MiniLaTeX

James Carlson

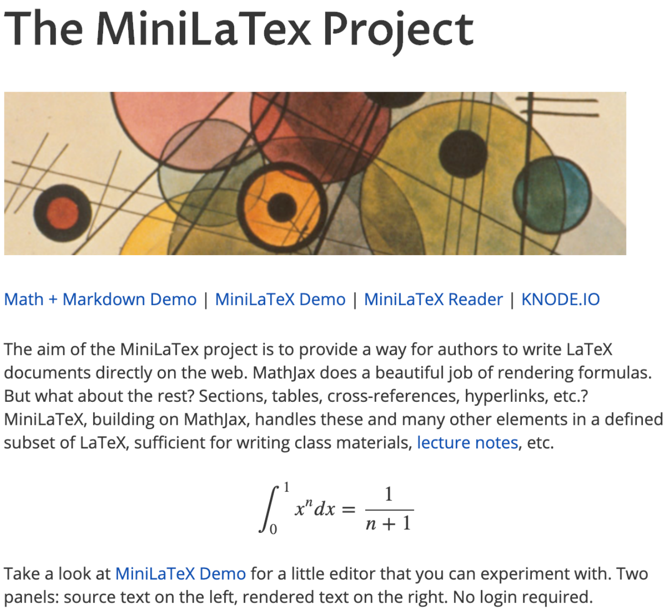
July 26, 2020

Contents

Abstract

MiniLateX is a subset of LaTeX that can be rendered on the fly to HTML. With it, one can build web apps with true HTML display that can be viewed on any device from smart phone to tablet to desktop. Typesetting occurs in real time, and error messages are displayed in-line in the rendered text. MiniLaTeX documents can be exported to standard LaTeX. We describe the main features of the MiniLaTeX application minilatex.lamdera.app, then describe some the technical work required to implement an on-the-fly LaTeX-to-Html compiler. This is done in Elm, a strongly typed language of pure functions that is designed for building web apps.

Website. minilatex.io.



The MiniLaTex Project

Math + Markdown Demo | MiniLaTeX Demo | MiniLaTeX Reader | KNODE.IO

The aim of the MiniLaTex project is to provide a way for authors to write LaTeX documents directly on the web. MathJax does a beautiful job of rendering formulas. But what about the rest? Sections, tables, cross-references, hyperlinks, etc.? MiniLaTeX, building on MathJax, handles these and many other elements in a defined subset of LaTeX, sufficient for writing class materials, lecture notes, etc.

$$\int_0^1 x^n dx = \frac{1}{n+1}$$

Take a look at MiniLaTeX Demo for a little editor that you can experiment with. Two panels: source text on the left, rendered text on the right. No login required.

minilatex.io

1 Introduction

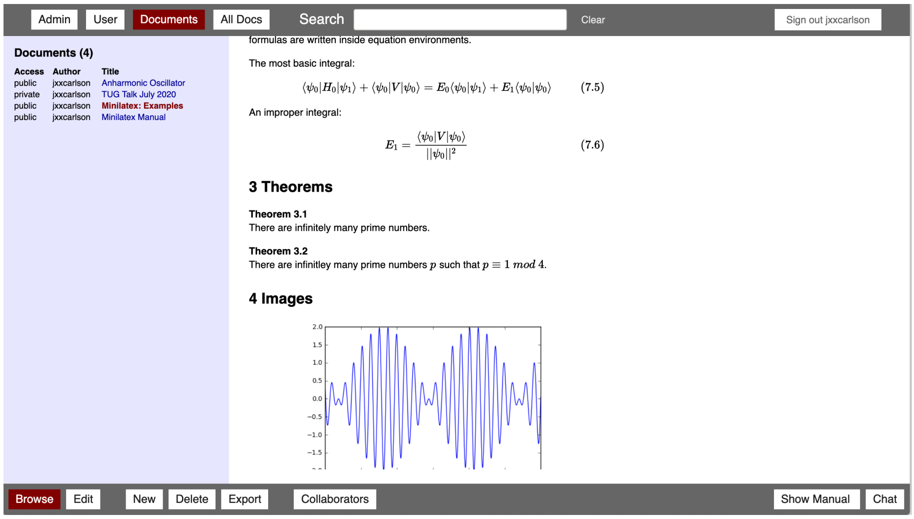
MiniLaTeX refers both t MiniLaTeX as a subset of LaTeX, and also to a system for managing documents written in MiniLaTeX:

1.1 Subset of LaTeX

A subset which is well-defined, has a grammar, and is small enough for one human to write an efficient parser by hand, yet large enough to be useful (lecture notes, problem sets, small articles, just learning LaTeX).

