Getting Started with PreTeXt

Virtual Workshop 2023

Getting Started with PreTeXt

Virtual Workshop 2023

Steven Clontz
University of South Alabama

Oscar Levin
University of Northern Colorado

Summer 2023

Preface

PreTeXt[1] is a document authoring system that allows you to convert the source of your document into a variety of output formats, including fully accessible webpages, PDF, Epub, Jupyter Notebooks, and braille. This "write once, read anywhere" approach has made it a popular choice for authors of Open Educational Resources, but PreTeXt can also be used to create other kinds of mathematical documents as well. Recent updates make this process much easier; there has never been a better time to get started with PreTeXt.

Participants of this workshop will be introduced to the fundamentals of authoring documents with PreTeXt and gain the technical skills required to work with it. Specifically, participants will learn how to:

- Use GitHub Codespaces to create an editable PreTeXt document in their web browser.
- Write and structure content using PreTeXt markup.
- Add content to the document, including mathematics, graphics, interactive exercises, and more.
- Build accessible and interactive webpages as well as a static PDF from the same PreTeXt source.
- Easily deploy the interactive webpages online (for free).

We will also share tips for converting existing documents into PreTeXt.

Prior to the workshop, participants should have some familiarity with LaTeX or other markup languages. No previous experience working with PreTeXt or HTML/XML is assumed. Participants should bring a laptop that can connect to JMM's provided wifi: no prior installations will be required as we will use PreTeXt's new GitHub Codespace-powered online authoring service.

[1] `pretextbook.org`

Contents

Chapter 1

Before you arrive...

Authoring in PreTeXt requires nothing more than a wifi connection and GitHub account. GitHub users also can share their content for free on GitHub Pages!

1.1 Setting up your GitHub account

To create a GitHub account, follow the instructions on GitHub's signup page.[1]

Be sure to note your GitHub username and password in your password manager (or however you usually keep track of login credentials) so you can log in again durring the workshop.

1.2 Apply for your GitHub Education discount

Educators and non-profit researchers can get many of GitHub's paid features for free.

Apply at Education.GitHub.com[1] to unlock these features (in our experience, applications are usually processed quickly for `.edu` email addresses).

1.3 And that's it!

Even just a last year, the process to get started writing PreTeXt involved several more steps. (Raise your hand if you don't care what a "PATH variable" is!)

The community is streamlining this experience daily, and we look forward to sharing how easy writing in PreTeXt is when we see you at the workshop

[1] `github.com/signup`

[1] `education.github.com/discount_requests/pack_application`

Chapter 2

Write Once, Read Anywhere

Objectives

At the end of this chapter, you'll

1. Become aware of the eleven PreTeXt Principles.
2. Be able to identify several features of PreTeXt.
3. Have a working GitHub Codespaces environment to suitable for authoring and editing in PreTeXt.

2.1 Setting up Codespaces

A **Codespace** is an authoring environment that lives in the "cloud", that is, a virtual machine hosted by GitHub that has all of the software needed to create great accessible documents, accessible with just a web browser.

This coding environment uses a web version of Virtual Studio Code, an open-source editor, along with the PreTeXt community's custom plugins and software to get started authoring quickly.

Follow the instructions at https://github.com/PreTeXtBook/pretext-codespace to get started. Let this run for a few minutes in the background while you review the rest of this section.

Note 2.1.1 Codespaces works best in the Chrome or Edge web browsers.

2.2 PreTeXt Principles

A more detailed monograph on PreTeXt's philosophy[1] is availabe in the PreTeXt Guide.

List 2.2.1 PreTeXt Principles

1. PreTeXt is a markup language that captures the structure of textbooks and research papers.
2. PreTeXt is human-readable and human-writable.
3. PreTeXt documents serve as a single source which can be easily

[1] pretextbook.org/doc/guide/html/philosophy.html

converted to multiple other formats, current and future.

4. PreTeXt respects the good design practices which have been developed over the past centuries.
5. PreTeXt makes it easy for authors to implement features which are both common and reasonable.
6. PreTeXt supports online documents which make use of the full capabilities of the Web.
7. PreTeXt output is styled by selecting from a list of available templates, relieving the author of the burden involved in micromanaging the output format.
8. PreTeXt is free: the software is available at no cost, with an open license. The use of PreTeXt does not impose any constraints on documents prepared with the system.
9. PreTeXt is not a closed system: documents can be converted to LaTeX and then developed using standard LaTeX tools.
10. PreTeXt recognizes that scholarly documents involve the interaction of authors, publishers, scholars, curators, instructors, students, and readers, with each group having its own needs and goals.
11. PreTeXt recognizes the inherent value in producing material that is accessible to everyone.

