Building a Website with Make4ht: An Overview

David Friant

January 22nd, 2026

Abstract

Creating a personal and/or professional website from scratch in the modern day is a way to take ownership of one's digital life. This article previews a series of several others related to the development of a website using the Make4ht LaTeX compiler. The process described requires a good working understanding of LaTeX first and foremost, but also HTML, CSS, Python, and a bit of Javascript. While this work is only a preview of the more in-depth articles, it fully demonstrates not just basic typesetting, but also tables, complex equations, code listings with syntax highlighting, and raster and vector images.

Key Words: LaTeX, Make4ht, HTML, CSS, IndieWeb

Motivation

There are many reasons to develop and maintain a personal website away from any of the giant social media companies which dominate today's internet landscape. Doing so allows you to have a place where you can display professional and/or personal projects while maintaining true ownership over where, when, and how they are presented. That is to say that it grants a degree of autonomy which is effectively impossible to achieve while shackled to any existing social media platform.

After making the decision to make your own website, there is the daunting decision of just how *precisely* to go about doing so. There is the obvious choice of using one of the many web host services[A] which provide templates and preconfigured website builders for ease of use. This is a perfectly valid choice if one only wants to separate from the big social media platforms. However, to some it may seem that you are only changing one corporate master for another, albeit less terrible. The alternative for those of such a mind, then, is to create your own website from scratch.

The natural starting point for such an endeavor is to learn how to use HTML, CSS, and some JavaScript. Fortunately, the W3Schools for all three are both free and of decent quality. Learning how to use HTML leads one quite naturally to an unfortunate conclusion: writing it by hand is an unpleasant experience.[B] Fortunately there are other formatting languages in the world. In particular, LaTeX should be familiar to any serious academic and would be a boon to learn even for those outside of academia looking to share their writing as it opens the door to customized typesetting.[C]

Even more fortunately, there are others who have had the motivation to use LaTeX for the production of HTML documents. Indeed, there are at least two active projects doing just that: latex2html (which will not be discussed further) and Make4ht. Make4ht is a LaTeX compiler just like pdfTeX, LuaTeX, and others, but it produces an HTML file instead of a PS, EPS, PDF, etc. file.

This series of articles is not intended as a full user guide for the Make4ht system, but instead as a log of sorts to show how it can be used used (both properly and improperly) to build a personal website. This includes both the places where Make4ht simply works smoothly and also those places where one must really work around the system to achieve their desired goal. It is a mixed bag that is not without its tribulations, but the highly educational journey and end result are worth it.

Goals

As this work pertains to the creation of one specific website, the goals listed hereafter are specific to it. Anyone who would make their own website should, of

[A] Services such as WordPress, Squarespace, and Canva for example.

[B] The verbosity of HTML really is a very unfortunate. Perhaps someday we will have a lighter weight alternative.

[C] Typst seems to be a modern up-and-coming competitor to LaTeX, but it's ability to output HTML is still experimental as of the time of writing.

course, draft their own requirements to suit their needs. With that stated, the goals below should be quite reasonable for someone wishing to create a website which consists mostly of "article" or "blog" type pages. Naturally, there will be some pages which require a unique touch (the main landing page, for example), but these goals are targeted such that the vast majority of the content can be generated from the LaTeX documents with little to no manual intervention.

- Source documents should be written in near-typical LaTeX to get high-quality PDF and HTML documents on compilation.
- The website built from these documents should nigh-exclusively provide static HTML with minimal JavaScript to provide limited functionality where CSS is insufficient.
- The website should be free from any specific framework or platform.
- There should be a high degree of customizability, with as much of it as possible stemming from CSS instead of being hard-coded into the webpages themselves such that the global style of the website can be changed in a single place and be reflected throughout.
- There should be both a light and dark color mode. All text, images, diagrams, code, etc. should remain easily visible in either mode.
- There should be high accessibility for disabled visually or dexterously impaired people. This should ideally take into account the screen-reader experience and ensure that all features are fully accessible using both just the keyboard and just the mouse.

Overview

Each of the following sections both describe and preview the specific features required for the website. Each of them should be considered a brief overview of the material included in a dedicated article which will be linked to in the section itself and may be otherwise found in the site navigation sidebar or by using the search function. If reading the HTML version of the document, the reader may wish to download the PDF version for comparison and vice versa.