MiniLaTeX apps

- Example of class notes: knode.io/424
- Read-only app for distributing content: reader.minilatex.app
- Simple no-signin app for learning LaTeX: demo.minilatex.app



Documents (4)

Access	Author	Title
public	jxxcarlson	Anharmonic Oscillator
private	jxxcarlson	TUG Talk July 2020
public	jxxcarlson	**Minilatex: Examples**
public	jxxcarlson	Minilatex Manual

formulas are written inside equation environments.

The most basic integral:

$$\langle \psi_0 | H_0 | \psi_1 \rangle + \langle \psi_0 | V | \psi_0 \rangle = E_0 \langle \psi_0 | \psi_1 \rangle + E_1 \langle \psi_0 | \psi_0 \rangle \tag{7.5}$$

An improper integral:

$$E_1 = \frac{\langle \psi_0 | V | \psi_0 \rangle}{||\psi_0||^2} \tag{7.6}$$

3 Theorems

Theorem 3.1
There are infinitely many prime numbers.

Theorem 3.2
There are infinitley many prime numbers p such that $p \equiv 1 \; mod \; 4$.

4 Images

minilatex.lamdera.app

1.2 Newly released app

Minilatex.lamdera.app was released for alpha testing on July 18, 2020.

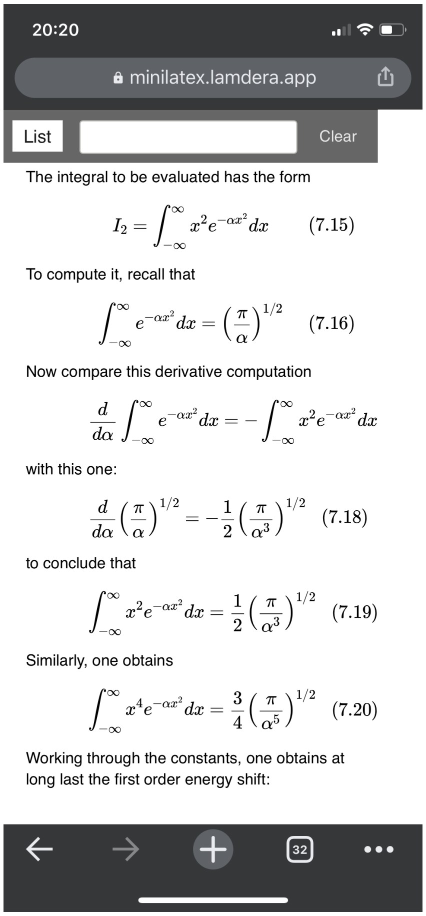
- **Guest access.** Sign in as guest with password minilatex to explore documents that have been made public by their authors. Both the source and rendered text are available, so you can see how MiniLaTeX documents are written.
- **User on any device.** Create and edit documents on a dektoop computer or tablet. Works in read-only mode on smart phones.
- **Accounts.** Sign up and establish a free account if you want to create MiniLaTeX documents, or just experiment.

1.3 Anywhere, any device

As a web app, https://minilatex.lamdera.app is available anywhere there is an internet connection. Just sign in and start working.

On your computer **On your phone** On a smart phone, the app displays public

documents only. They can be read, but not created or edited. There are two modes. In the first, a list of documents is presented If you click on a title in the list, the document is displayed. To display the document list again, click on the List button in the header. Use the search bar in the header to display fewer documents. For example, if you put minilatex or just mini, only documents with MiniLaTeX in the title are shown.



The integral to be evaluated has the form

$$I_2 = \int_{-\infty}^{\infty} x^2 e^{-\alpha x^2} dx \qquad (7.15)$$

To compute it, recall that

$$\int_{-\infty}^{\infty} e^{-\alpha x^2} dx = \left(\frac{\pi}{\alpha}\right)^{1/2} \qquad (7.16)$$

Now compare this derivative computation

$$\frac{d}{d\alpha} \int_{-\infty}^{\infty} e^{-\alpha x^2} dx = -\int_{-\infty}^{\infty} x^2 e^{-\alpha x^2} dx$$

with this one:

$$\frac{d}{d\alpha} \left(\frac{\pi}{\alpha}\right)^{1/2} = -\frac{1}{2}\left(\frac{\pi}{\alpha^3}\right)^{1/2} \qquad (7.18)$$

to conclude that

$$\int_{-\infty}^{\infty} x^2 e^{-\alpha x^2} dx = \frac{1}{2}\left(\frac{\pi}{\alpha^3}\right)^{1/2} \qquad (7.19)$$