2.3 PreTeXt is XML

Since PreTeXt uses the XML markup language, all content is structured in terms of **elements**. The root `pretext` element nests many other elements inside of it. This is accomplished by surrounding everything with a starting `<pretext>` **tag** and an ending `</pretext>` tag. (Folks with HTML experience will find this pattern familiar, akin to the "HTML" root element.)

Listing 2.3.1 is a very simple PreTeXt/XML document. (The first line is boilerplate that lets various programs know the rest of the file is XML, and the third-to-last line is an example of a comment that won't appear in the output.)

```
<?xml version="1.0" encoding="UTF-8"?>
<pretext>
    <article>
        <title>Hello world!</title>
        <p>Welcome to PreTeXt!</p>
        <!-- TODO: find something more to say... -->
    </article>
</pretext>
```

Listing 2.3.1 Source of a simple PreTeXt book project.

2.4 Books and Divisions

There are several documents you can write in PreTeXt, such as `<article>`s and `<slideshow>`s. This tutorial will focus on `<book>`s.

A <book> typically includes <frontmatter>, and <backmatter>.

Between <frontmatter>, and <backmatter> are either several <chapter>s or <part>s. If used, <part>s are subdivided into <chapter>s. Then <chapter>s subdivide into <section>s, and <section>s can have <subsection>s.

Each of these subdivisions needs a <title>, and may have an <introduction> or <conclusion>.

Listing 2.4.1 puts some of these elements together for a simple PreTeXt project (information on the other elements will come in later sections).

```
<?xml version="1.0" encoding="UTF-8"?>
<pretext xml:lang="en-US">
  <!-- (author configurations go in docinfo) -->
  <docinfo>
    <macros>
      \newcommand{\R}{\mathbb R}
    </macros>
  </docinfo>


  <book xml:id="my-great-book">
    <title>My Great Book</title>
    <subtitle>An example to get you started</subtitle>

    <frontmatter xml:id="frontmatter">
      <titlepage>
        <author>
          <personname>You</personname>
          <department>Your department</department>
          <institution>Your institution</institution>
        </author>
        <date>
          <today />
        </date>
      </titlepage>
    </frontmatter>

    <chapter xml:id="chapter-welcome">
      <title>Welcome!</title>
      <introduction>
        <p>This chapter is about the real numbers
            <m>\R</m></p>
      </introduction>

      <section xml:id="section-getting-started">
        <title>Let's get started</title>
        <p>Can you solve <m>ax^2+bx+c=0</m>?</p>
      </section>

      <section xml:id="section-learning-more">
        <title>But wait, there's more!</title>
        <p>Did you know that
            <me>x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}</me>?</p>
      </section>

    </chapter>

    <backmatter xml:id="backmatter">
      <title>Backmatter</title>

      <colophon>
        <p> This book was authored in <pretext />. </p>
      </colophon>

    </backmatter>

  </book>

</pretext>
```

Listing 2.4.1 Source of a simple PreTeXt book project.

2.5 Paragraphs, Lists, and Blocks

Within each division (chapter, section, etc., see Section 2.4) of your book, you likely want some content (e.g. what you're reading right now!).

Written content is usually structured as **paragraphs**, `<p>` for short. If you've ever written HTML, this tag may be familiar to you, but be warned: while PreTeXt is XML (Section 2.3), *PreTeXt is not HTML*! There is some overlap: you can *emphasize* words or phrases with `<em>` for instance. However, while HTML uses the full word "code" for its tag, PreTeXt uses the shortened `<c>` tag.

Note that these elements are all **semantic**: they express the *meaning* of content, not its presentation. For example, the word "semantic" was a `<term>`, we just defined, while we merely emphasized "meaning" with `<em>`. The presentation of these concepts may vary by output format, likely using some combination of boldface, italics, or underlining.

Heads up! We'll talk about customizing presentation later, but it's important to remember that the PreTeXt community separates such "publication" decisions away from the work of "authoring" content.

