Author's guide to texedbook

Riley Hanus, PhD

Abstract

The `texedbook` code base is a tool for publishing articles, educational content, and textbooks online without the need to learn html, css, and javascript. The author writes the content following this guide, compiles it using `latexmk`, and after following the set-up instructions in the `README.md` runs

```
python make_texedbook.py ./path/to/latex/project/directory
```

which generates a collection of html and css files in `./output/` that can be directly published online. The product is a computer and mobile friendly webpage, with all document features preserved and all content embedded. In addition to supporting most of the native latex features, `texedbook` provides tools for embedding digital and interactive content such as videos, quizzes, code editors and compilers, etc.

This article acts as an author's guide to `texedbook` (`tex`: Latex based, `ed`: education focused, `book`: classic textbook functionality maintained) and will demonstrate the document features that are explicitly supported, and the specific way they must be used to ensure a clean output html page. First, the native latex capabilities will be demonstrated including figures, equations, tables, cross-referencing, citations, etc. Then the `texedbook` specific features will be presented which allow the author to embed digital content (anything that can be contained in an iframe) into the resulting webpage straight from the latex document.

1 Motivation for texedbook project

When writing anything technical in nature, the need for proper writing tools is glaring. Without a framework to efficiently manage citations, cross-reference document items (e.g. sections, equations, figures), write math, etc., writing anything with technical substance becomes prohibitively difficult. Latex, despite its quirks, is a very good framework to manage these critical writing tools and compiling beautifully typeset pdf documents.

At the same time, publishing and education has transitioned to a digital-first experience for the consumer, yet publishers still cling to a print-first model. There is a clear need for a tool enabling authors to publish digital-first articles, books, and courses. The natural medium for this digital-first publishing is an html, css, and javascript based webpage.

`texedbook` combines the strengths of latex for authoring articles and books with the versatility and universality of an interactive webpage.

2 Required file naming and structure

`texedbook` requires specific file naming and those files must live in specific locations within your project directory. This ensures proper handling of figures and that the html code generated from your project can be templated properly into your final result.

Specifically, the main latex document must be called `main.tex` and must be in the base of your project directory. `texedbook_envs.tex` contains required commands and environments, and best practice is to leave it right next to `main.tex`. In addition, if your project has references, it is best practice to leave `references.bib` next to `main.tex` as well. You may use `\input{doc1}` in the preamble and `\include{doc2}` in the document as usual. However, they must be called as `\include{doc2}`, not `\include{doc2.tex}` since will tell the `latexpand` command (native to the full latex installation and used here) to look for `doc2.tex.tex`. Any figures your project may have must be in pdf format and must live in a `figures/` directory which itself lives next to `main.tex`.

```
main-project-directory/
    main.tex (required)
    texedbook_envs.tex (required)
    doc1.tex
    doc2.tex
    references.bib
    figures/ (required)
        figure-1.pdf
```

3 Native latex features

This section will provide a non-exhaustive documentation of the native latex features that are supported by `texedbook`. It will also provide their specific usage that yields a clean html output.

3.1 Figures

All figures used in the document must be contained in a directory named `./figures/` and that folder must be located in the same location as `main.tex`. In addition, all figures must be in pdf format.

Ensuring that figures are displayed correctly in the resulting html is tricky. The simplest and most effective way is to assign that **all figures span width of the text on the page**. This display condition reliably converts from latex to html and translates from computer to mobile formats well, which is not the case for all display options. Since the aspect ratio of figures is locked by design,

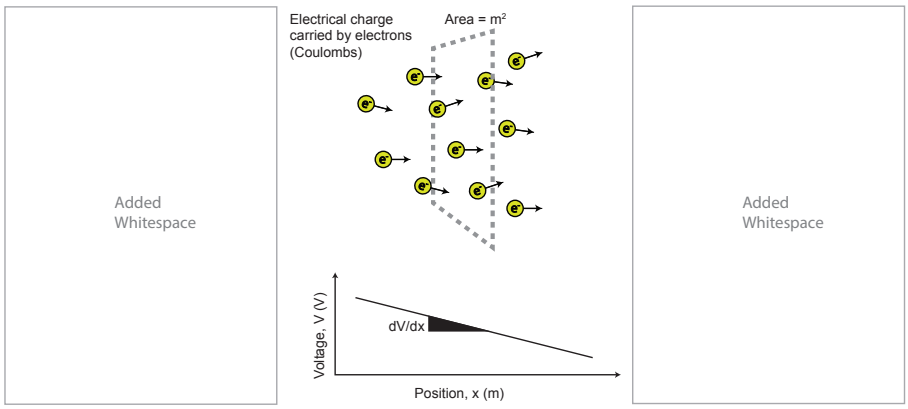

Figure 1: Example figure. Since the convention for texedbook is for figures to span the text width, whitespace added to control size of the image.

this may cause your figure to display very large in the pdf and html. If you would like the figure to display smaller, simply add white space to the left and right of the pdf as is done in Figure 1.