Similarly, one obtains

$$\int_{-\infty}^{\infty} x^4 e^{-\alpha x^2} dx = \frac{3}{4}\left(\frac{\pi}{\alpha^5}\right)^{1/2} \qquad (7.20)$$

Working through the constants, one obtains at long last the first order energy shift:

1.4 Real-time typesetting

Changes to the rendered text of your documents are displayed as you type, as are any error messages. The. error messages are displayed in the rendered text. While editing, only parts of the text which have changed are recompiled. The advantage of this strategy is that recompilation is very fast. The disadvantage is that things like tables of contents, cross-references, etc., can get out of sync. To put things back in sync, click on the Full Render button ion the footer.

NOTE. MiniLaTeX is paragraph-centric, meaning that the smallest unit of recompilation is the *logical paragraph*. A logical paragraph is either an ordinary paragraph or an outer begin-end block. For this reason, it is good practice to use plenty of white space in MiniLaTeX documents: a blank line before and after defines a logical paragraph. The source text of your document will be easier to read, and the compiler will better do its work.

1.5 Export to Standard LaTeX

Use the Export button in the footer. The exported document will be saved in the Downloads folder unless you tell your computer otherwise. If your document has images, a button Images will appear to the right of Export. Click on that button to bring up a window that lists the images in your document. Right-click on these images to download them. They should be stored in folder image in the same place as your exported document. Below are two links exported documents:

- MiniLaTeX: Examples
- Anharmonic Oscillator
- This document

2 Features

No setup. MiniLaTeX documents require no setup: no preamble, no \usepackage, no \begin {document} etc. Just start typing. On the otherand, all of the relevant boilerplate is present in exported documents so that they can be processed in the way that you woiuld normally process a LaTeX document.

Live typesetting, error messages. Source text is typeset and rendered on the fly, as demonstrated in this VIDEO. If the user makes syntax errors (or if an expression cannot be parsed because it is incomplete), a color-coded error message is displayed in the rendered text. We are working to further improve error messages and error handling.

Documents are saved at intervals of roughly 400 milliseconds while editing.

Images. You can place an image in a MiniLaTeX document if it it exists somewhere on the internet and you have its "limage ocation" or URL. You can usuallly find the location/URL of an image that you see in a browser by right-clcking on it. The URL for the image displayed below is

```
https://psurl.s3.amazonaws.com/images/jc/beats-eca1.png
```

Place an image like this:

```
\image{URL}{Caption}{Format}
```

The caption and format are optional. The format has two parts, both of which are optional: the width and the placement. The width is a fragment like width: 350. Placement can be one of the following:

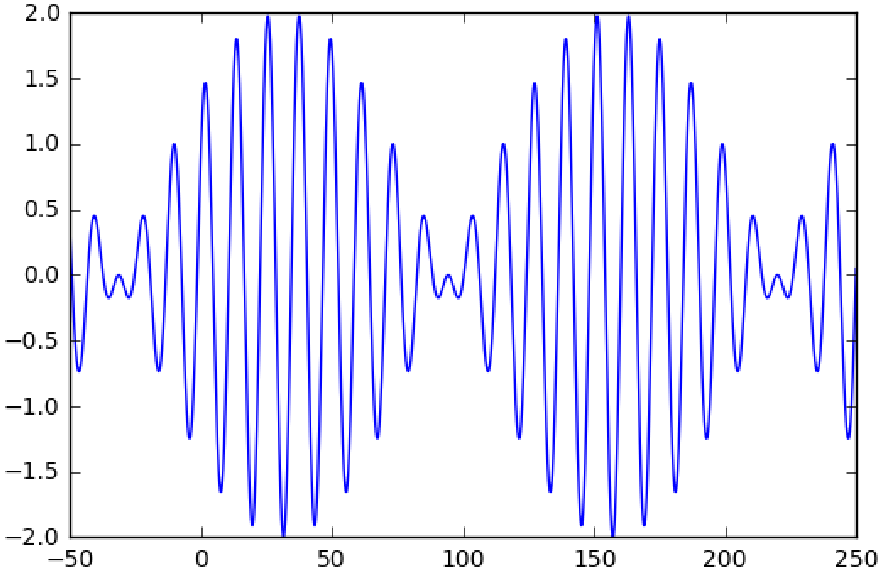


Figure 1. Two-frequency beats

```
align: center
float: left
float: right
```

SVG Images SVG images can be placed like this:

```
\begin{svg}
CODE FOR SVG IMAGE
\end{svg}
```

In the example below, the SVG code was produced by a graphics program. You can also produce SVG code by hand, or by writing a program.

