Creating Accessible Documents Using Markdown

Matt Bush

Creating Accessible Documents Using Markdown

Created by Matt Bush, Department of Chemistry, University of Washington

Table of Contents

Introduction

As shared by Provost Tricia R. Serio:

> On April 24, 2024, the Department of Justice published a rule on digital accessibility under Title II of the Americans with Disabilities Act (ADA). This rule requires the University's digital content to be accessible starting on April 24, 2026. Of note for instructors, this applies to course content such as videos, images, slide decks, documents, audio files, e-textbooks, course webpages and online tools.

Based on the principles of Accessible Instructional Materials (AIM) for faculty, I thought that Markdown would be an ideal way to quickly write AIM. Markdown is a lightweight markup language that uses plain text formatting syntax to create rich text. It's designed to be easy to write and read in its raw form, yet convertible to HTML and other formats.

What is Markdown

Markdown is a lightweight markup language. It allows you to write using an easy-to-read, easy-to-write plain text format that converts to structurally valid HTML.

Basic Markdown Syntax Examples

Headings

```
# Heading 1
## Heading 2
### Heading 3
```

Text Formatting

```
*This text will be italic*
**This text will be bold**
***This text will be bold and italic***
```

Lists

Bulleted list:

```
* Item 1
* Item 2
  * Subitem 2.1
  * Subitem 2.2
```

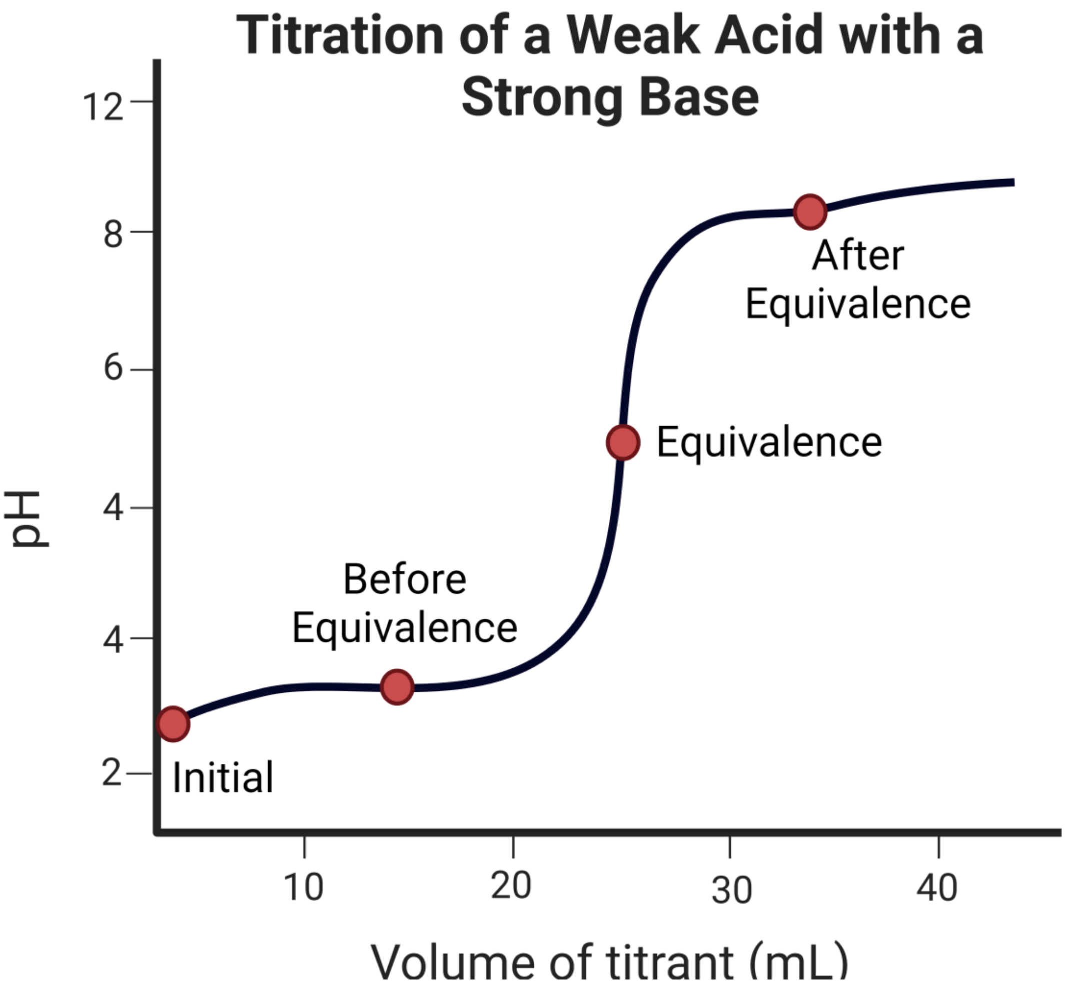
Numbered list:

```
1. First item
2. Second item
3. Third item
```

Links and Images

```
[Link text](https://example.com)
![Alt text for image](image.jpg)
```

Note that this structure makes it really easy to incorporate tags with links and images, which is often a critical challenge in generating AIM.