An example of the latex code used to add a figure is below.

```
\begin{figure}[t]
    \centering
    \includegraphics[width=0.99\textwidth, keepaspectratio]{figures/fig.pdf}
    \caption{Example figure.}
    \label{fig:example}
\end{figure}
```

The `[t]` places the figure at the top of a page in the pdf, and does nothing to the html output. The html output will place the figure where it appears in the code. The `\centering` center justifies the figure. The `\includegraphics[]{}` adds in the figure, the `width=0.99\textwidth` sets the width of the figure to 99% of the text width, and the `figures/fig.pdf` points to the figure which must be placed in a `./figures/` directory and must be pdf format. The `\label` allows the figure to be cross-referenced.

3.2 Equations

Equations are an inherently tricky problem for digital publishing. The core of the problem lies in the fact that html was designed around the standard alphanumeric alphabet, and math requires a wider range of complex symbols and typesetting. The default usage of `texedbook` leverages `mathjax`, allowing all the native latex equations to be reliably reproduced in the html output.

In line equations can be included $n\lambda = 2d\sin\theta$, as well as displayed equations

$$n\lambda = 2d\sin\theta. \tag{1}$$

Table 1: Example table.

Sample	Data 1	Data 2	Data 3
A	53	62	75
B	51	61	72
C	58	69	71

Cross-referencing equations is done using the `\mjref{}` function provided in `texedbook_envs.tex`. For example, Eq. 1 is Bragg's Law which is commonly used to find diffraction conditions of light of wavelength λ interacting with a crystal with lattice spacing d at an incident angle of θ. The `\mjref{}` function simply substitutes `\ref{}` when the latex is compiled. However, when `tex4ebook` converts the project to html, the `aux/config.cfg` file tells `tex4ebook` to insert `\eqref` in place of all `\mjref`s, which `mathjax` knows to look for in the html and use for equation numbering and cross-referencing.

3.3 Tables

Tables are inherently difficult to write in latex. Tables Generator is a great tool for generating latex code for tables. Note that the table styling in the html output is different then the latex output.

An example of the latex syntax for including a table is given below.

```
\begin{table}[t]
    \caption{Example table.}
    \centering
    \begin{tabular}{|c|c|...}
    ...
    \end{tabular}
    \label{table:example}
\end{table}
```

The `[t]` places the table at the top of a page in the pdf, and does nothing to the html output. The html output will place the table where it appears in the code. The `\centering` center justifies the table. The table syntax goes within `\tabular` environment. The `\label` allows the table to be cross-referenced.

3.4 Code Syntax

Code syntax can be displayed using the verbatim environment native to latex

```
\begin{verbatim}
    for i in list:
        print('some code')
\end{verbatim }
```

This will render in the html as a displayed code block that scrolls if a text overflow happens. This way readers will never miss the syntax they need, no matter the size of thier screen.

Code can be written in-line using the `\verb` function. This function comes from the `verbatim` package and is unique in that it doesn't use the curly brackets for its argument. As implemented here, it cannot be used as an argurment in most commands such as, `\section{}` or `\caption{}`. It takes the first non-letter character as the open argument and looks for that same character again to close the argument. I typically use `\verb'code here'`. This is meant for short (one word) expressions; if the content is too long it will be cut off. For long syntax use the `\begin{verbatim}` environment.

3.5 Cross-referencing and hyperlinks

Cross-referencing can be used as normal in latex and all the hyperlinking functionality will be preserved in the html. For example, Section 3, Table 1, and Figure 1. The only important exception is Equations where the `\mjref` command must be used such that mathjax can properly label and cross-reference equations locally in the html page using javascript (see Section 3.2).

Hyperlinks to external webpages can be added in latex using the `\href` command.

```
\href{https://www.webpage.com}{Display name in document}
```

3.6 Citations

The native bibliography features are maintained. For example, one can cite an article [1] or multiple articles [2, 1, 3]. Note that the hyperlinking to the bibliography in the pdf, a result of using the `hyperref` package, is maintained in the resulting html. The default bibliography style used in `texedbook` is `unsrturl`. Note that this style supports hyperlinking to the url, and/or doi if that field is present in the .bib entry, and the hyperlinking is maintained in the resulting html. It is recommended that the doi field is used instead of the url, since line breaking needed to typeset urls is handled poorly by latex. Latex's performance on typesetting doi hyperlinks seems to work more reliably.

4 Texedbook features

This section will demonstrate special `texedbook` environments and commands that allow the author to embed digital content directly from the latex code into the resulting webpage. There are several commands provided in `texedbook_envs.tex` to achieve this. `\InsertIframe` is specifically for iframes (explained in Section 4.1) and `\InsertHTML` is more generic and allows the author to include any html element they wish. The arguements the author inputs tells `texedbook` what to render in the pdf as well as what to template into the output html.