For users coming from LaTeX, rest assured your mathematical formulas work in PreTeXt. Inline mathmode $ax^2+bx+c=0$ is invoked with `<m>ax^2+bx+c=0</m>`, while display mathematics like

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

is available via `<me>x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}</me>`. (Users from LaTeX will also appreciate that quotes are surrounded with `<q>` in PreTeXt to handle the different way quotation marks are handled in LaTeX vs most other markup languages.)

You may also have lists withinn paragraphs, ordered `<ol>` and unordered `<ul>`, nested as needed. Each list item is represented by `<li>`.

- A single item.
- An item with an ordered list.
    1. First item.
    2. Second item.

Of course, often you have important **blocks** of content to include, such as `<definition>`s or `<claim>`s.

Definition 2.5.1 PreTeXt is an uncomplicated XML language for describing scholarly documents. ◊

Claim 2.5.2 *PreTeXt is the language that will replace LaTeX for authors.*

Proof. Left to the reader. ■

Such content is automatically numbered appropriately. Each of the blocks above is structured with a `<statement>`, and Claim 2.5.2 additionally features a `<proof>`.

Content is often "knowled". A **knowl** is a piece of context-independent information that is useful to transclude elsewhere in the HTML build of your document. For example, in the HTML build for this document, the above proof is knowled by default, and clicking the referenced "Claim" in the previous paragraph expands its knowl to reveal the claim for the reader.

Note 2.5.3 Because this document was edited directly on GitHub using Codespaces, and served with GitHub pages, finding its source is simple: head to its repository[1] and find the corresponding source file[2]. Check the link out to see exactly how each claim, list, etc. in this chapter was marked up!

2.6 Figures and Diagrams

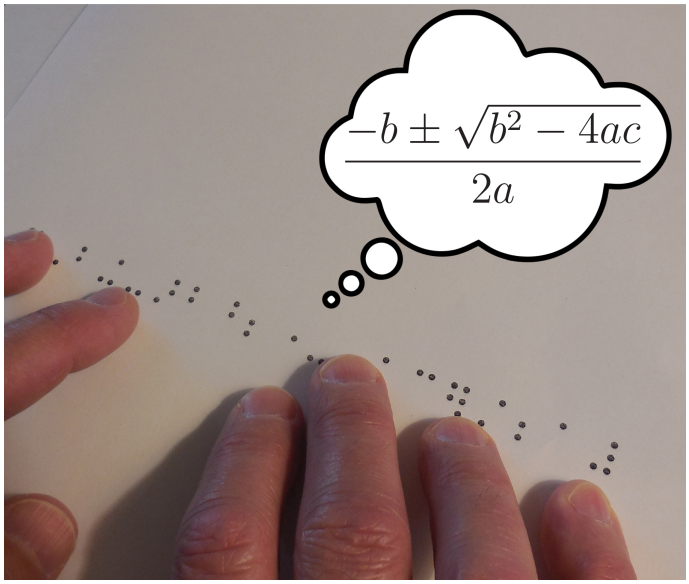


Figure 2.6.1 Photograph taken from AIM Press Release on braille, provided as a JPEG in the project `assets` directory.

