</td>
<td style="text-align: center;">House</td>
</tr>
<tr>
<td style="text-align: center;">Rabbit</td>
<td style="text-align: center;">Thump</td>
<td style="text-align: center;">Burrow</td>
</tr>
```

```
</tbody>
</table>
```

Table: NVDA

The HTML reads sensibly in NVDA, indicating columns and rows:

```
out of caption  column 1  Animal     column 2  Sound     column 3  Habitat
row 2  Animal   column 1  Cat
Sound   column 2  Meow
Habitat   column 3  House
row 3  Animal   column 1  Dog
Sound   column 2  Bark
Habitat   column 3  House
row 4  Animal   column 1  Rabbit
Sound   column 2  Thump
Habitat   column 3  Burrow
out of table
```

Links

In LaTeX, links are created using the `hyperref` or `url` package.

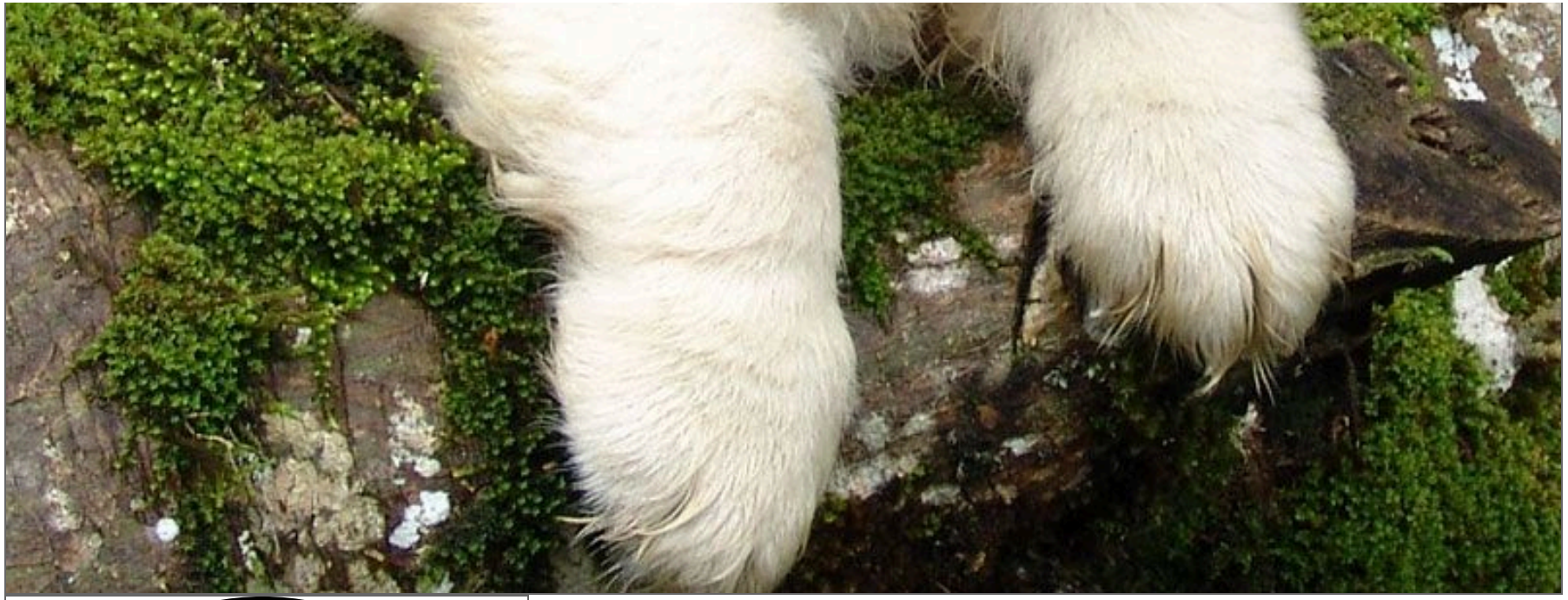
```
\href{https://www.latex-project.org/}{The LaTeX Project}
```

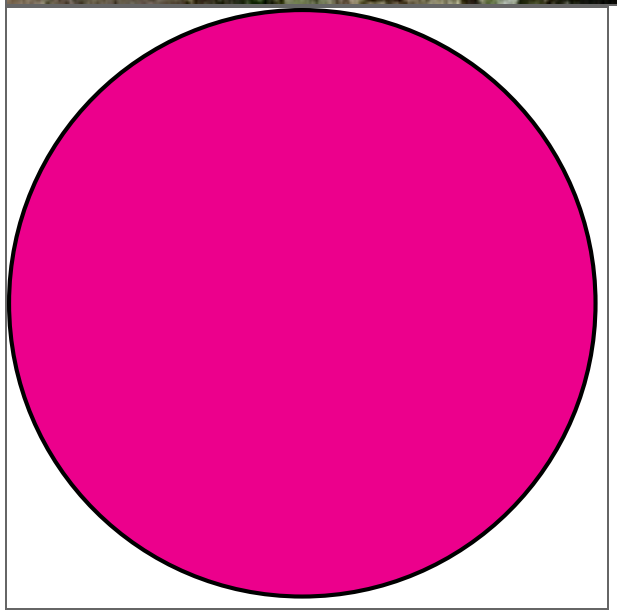
HTML is accessible provided you are following best practices:

```
<p><a href="https://www.latex-project.org/">The LaTeX Project</a></p>
```

Images

All non-text content needs a text alternative. The way this is achieved with images is to provide alt text.





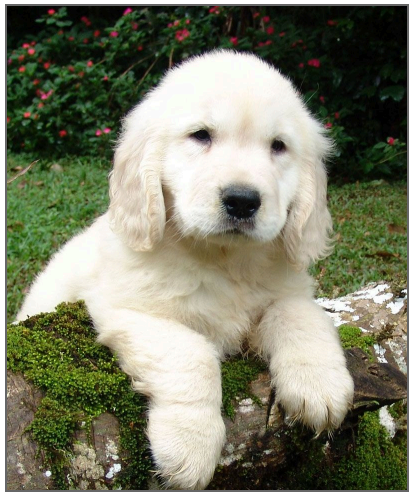
Specifying the resource path

- If you have images, they might not appear in your conversion, so you may to specify your resource path explicitly.
- How? Add the `--resource-path=<img_path>` . I have a photo of a puppy in an `imgs` folder that is in the same directory as my `sample.tex` file.

```
pandoc sample.tex --mathjax --toc -s --resource-path=imgs -o sample.html
```

Pandoc default images

One problem is that the result is **not accessible** since there is not an alt tag for the image of the puppy in the html.



Add alt tag in LaTeX

If your image is a produced using the `graphicx` package, then simply add alt text as an option to `\includegraphics[alt={Alt text here}]{path_to_img}`.

```
\begin{figure}[ht]
\centering
\includegraphics[width=200px, alt={An image of a puppy on a mossy log.}]
{imgs/puppy_wiki.jpg}
% for HTML, width needs to be absolute units instead of width=.5\textwidth
\caption{A cute puppy from Wikipedia}
\label{fig:puppy}
\end{figure}
```

```
<figure id="fig:puppy">
<p><img src="imgs/puppy_wiki.jpg" width="200" alt="An image of a puppy on a mossy log."
/></p>
<figcaption>A cute puppy from Wikipedia</figcaption>
</figure>
```

Problem: if the image is complex, the alt text might be too long.

W3C longdesc recommendation

W3C makes several recommendations.

- W3C suggests using the `<longDesc>` property on the `<img>` element.
- This property is still usable in many browsers and screen readers but is depreciated (**link to longDesc**).

W3C text link recommendation

W3C alternatively suggests providing a *text link* to the description next to the image.

```
<p><img src="imgs/puppy_wiki.jpg" alt="An image of a puppy on a mossy log." />
<br>
<a href="image-descriptions/puppy_wiki.html">Long description of the puppy.</a></p>
```

The shortcoming of this approach is that the description and the image are not semantically linked.

MDN recommendation

MDN recommends encapsulating the image in an `<a>` element.
Don't write:

```
<img
  src="imgs/puppy_wiki.jpg"
  alt="An image of a puppy on a mossy log."
  longdesc="image-descriptions/puppy_wiki.html" />
```

Write this instead:

```
<a href="image-descriptions/puppy_wiki.html">
  <img src="imgs/puppy_wiki.jpg" alt="An image of a puppy on a mossy log." />
</a>
```

MDN recommendation

- The benefit of this approach is that the long description and the image are structurally linked but there is no semantic relation.
- there is no indication to the user that the link is to a long description of the image (the link could be to anywhere).

An improvement

I think a better approach would be to explicitly indicate that the link is to a long description:

```
<a href="image-descriptions/puppy_wiki.html"><img src="imgs/puppy_wiki.jpg" alt="An image of a puppy on a mossy log." /> Long description of puppy.</a>
```

W3C figure group

A similar (and probably better) approach suggested by W3C is to place the `<img>` element in a `<figure>` with the designated role `group`.