Cannot yet render Svg images; convert to some other format, e.g., png When a MiniLaTeX document is exported, a placeholder is put in the place of the SVG image. The TeX processors I am a aware of require png, pdf, etc., so you will have to ocnvert the SVG image to one of those formats.

Collaboration. To add a collaborator to a document, click on the Collaborators button in the footer. Enter collaborators in the resulting popup window one collaborator per line. Collaborators may edit a document synchronoulsy or asynchronously. In the former case, when both authors are online, changes by one author are reflected a fraction of a second later in the other others app. To facilitate collaboration, there is a chat window (lower right, in footer). The chat feature needs a good deal more work.

3 The MiniLaTeX compiler

LaTeX documents consist of snippets of math-mode LaTeX (the equations and formulas) and text-mode LaTeX (everything else, e.g. section headings, lists, and environments of various knds). The strategy that makes writing a compiler that transforms MiniLaTeX to HTML feasible is divide and conquer. Parse the the document, identifying the math elements as such, but not transforming them. Then render the resulting abstract syntax tree (AST) to Html, again passing on the math elements. Finally, have MathJax render the math elements. One can also use KaTeX.

An abstract syntax tree is a tree of things that expresses a grammatical analysis of the source text. It is akin to what one gets when one diagrams a sentence in English class, something the present author did in Marge Lee's sophomore class in high school. More on this later.

The MiniLaTeX compiler consists of two parts, a *parser* and a *renderer*. The first is a function

$$parse : \text{Source text} \to \text{AST}$$

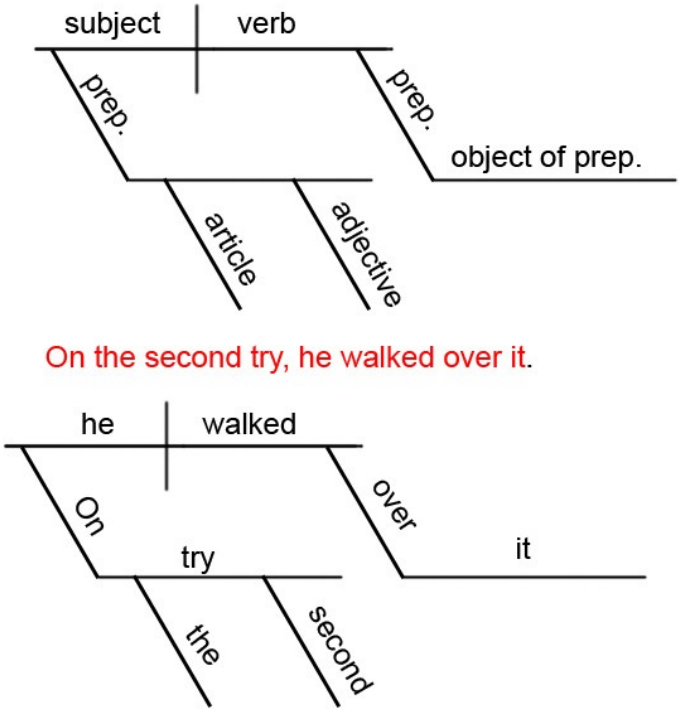
The second is a function

$$render : \text{AST} \to \text{HTML}$$

Here we are simplifying somewhat because of various optimizations that are needed, but this is the general idea. The compiler is the composite of these two functions:

$$compile = render \circ parse$$

In designing a parser, one generally starts with a grammar. However, there is no published grammar for LaTeX or for TeX, so one has to improvise. The approach taken with MiniLaTeX was first to imagine the type of the AST, then to write a parser using parser combinators. The parser then consists of a bunch of functions which call one another. They are in rough correspondence with the productions of a grammar, and so one can write down a grammar after the fact. Ideally one should have written the grammar, then the parser, but real life is often messy.



On the second try, he walked over it.