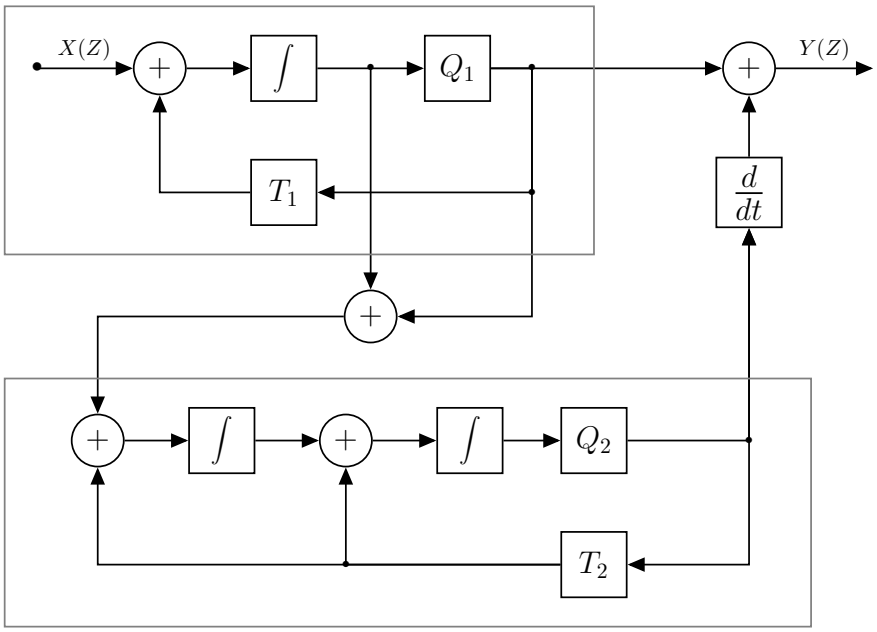


Figure 2.6.2 Electronics Diagram generated with Tikz code

[1] `github.com/StevenClontz/pretext-getting-started-2023/`
[2] `github.com/StevenClontz/pretext-getting-started-2023/blob/main/source/ch-intro.ptx`