Basic Typesetting

LaTeX, at its very core, provides a number of fundamental typesetting operations which should be considered necessary for any text-focused website such as:

- Paragraphing
- Sectioning
- Itemizations and Enumerations (Ordered and Unordered Lists)
- Basic Text Decorations
- Internal and External Links

The paragraphing and sectioning have already been well demonstrated by this document. Unordered lists have been demonstrated as well, but ordered lists have not. The following list mixes them as is allowed by both LaTeX and HTML:

- The articles in this project are:
  1. Overview
  2. LaTeX and Make4ht
  3. References and Footnotes
  4. Equations and Tables
  5. Code, Minted, and Pygments
  6. Raster and Vector Images
  7. Styling with CSS
  8. Building a Build System
- Future project topics might include:
  - Recreational Mathematics
  - Hobby Electronics
  - Creative Writing
  - Programming Projects
  - Music Compositions
  - Etc.

Simple text decorations are another fundamental feature that is easily accessible as demonstrated: **Bold Font**, *Italic Font*, `Monospace Font`, Underlines, and Overlines. Slightly more advanced decorations such as Double Underlines, Wavy Underlines, Dashed Underlines, Dotted Underlines, and ~~Strikethroughs~~ are also available with relative ease.

Internal and External links are also quite readily available, as seen with these internal links: §Goals, §Overview, §Basic Typesetting, and §Code Highlighting. Here is an external link to the Wikipedia article on LaTeX, and here is a link to the next article in this series which delves into much greater detail on what it takes to get the fundamentals of Make4ht working.

References and Footnotes

Footnotes[D] and references[1] provide a slight complication as the desired HTML behavior is substantially more complicated than simply putting text at the bottom of the page. As demonstrated here on the HTML version of this document, it is possible to have tooltip-like pop-ups for both footnotes and references in addition to placing an actual footnote at the bottom of the page just as with the PDF version of this document. Note that a good PDF viewer will likely preview the references and footnotes when hovering over the marker, not unlike a tooltip.

[D] Footnotes such as this one!

The process of getting these two features set up correctly is a bit more involved than the fundamental typesetting due to the dynamic nature of the feature, as such this link leads to a dedicated article on getting them working.

Equations and Tables

LaTeX has powerful typesetting abilities for advanced math. Make4ht supports a few different methods for outputting math: as a raster or vector image, MathJax, [2] and MathML.[3] Of these, MathML was chosen for the website. The following equations provide some arbitrary examples of the typesetting:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} w_i \left(x_i - \bar{x}\right)^2}{\sum_{i=1}^{n} w_i}} \tag{1}$$

$$f(x) = \begin{cases} -1, & x \leq -1 \\ -\frac{1}{2}x^3 + \frac{3}{2}x, & -1 < x < 1 \\ 1, & x \geq 1 \end{cases} \tag{2}$$

$$\bar{f}_{[a,b]} = \frac{1}{b-a}\int_a^b f(x)dx \tag{3}$$

While equations are largely straightforward (if one is already familiar with LaTeX, that is), tables are an entirely different beast. Due to the semantic differences between how a LaTeX table and an HTML table are described, Make4ht natively uses some rather unfortunate workarounds to make the output look decent, though at the cost of making the underlying HTML rather unnecessarily verbose and potentially ill-formed. As such, this link leads to the article which discussed the steps required to get good looking tables that are also well-formed HTML code. Note that this makes heavy use of Python code for post-processing the Make4ht output into well-formed HTML. It is advised that the reader be at least familiar with Python's syntax and have some hands-on experience before reading.

Table 1 showcases a periodic table of the elements made from scratch using almost entirely standard LaTeX code. Examples of more complex tables with more sophisticated multirow/multicolumn behavior and nested tables can be found in the dedicated article.

Table 1: The Periodic Table of the Elements with embedded links to the Wikipedia page for each element.