In practice, `texedbook` searches the latex project for the specific pattern `\InsertXXX{...}` and grabs the arguements needed to template the html into the final webpage. Therefore, you will see an extra space before the open curly bracket when displaying the latex code syntax for these functions in this document. This avoids unintentionally triggering the templating functions in `make_texedbook.py`, and insures that this Author's Guide works properly. The author should also **avoid unintentionally triggering the templating functions by writing the exact synax for these commands in places it doesn't belong (e.g. in comments, verbatim enviroments, etc.).**

4.1 Iframes

An inline frame, or iframe, is an html element that loads a webpage within another webpage. When you see a YouTube video on a website other than www.youtube.com, it is likely being placed their using an iframe. However, as you can infer from the definition, iframes are extremely versatile and can be used for far more than embedding YouTube videos. Essentially, anything online can be embedded using an iframe. Often times there will be a *share* button next to a sharable feature on a webpage, and one of the sharing options can be *embed*, or have the `<>` symbol. This option will provide a piece of html code for the iframe that can be copy and pasted. An example of an html iframe is given below.

```
<iframe width="560" height="315"
        src="https://www.youtube.com/embed/..."
        title="YouTube video player" ...></iframe>
```

The `\InsertIframe` command provided in `texedbook_envs.tex` allows the author to embed any iframe into the output html, directly from the latex project. The pdf will display a *digital content box* containing a hyperlink to the source URL contained in the iframe code. The output html will contain the iframe properly embedded. There are four arguments for `\InsertIframe`, and they are detailed here.

```
\InsertIframe {<iframe ...></iframe>}
              {hyperlink display text}
              {caption}
              {type of digital content}
```

Below is this command in practice, where a YouTube video from MIT's OpenCourseWare, a Google Form quiz, and a Spreadsheet are all embedded. As you can see in the compiled pdf, a fancy *Digital Content* box and hyperlink are rendered. In the html output, the iframe is properly embedded.

Digital Content : Video

Intro to Copyright Law | MIT OpenCourseWare 6.912, Lecture 1

Digital Content : Google Form

Example Quiz | Test your understanding of texedbook

Digital Content : Spreadsheet

Google Sheet | GOOG stock price summary

4.2 Special embed commands

There also may be scenerios when the embed code is not provided. This is the case for videos hosted on the course management platform Panopto. In this case simply replacing the 'Video' with 'Embed' in the url and placing it in a generic <iframe> element embeds the video properly. Therefore, special commands are provided which generates the <iframe> for the author with the following syntax. See all the special commands provided in `texedbook_envs.tex`. An example for a Panopto video is given here.

```
\InsertPanoptoVideo {https://www.url-to-video.com/xxx}
                    {hyperlink display text}
                    {caption}
```

Below is this command in practice, where a lecture from Prof. G. Jeff Snyder's course on electronic and thermal transport at Northwestern University is embedded. As you can see in the compiled pdf, a fancy *Digital Content* box and hyperlink are rendered. In the html output, the iframe is properly embedded.

Digital Content : Video

MSE 485 2-11 | Prof. G. Jeff Snyder's lecture on Complex Fermi Surfaces

4.3 Generic HTML

The author may desire more flexibility. The `\IncludeHTML` command is quite generic and versitile. Its first argument is simply the html element that will be templated directly into the output html code. Its second arguement is the latex code that will be compiled into the pdf.

```
\InsertHTML {html code for output (e.g. <div ...></div>)}
            {latex code for compiled pdf}
```

See this command implemented below where another YouTube video is embedded in the output html, and a simple hyperlink is compiled in the pdf. Three Blue One Brown video on the Fourier Transform.

References

[1] Riley Hanus, Ramya Gurunathan, Lucas Lindsay, Matthias T. Agne, Jingjing Shi, Samuel Graham, and G. Jeffrey Snyder. Thermal transport in defective and disordered materials. *Applied Physics Reviews*, 8(3):031311, 08 2021. doi:10.1063/5.0055593.

[2] Riley Hanus, Matthias T. Agne, Alexander J. E. Rettie, Zhiwei Chen, Gangjian Tan, Duck Young Chung, Mercouri G. Kanatzidis, Yanzhong Pei, Peter W. Voorhees, and G. Jeffrey Snyder. Lattice softening significantly reduces thermal conductivity and leads to high thermoelectric efficiency. *Advanced Materials*, 31(21):1900108, 2019. doi:https://doi.org/10.1002/adma.201900108.

[3] Shawn A. Gregory, Riley Hanus, Amalie Atassi, Joshua M. Rinehart, Jamie P. Wooding, Akanksha K. Menon, Mark D. Losego, G. Jeffery Snyder, and Shannon K. Yee. Quantifying charge carrier localization in chemically doped semiconducting polymers. *Nature Materials*, 20(10):1414–1421, 2021. doi:10.1038/s41563-021-01008-0.