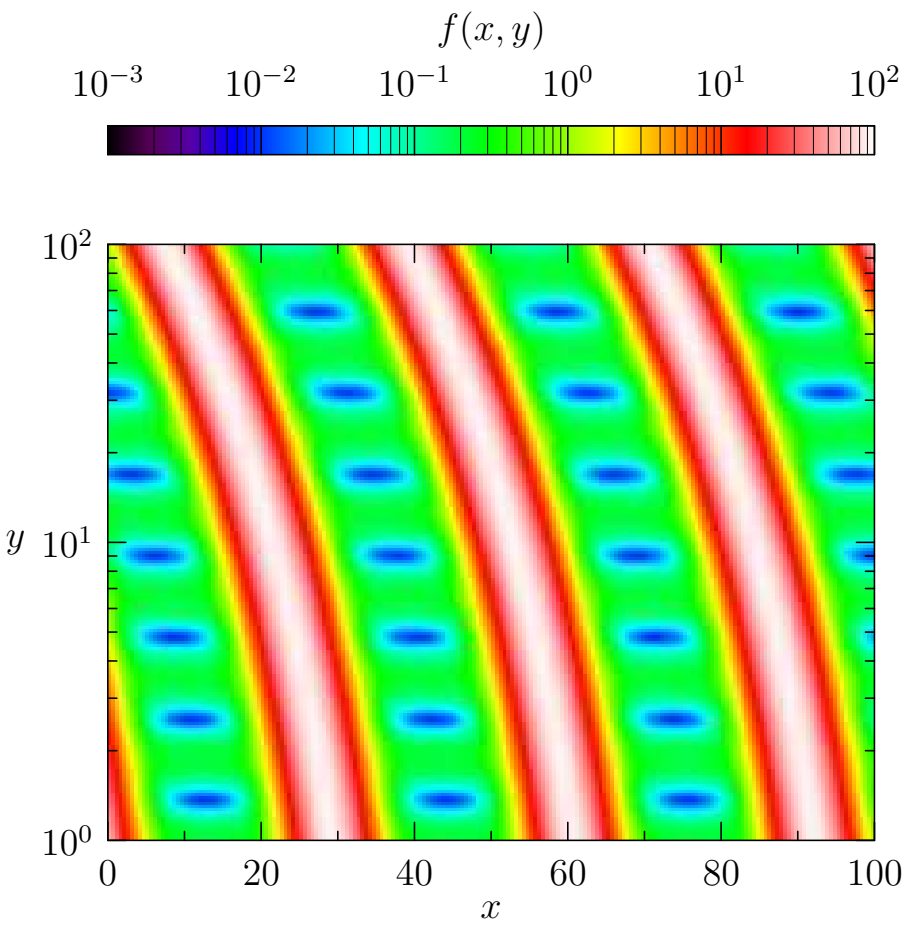


Figure 2.6.3 Contour Plot generated from Asymptote code

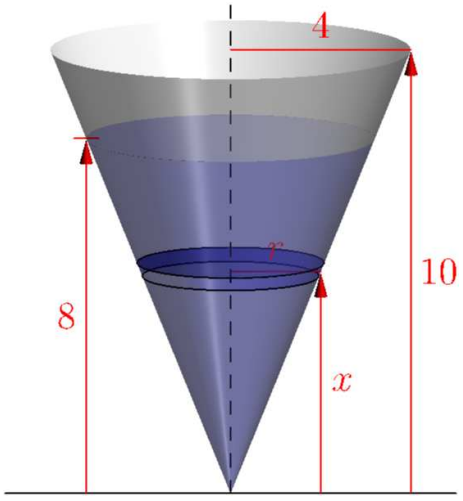


Figure 2.6.4 Work Cone generated from Asymptote code

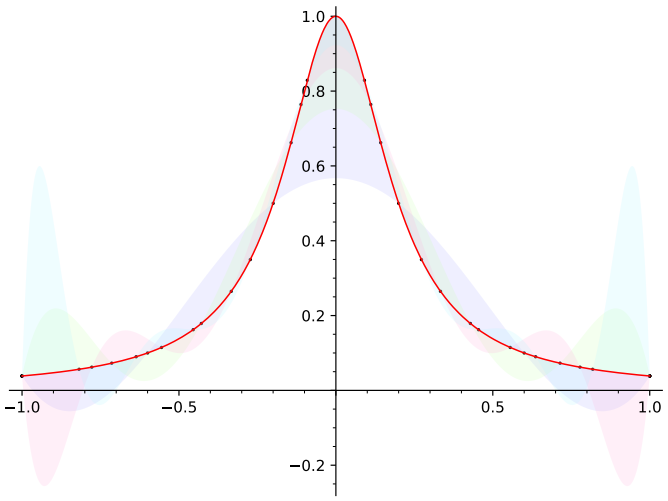


Figure 2.6.5 Polynomial approximations of $f(x) = 1/(1 + 25x^2)$ generated from SageMath code

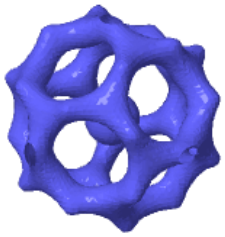
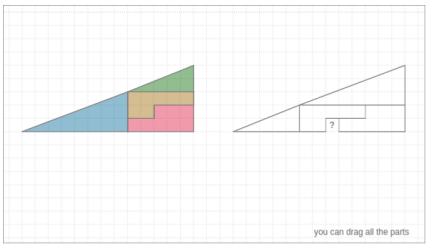


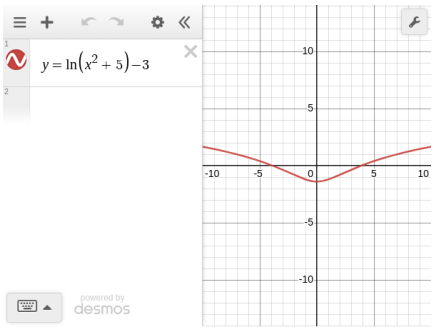
Figure 2.6.6 An implicitly defined 3D surface generated with SageMath code

2.7 Interactives



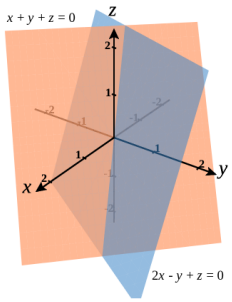
Interactive[1]

Figure 2.7.1 Right Triangle Paradox powered by Geogebra



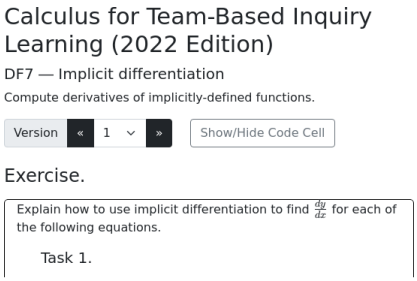
Interactive[2]

Figure 2.7.2 Graph of $ln(x^2 + 5) - 3$ powered by Desmos



Interactive[3]

Figure 2.7.3 Intersection of two planes powered by CalcPlot3D



Interactive[4]

Figure 2.7.4 Implicit Differentiation exercises powered by CheckIt

[1] `geogebra-right-triangle.html`
[2] `desmos-natural-log.html`
[3] `calcplot3d-intersection-planes.html`
[4] `interactive-checkit.html`

2.8 Authoring an Exercise

Checkpoint 2.8.1 Manually-authored exercise. What is $2 + 2$?