```
<figure role="group">
    <img src="chart.png"
        alt="Bar chart showing monthly and total visitors for the first quarter 2014 for
sites 1 to 3">
    <figcaption>
        <a href="2014-first-qtr.html">Example.com Site visitors Jan to March 2014 text
description of the bar chart</a>
    </figcaption>
</figure>
```

The first item of the group is the image with its alt text. The second item of the group is the `<figurecaption>`, which consists of a link to a long description of the image.

How the LaTeX should look

In converting a LaTeX document, we can sort of achieve this effect by

1. placing the image in a `figure` environment and then
2. linking to an `.html` file containing the longer description of the image in the caption.

```
\begin{figure}[ht]
\centering
\includegraphics[width=200px, alt={An image of a puppy on a mossy log.}]
{imgs/puppy_wiki.jpg}
\caption{\href{image-descriptions/puppy_wiki.html}{Long Description of the puppy.}} %
caption is to a long description of the puppy.
\label{fig:puppy_longdescription}
\end{figure}
```

```
<figure id="fig:puppy">
<p><img src="imgs/puppy_wiki.jpg" width="200"
alt="An image of a puppy on a mossy log." /></p>
<figcaption><a href="image-descriptions/puppy_wiki.html">Long Description of the puppy.
</a></figcaption>
</figure>
```

Cosmetic Items: Template Styling

Pandoc using a default template to style documents. Template variable flags can be used to set styling (or set in a YAML file):

```
pandoc pandoc input_file.md -s -M geometry=margin=1in -M fontsize=12pt -M colorlinks=true
-V linkcolor=magenta -V urlcolor=magenta -o output_file.html
```

Why not Pandoc?

At this point, it seems like `pandoc` can do everything! Let's consider two limitations.

Problem 1: Some images are produced by LaTeX drawing packages. A popular one is **TikZ**. Pandoc will not convert these images. There are three options.

Solution 1

- Put each diagram in a separate file
- Make it a standalone class
- Create a pdf of each diagram, then either
- Use **pdf2svg** or
- the Poppler command (`pdftocairo -svg circle.pdf circle.svg`) to convert each pdf into an `.svg`

Finally use the `iftex` package to use the LaTeX diagram if creating a pdf or the .svg if creating an HTML.

Solution 1: Example

I placed the `circle.svg` in the imgs folder:

```
\usepackage{iftex}

\ifPDFTeX
  % TikZ for PDF
  \begin{tikzpicture}
  \draw[fill=magenta] (0,0) circle (1cm);
  \end{tikzpicture}
\else
  % Image for HTML
  \includegraphics[width=100px]{imgs/circle.svg}
\fi
```

Solution 2: Pandoc filter

You could use a python pandoc filter to process latex tikz into images: **tikz.py**

Solution 3: Use lwarp

- Solution 1 is a lot of work for most people!
- Solution 2 is to use **lwarp**.

Problem 2

A second problem concerns the use of various packages. Pandoc just won't work with certain packages, which many writes of LaTeX rely upon, e.g., **tcolorbox**

```
\usepackage{tcolorbox}
\newtcolorbox{myspecialbox}[1]{
    colback=blue!5!white,
    colframe=blue!75!black,
    title=#1
}
```

```
\begin{myspecialbox}{My Exercise Box}
Here is some text in a box using tcolorbox. It has an option where I can specify the
title. It won't convert to HTML properly.
\end{myspecialbox}
```

Resources

1. **accessibility package documentation**
2. **axessibility package documentation**
3. **pandoc**
4. **pandoc general writer options**
5. **pandoc options affecting specific writers**
6. **lwarp**

Speaker notes