H																	He
Li	Be											B	C	N	O	F	Ne
Na	Mg											Al	Si	P	S	Cl	Ar
K	Ca		Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr
Rb	Sr		Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe
Cs	Ba	†	Lu	Hf	Ta	W	Re	Os	Ir	Pt	Au	Hg	Tl	Pb	Bi	Po	At	Ra
Fr	Ra	‡	Lr	Rf	Db	Sg	Bh	Hs	Mt	Ds	Rg	Cn	Nh	Fl	Mc	Lv	Ts	Og

†	La	Ce	Pr	Nd	Pm	Sm	Eu	Gd	Tb	Dy	Ho	Er	Tm	Yb
‡	Ac	Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No

Code Highlighting

As a one of the major types of content envisioned for the website is code (indeed it will already be necessary for the next article), it is imperative to have syntax highlighting available for all languages likely to be displayed. Getting this to work in a desirable fashion is not entirely straightforward and unfortunately requires sidestepping a lot of Make4ht's machinery in order to get it to work well. That said, it is still entirely possible using the Minted[4] package for LaTeX, which is based on the Pygments[5] Python library as seen in Listing 1:

Listing 1: One of the snippets of python code which handle the syntax highlighting. Note that the quality of the highlighting is dependent on how well the language is supported in Pygments.

```
0 import sys
1 from pygments import highlight
2 from pygments.lexers import (get_lexer_by_name)
3 from pygments.formatters import HtmlFormatter
4
5 class CodeHtmlFormatter(HtmlFormatter):
6     def wrap(self, source):
7         return self._wrap_code(source, include_div=False)
```

```
8       #
9
10      def _wrap_code(self, source, *, include_div):
11          yield 0, '<pre class=\'' + sys.argv[2] + '\'><code>'
12          for i, t in source:
13              if i == 1:
14                  # it's a line of formatted code
15                  t += '</code><code>'
16              #
17              yield i, t
18          #
19          yield 0, '</code></pre>'
20      #
21  #
22
23  def main():
24      # argv[2] = lexer name
25      # argv[3] = file path
26      file = open(sys.argv[3], "rt")
27      text = file.read()
28      file.close()
29
30      lexer = get_lexer_by_name(sys.argv[2])
31      from pygments.formatters import HtmlFormatter
32      print(highlight(text, lexer, CodeHtmlFormatter()))
33  #
34
35  if __name__ == "__main__":
36      main()
37  #
```

The details of how to use the Minted Package for LaTeX and sidestep it to directly use the Pygments library can be found in its own article here.

Raster and Vector Images

Images are easily one of the most critical features for any website and thus must be handled thoroughly. Fortunately, raster images (images based on pixel data) such as PNGs and JPEGs are handled out-of-the box with Make4ht. This can be seen demonstrated in Figure 1. Note in comparing the HTML and PDF versions of this document that the image is correctly scaled to the width of the column it is in. Animated raster graphics such as GIFs would be a nice addition to this system, but they aren't considered critical for the base function of the website in part due to the complications of embedding them in PDF files.

Vector graphics are, at a base line, not any more difficult than raster graphics to include. It is a simple matter of including the file as one would any other. However, LaTeX also has powerful drawing and graphing capabilities thanks to the PGF and TIKZ packages. These allow the author to draw sophisticated vector graphics directly in the source code of their LaTeX document. While this does also work with vanilla Make4ht, it was found to be somewhat lacking, and thus a workaround was developed to ensure that the PGF and TIKZ drawings end up as SVGs embedded directly within the HTML files themselves.

Figure 1: The Blue Marble. Photo taken by Harrison Schmitt during the Apollo 17 mission. © NASA 1972. Public Domain.

Figures 2 and 3 provide demonstrations of the TIKZ and PGFPlots packages, respectively. Note the ability for elements of the images to change color when swapping between light and dark modes in the HTML version of the document.

A much longer, detailed description of the process to get both types of graphics working can be found here.

Figure 2: The Circle of Fifths for both major and minor keys. Clockwise rotation results in in a sequence of perfect fifths while counterclockwise rotation results in a sequence of perfect fourths. Made using TIKZ.