Diagramming sentences

3.1 AST

Every value in a statically typed language like Haskell, ML, or Elm, has a *type*. Below is the type of AST for the MiniLaTeX compiler.

```
type LatexExpr
    = LXString String
    | Comment String
    | Item Int LatexExpression
    | InlineMath String
    | DisplayMath String
    | SMacro String (List LatexExpr) (List LatexExpr) LatexExpr
    | Macro String (List LatexExpr) (List LatexExpr)
    | Environment String (List LatexExpr) LatexExpr
    | LatexList (List LatexExpr)
    | NewCommand String Int LatexExpr
    | LXError (List (DeadEnd Context Problem))
```

Below is a series of examples that give an idea of how the parser works. The examples are generated using a command-line tool that applies the parser to a string of text supplied by the user.

```
> Pythagoras
LXString "Pythagoras"

> \strong{Pythagoras}
Macro "strong" [] [LatexList [LXString "Pythagoras"]]

> \strong{Pythagoras} says that $a^2 + b^2 = c^2$
Macro "strong" [] [LatexList [LXString "Pythagoras"]]
  , LXString "says  that "
  , InlineMath "a^2 + b^2 = c^2"
```

Below is the top-level parser function. Note the close correspondence with the AST type. The `lazy` function is needed for recursion. The `oneOf` function is a combinator that takes a list of parrsers and produces a new parser from them. The resulting parser operates as follows: try the first parser. If it succeeds, return the result. If not, try the next parser. And so on. If the list is exhausted without success, return an error.

```
latexExpression : LXParser LatexExpr
latexExpression =
    oneOf
        [ texComment
        , displayMathDollar
        , displayMathBrackets
        , inlineMath
        , newcommand
        , macro
        , smacro
        , words
        , lazy (\_ -> environment)
        ]
```

Below is the parser for macros. It uses a *parser pipeline* in the following way. First, find the name of the macro, then the list of optional arguments, then the list of actual arguments, then eat any whitespace (spaces, newlines). Second, feed the values found in the order found to the constructor `Macro` for the `LatexExpr` type. In this example, `whitespace` is a parser for white space, which can come in different flavors depending on context e.g., spaces only or spaces and newlines.

```
macro : LXParser () -> LXParser LatexExpr
macro =
    succeed Macro
        |= macroName
        |= itemList optionalArg
        |= itemList arg
        |. whitespace
```

One can continue down the rabbit hole, explaining the parsers and `macroName`, `optionalArg`, `arg` and the combinators `itemList`, etc, but we stop here. What is important to understand is that there are really only three things in something like the MiniLaTeX parser: primitive parsers, combinators that choose among alternatives, and combinators that sequence other parsers.

3.2 Rendering

The top-level renderiing function is much like the top-level parsing functions. It analyzes the type of a LatexExpr and dispatches the appropriater renderer, which may in turn call other rendering functions, including the top level one. And so on, down the rabbit hole of function calls we go.

```
render : String -> LatexState -> LatexExpr -> Html msg
render source latexState latexExpr =
    case latexExpression of
        Comment str ->
            Html.p [] [ Html.text "" ]

        Macro name optArgs args ->
            renderMacro source latexState name optArgs args

        Item level latexExpr ->
            renderItem source latexState level latexExpr

        InlineMath str ->
            Html.span [] [
              inlineMathText
                latexState
               (evalStr  latexState.macroDict str) ]

        DisplayMath str ->
            Html.span [] [
              displaMathText
                latexState
               (evalStr  latexState.macroDict str) ]

        Environment name args body ->
            renderEnvironment source latexState name args body

        LatexList latexList ->
            renderLatexList source latexState latexList
```

3.3 Code

Code for the MiniLaTeX compiler is at github.com/jxxcarlson/meenylatex. The strange name is to reserve the name github.com/jxxcarlson/minylatex for a future stable version with a polished API. The repository just mentioned also contains simple demos t show how to use the compiler.

The code for the minilatex.lamdera.app is at github.com/jxxcarlson/lamdera-minilatex-app.

All code is open source. The web application minilatex.lamdera.app is offered as a service. It is free for now, and the intent is to (a) keep it free for students, (b) eventually charge a small annual fee for other users. Bills have to be paid and code has to be maintained! Note that guests may browse the site without establishing an account.

This document was written in MiniLaTeX and is avaiable at https://minilatex.lamdera.app with title *TUG Talk: MiniLaTeX*

4 Plans

I am very interested in feedback from the community regarding features, bugs, etc. Of special interest is: what should the subset of LaTeX be. Below are some ideas.

1. An image uploader. With this in place, you can post images on a server and use the resulting URL to place images in your MiniLaTeX documents.
2. Address documents by URL.
3. Better error messages
4. An IDE/editor with sync of source and rendered text, syntax highlighting, etc.
5. There will eventually be a small annual fee (unless you are a student) so that we can sustain the project over the long term. If you decide against the annual fee, your content will remain on the site. Your account can be re-activated at any time.

5 Acknowledgements

I would like to thank Evan Czaplicki, Ilias Van Peer, Mario Rogic, and Luke Westby, all of the Elm community, and Davide Cervone, MathJax.org, for their generous and invaluable help. I also wish to thank the Simons Foundation for its support of this project.