Hint. We're being a bit tricky here...

Answer. 22

Solution. 22, where + is the concatenation operator.

Checkpoint 2.8.2 Using WebWork. Compute $1 + 7$.
The sum is __.

Answer. 8

Solution. $1 + 7 = 8$.

Checkpoint 2.8.3 was taken from Linear Algebra for Team-Based Inquiry Learning[1].

Checkpoint 2.8.3 Generated by CheckIt. Consider the following system of linear equations.

$$\begin{array}{rcrcrcrcr} 4\,x_1 & + & 3\,x_2 & + & 18\,x_3 & - & 11\,x_4 & = & -14 \\ -3\,x_1 & + & x_2 & - & 7\,x_3 & + & 5\,x_4 & = & 4 \\ 3\,x_1 & + & 3\,x_2 & + & 15\,x_3 & - & 9\,x_4 & = & -12 \end{array}$$

(a) Explain how to find a simpler system or vector equation that has the same solution set.

Answer.

$$\text{RREF}\left[\begin{array}{cccc|c} 4 & 3 & 18 & -11 & -14 \\ -3 & 1 & -7 & 5 & 4 \\ 3 & 3 & 15 & -9 & -12 \end{array}\right] = \left[\begin{array}{cccc|c} 1 & 0 & 3 & -2 & -2 \\ 0 & 1 & 2 & -1 & -2 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

(b) Explain how to describe this solution set using set notation.

Answer. The solution set is $\left\{ \left[\begin{array}{c} -3\,a + 2\,b - 2 \\ -2\,a + b - 2 \\ a \\ b \end{array}\right] \middle| \, a, b \in \mathbb{R} \right\}$.

Checkpoint 2.8.4 Runestone-powered True/False Question. This statement is true or false.
True or False?

Answer. True.

Solution. True.
Feedback goes here.

Checkpoint 2.8.5 Runestone-powered Matching Problem. A multiple choice question

A. right answer 1

B. right answer 1

C. wrong answer 1

D. wrong answer 2

[1] teambasedinquirylearning.github.io/linear-algebra/

Answer. A, B.

Solution.

A. *Correct.*

answer specific feedback

B. *Correct.*

answer specific feedback

C. *Incorrect.*

answer specific feedback

D. *Incorrect.*

answer specific feedback

Checkpoint 2.8.6 Runestone-powered Parsons Problem. Rearrange the blocks in alphebetical order, ignoring blocks beginning with a number.

- C
- B
- 103
- D
- A

Solution.

- A
- B
- C
- D

Checkpoint 2.8.7 Runestone-powered Matching Problem. Match the letters with their order in the alphabet

A
B
C

Solution.

A	1
B	2
C	3

2.9 Not Just HTML, Not Just PDF

This document is available in HTML format at https://stevenclontz.github.io/pretext-getting-started-2023/, and the same document is available in a PDF format at https://stevenclontz.github.io/pretext-getting-started-2023/print/main.pdf.

Obtaining these formats are as easy as running `pretext build web` and `pretext build print` respectively. PreTeXt supports other types of output as

well, including Jupyter notebooks, ePub, and tacticle braille code.

You're encouraged to view authoring in PreTeXt as an *investment*: you may not need the braille output today, but the little extra thought and care required to author in PreTeXt will allow you to provide this version of your document to a blind student tomorrow.

2.10 Wrapping Up

Go check back on the Codespace you created in Section 2.1. If it's up and running, you're ready to move on to the next section!

Chapter 3

Codespace Workflow

Once you create content in PreTeXt, it is time to **build** it, perhaps **generate assets** (if your project has such assets), **view** the result to make sure it looks good, and when you are ready, **deploy** your web output to a live webpage for others to marvel at.

In this chapter we will walk through the steps to do this inside a codespace environment.

3.1 Build and Generate

Inside the VS Code window you opened through codespaces, have a `.ptx` file open. You can build your entire project in a few different ways.

- Click the "PreTeXt" button in the VS Code status bar (just left of center on the bottom of the window), or use the keyboard shortcut `CTRL+ALT+p`, to bring up the PreTeXt Commands menu. Select "Build" from the menu. This will bring up a list of build targets. Select "web" from the list. This will build your entire project, and put the output in the `build/web` folder.
- You can type `pretext build web` from the terminal (this uses the CLI directly). If you don't see a terminal window at the bottom of the screen, you can open it with `CTRL+\``.
- You can use the keyboard shortcut `CTRL+ALT+b`. This will build the project using the last build target you used. If you haven't built yet, it will use the default build target, which is "web".