Code

~~~~
`Inline code` looks like this

```
// Code block
function example() {
  return "hello world";
}
```
~~~~

Blockquotes

~~~~
> This is a blockquote
> It can span multiple lines
~~~~

Equations Using Mathjax/Latex

~~~~
$$Latex = works$$
~~~~

$$Latex = works$$

Key Markdown features include:

- Headings with # symbols

- Text styling like **bold** and *italic*
- Lists (bulleted and numbered)
- Links and images
- Code blocks and inline code
- Blockquotes
- Horizontal rules

Markdown is widely used for:

1. Documentation and README files
2. Forum and comment systems (Reddit, Stack Overflow)
3. Messaging platforms (Slack, Discord)
4. Note-taking applications
5. Content management systems
6. Technical writing

It's popular because it maintains readability while adding formatting capabilities, works across platforms, and doesn't require special tools to create or view.

Disclaimer

Although I now use Markdown in the preparation of teaching materials, I do not necessarily recommend this approach to others. There is still a lot of friction in using Markdown with Canvas.

Methods For Using Markdown With Canvas

Creating Canvas Pages Using Markdown and HTML

For CHEM 428/528 during Winter 2025, I tried the following:

1. Write course pages using Markdown in VS Code with the "Markdown All in One" extension. That extensions supports the automatic creation of table of contents and exporting HTML. Here are example of a page in Markdown and a preview of the formatted Markdown.
2. Export an HTML file.
3. In Canvas, open a page, go to Viewer>>"HTML Editor", and paste the HTML code into the page.

Reflections

- Overall, this worked pretty well, even for LaTex equations!
- My greatest frustration was relative URLs (in Canvas) often behaved inconsistently and in unexpected ways. This seemed to be a Canvas problem. Ultimately, I switched to absolute URLs, which is not best practice.
- My assumption is that the "Markdown All in One" extension uses Pandoc to generate html. That may be more flexible for some workflows, but I haven't tested it for this purpose.

Creating Accessible PDFs from Markdown Using Pandoc and Word Yields Accessible PDFs.

After a few different attempts, I settled into using Pandoc, which is a powerful command line tool for converting documents. My goal was to have PDFs with equations and PNG figures.

Note that most Markdown viewers and tools implement the "Commonmark" standard, whereas Pandoc defaults to an older, stricter standard. If you have Latex equations, I suggest the "Commonmark with Extensions" flag (`commonmark_x`) that I use below.

This pipeline generally yields PDFs that have a 98% Ally score.

1. Create a .DOCX file using Pandoc:

```
pandoc -r commonmark_x accessible-documents.md -o accessible-documents.docx
```

2. Open the .DOCX file in Word and save as a .PDF file.

These PDFs get dinged by Ally because they "lack a PDF title". Ally suggests steps to add a title in Word before PDF creation. From Ally:

> A PDF title is a more descriptive and meaningful version of the file name. PDF titles are often visible in the PDF window or tab. This visibility makes it easier to distinguish multiple PDFs before diving in.

To make the Word documents look how you want, instead generate the .DOCX files using:

```
pandoc -r commonmark_x accessible-documents.md -o accessible-documents.docx --reference-
doc reference.docx
```

See in reference-doc in https://pandoc.org/MANUAL.html#options for instructions for creating a reference file.

Note that Pandoc seems to be more fussy about Markdown formatting than other Markdown engines that I have used. I haven't investigated whether there are settings that can be tweaked for this.

Pipeline Using LaTex for PDF Generation Did Not Yield Accessible PDFs

This appears to be limitations in the underlying LaTeX engines, e.g., XeLaTex. From Pandoc:

> Pandoc defaults to LaTeX to generate PDF. Tagging support in LaTeX is in development and not readily available, so PDFs generated in this way will always be untagged and not accessible. This means that alternative engines must be used to generate accessible PDFs.

The Pandoc documentation suggests some options for implementing tagging, but I have not pursued those further.

Direct conversion (`pandoc.exe input.md -o output.pdf`) did not yield accessible PDFs for me.

Indirect conversion via HTML using:

```
pandoc.exe input.md -o output.html
pandoc.exe output.html -o output.pdf
```

This did not yield accessible PDFs for me.

Indirect conversion via DOCX using:

```
pandoc input.md -o output.docx --reference-doc reference.docx
pandoc output.docx -o output.pdf
```

This did not yield accessible PDFs for me.

Asks for UW IT

Based on my experiences, I have identified two critical functions that would facilitate the creation of AIM:

1. Support the creation of Canvas pages using Markdown with MathJax equations. This would be transformative for creating content for STEM fields.
2. Provide clear instructions for using relative URLs in Canvas. Currently, relative URLs on the homepage do not behave in a consistent manner, and in my experience, have to be replaced with absolute URLs. This hinders content development and the recycling of pages on multiple Canvas sites.

Acknowledgements

This documented was edited with assistance from Claude 3.7 Sonnet.