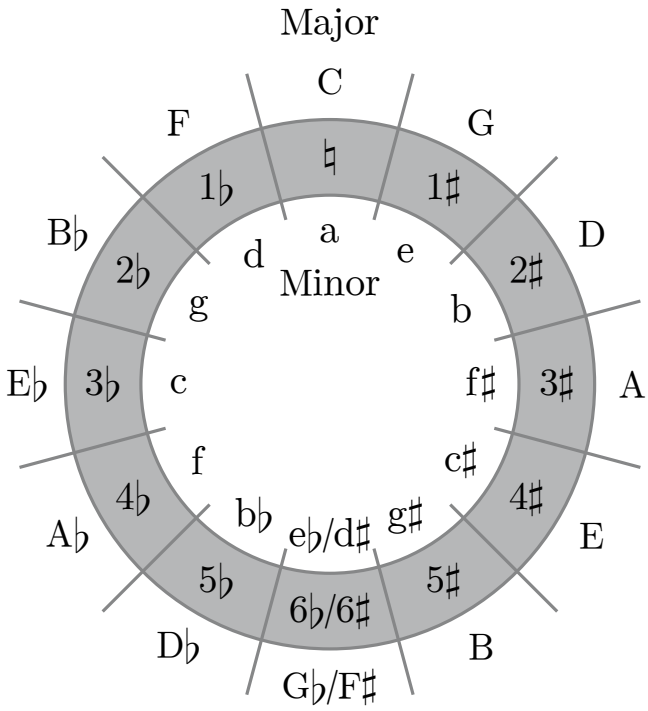
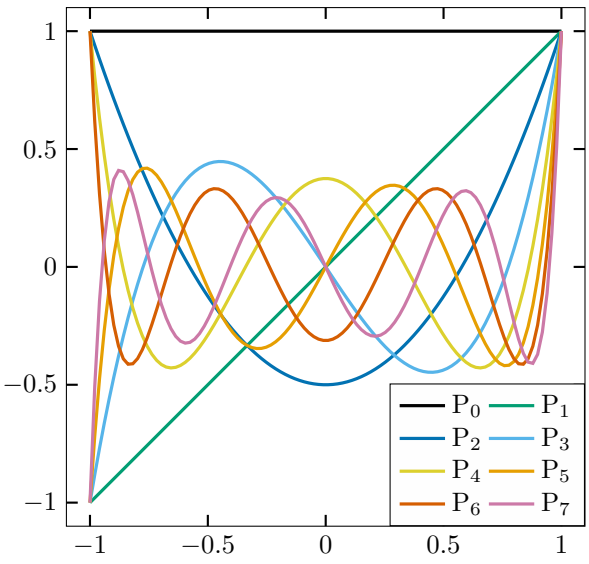


Figure 3: The first eight (up to $n = 7$) Legendre Polynomials on the domain $-1 \leq x \leq 1$. Made Using PGFPlots.



Styling with CSS

With the HTML documents being generated and post-processed into the desired form, it is described here how to set up a CSS file that may be reused across multiple documents. This has two major benefits. First, it allows for style changes to happen in one place instead of across many files and, thus, ensures a consistent theme across an entire website or subset of it. Second, it reduces the size of each HTML document by allowing the reuse of the same styling. This may not be a large consideration in the modern day, but it does save a non-trivial amount of memory when used across many documents. It is also always good to minimize network traffic and respect users' and readers' RAM, if only out of principle.

The linked article discusses at length some details about particular features such as the recent light-dark function of CSS, tool-tips appearing on hover, automatic line numbering for code listings as seen in Listing 1, image scaling, and dynamic vector graphic coloring.

Building a Build System

All of the content discussed thus far has been dedicated to setting up a system that allows for the compilation of a single PDF and HTML file from a TeX file. The tentatively final[E] article in this series, linked here, discusses the development of a build system for an entire website which only recompiles files which have been updated while ensuring that the navigation links in all files stay up to date and well linked.

[E]There are some features which might be included in the future such as GIFs (as previously mentioned), sheet music, audio, and perhaps even proper video.

References

[1] Make4ht. https://www.kodymirus.cz/tex4ht-doc/BasicTutorial.html. Accessed: 2026-01-10.

[2] Mathjax. https://www.mathjax.org/. Accessed: 2026-01-10.

[3] Mathml. https://en.wikipedia.org/wiki/MathML. Accessed: 2026-01-10.

[4] Minted. https://ctan.org/pkg/minted. Accessed: 2026-01-10.

[5] Pygments. https://pygments.org/. Accessed: 2026-01-10.