Note 3.1.1 All shortcuts listed in this guide assume you are using Windows or Linux. If you are using a Mac, you will need to use `CMD` instead of `CTRL`.

If your documentment contains some more complicated elements, you might need to **generate** them for them to show up. The elements that require this are (depending on what your build target is):

- `<latex-image>`
- `<sagemath>`
- `<asymptote>`
- `<youtube>` (for thumbnail previews)
- `<webwork>`

- `<codelense>`

You can generate assets in much the same way you run a build. The PreTeXt command menu has a `Generate` choice that you select, and then select your target (different target formats have different requirements for what is generated). You can also type `pretext generate` in the terminal, or use the keyboard shortcut `CTRL+ALT+g`.

Note 3.1.2 Note that generating assets requires additional software, like LaTeX or Sage. Depending on your codespace setup, you can likely fix erorrs by entering the following command in the terminal:
`sudo bash ./.devcontainer/postCreateCommand.sh`

Once you install this software once, you should be good to go as long as your codespace exists.

3.2 Previewing

You can check the output of what you built using the **View** command. Again, you can access this in multiple ways. If you use the PreTeXt Commands menu ans select "view", you will get a choice for your viewer. We currently suggest using "Live Preview", although on codespace this requires a few extra steps:

1. When the side preview opens, it will ask you if you want to open in an external browser. Click "Open in browser". If you don't see the pop-up, you can clicke on "Port: 3000" in the bottom status bar of VS Code and select "Open in browser" from the pop-up menu.
2. Now close the new tab that opens, and also close the side panel in VS Code that opened.
3. Finally select "View" again, and it should just work. You can drag the VS Code tab to un-split the window to make it easier to view.

You can also experiment with the CLI view command, by selecting that from the pop-up menu, or by typing `pretext view web` in the terminal. This should pop up a new browser tab with the preview. The only reason we caution against this currently is that the local server that gets started to preview the files will keep running as long as your codespace is active, and we don't understand how this affects resource use.

When you make edits to your source files, you will need to build again, and then refresh the preview window to see the changes.

3.3 Saving your work

Using codespaces will keep all your files "in the cloud" on GitHub's server. As long as you don't delete your codespace, your files will be saved there. However, you will want to **push** these files to your **git repository** on GitHub to make this save perminent. This has the benefit of allowing collaborators to access your files as well (your codespace is unique to your account).

There is a *lot* to learn about git, but luckily using VS Code lets you do everything you need using menus (you don't need to use the command line, unless you want to). Everything can be controlled using the **Source Control** view: it should be third from the top on the very left of the window, an icon with splitting paths, and likely a badge showing how many files you have changed.

Here are the basic concepts you need to understand.

- The program **git** keeps track of all the changes you make to files inside of your **repository** (in this case, the folder containing your project).
- Once you have edited your files and are happy with all of them, you tell git to track the set of changes as a **commit**. This creates a handy *breakpoint* you could return to if you want to go back to an earlier version.

  There are two steps to creating a commit (which you can often do all at once in practice):

  1. You **stage** the files you want to update in the commit.
     You **commit** the stage files including a **commit message**.

  Doing this in two steps can be helpful if you want to commit only some of the files that have changed.
- Once you have one or more commits, you need to sync these changes with GitHub. To "upload" your changes, you **push** the repository. To download changes that you are someone else made, you **pull** the repository.

Now, how do we do these things in VS Code? Start by looking at the Source Control view. You will notice a list of files that were changed. You can click on any of these to see what the changes are (you will see a side-by-side view of the original and updated version).

If you are comfortable staging and committing in one step, you can simply write yourself a short message in the textbox above the big green "Commit" button, and click the button. If you want to stage first, click the + next to each file under "changes" to stage them.

The green button should now turn into a "Sync" button. When you click that, it will do a quick pull and then a push, to sync changes with GitHub.

The only small point about using git is that no all files will be tracked. This is on purpose, since temporary files really should not be "remembered" using this version control setting. Which files or types of files are ignored by git is controled by the ".gitignore" file in your repository.

In particular, we do not track the output of builds. Git is used to track progress on your source, which you build into output at any time. If you want others to be able to see the output of your work without building it themselves, you need to deploy your work.

3.4 Deploy

So you have worked tirelessly to prepare course notes or a book, built and previewed, synced changes using git, and now you are ready to share the results of your efforts with the world. It's time to **deploy** your project.

With our codespace setup this is simple. From the "PreTeXt Commands" menu, click on "Deploy". This will automatically take the most recent build of your web target and host it through GitHub Pages[1]. Watch the output pane for a link to your published site. (It can take a few minutes for the site to get set up or updated; there should be another link to view the progress of the GitHub "action" that reports the progress.)

[1] pages.github.com/

Chapter 4

Practice Activities

Let's get our hands dirty and try writing some PreTeXt. If you have some ideas of what you want to try, go for it. Otherwise, try to implement as many of the following as you can.

4.1 Basic PreTeXt

Practice 4.1.1 Paragraph contents.

(a) Write a paragraph of text that includes an *emphasized* word, a "quoted" word, and inline math using a variable like x or y.

(b) Write the definition of a term inside a paragraph (not as a numbered definition). The defined word should be clear when you ready it. For example: a **slark** is a hybrid of a sloth and a shark.

(c) Write a paragraph that includes a math equation on a displayed line. Like this:

$$f(x) = \begin{cases} 1 & x = 1 \\ 0 & x \neq 1 \end{cases}.$$

Practice 4.1.2 Lists and Paragraphs. Write a numbered list of your favorite three animals and an unnumbered list of your three favorite numbers (in no particular order). Put text introducing the lists between them.

Practice 4.1.3

(a) Create the following table using the pretext `tabular` environment.

1	2	3
4	5	6

(b) Create the same table using a LaTeX array environment.

1	2	3
4	5	6

4.2 Blocks

Practice 4.2.1

(a) Write a numbered definition. Don't forget to highlight the word that is being defined.

(b) Oh no! You just realized that your definition is actually more of an axiom. What are you going to do!?!?

Practice 4.2.2

(a) Write a lemma. Since it is probably obvious, you can leave off the proof.

(b) Write a theorem, and include a proof. The proof should use (and reference) the lemma (which means you need to go back and identify the lemma some how).

(c) What other types of mathematical statements might you want to include now? Maybe you put a remark or a note or a corollary? What would you want to do? What can you do?

Practice 4.2.3

(a) Write a numbered example.

(b) Sometimes an example is of a problem you would ask students to solve. You might include a hint or solution for such examples. Give an example of such a thing, with a hint and solution.

Practice 4.2.4

(a) Write an **activity** that you would want your students to try. Give your activity a hint.

(b) Sometimes an activity has multiple parts. You could give each part as an item on a numbered list, but that would not allow parts to have their own hint/answer/solution.

Instead, you can make each part its own `<task>`. Try writing a multi-part activity, some parts having hints, others having a solution.

Exercises can appear at the end of a section, in an `<exercises>` division, or they can be "checkpoint" exercises that show up in the middle of a section.

Practice 4.2.5 Write a checkpoint exercise. Up to you whether it gets parts (i.e. tasks).

What we learned.

You can do lots of things in PreTeXt.

Practice 4.2.6 Write a summary box like the one above. You might say that such a box "assembles" the concepts you have discussed...

4.3 Challenges

Want to push yourself a bit? Try some of these bonus activities

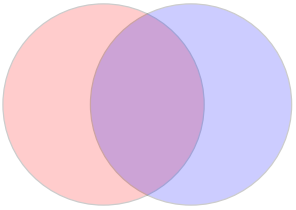
Challenge 4.3.1 Replicated the following:

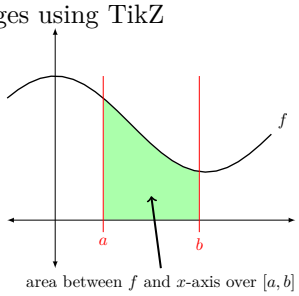
$f(x) = x^2$

x	$f(x)$
1	1
2	4
3	9

Table 4.3.1 Table of Values

Challenge 4.3.2 Create the folowing images using TikZ





4.4 Adapting existing content

Activity 4.4.1 Nest the following PreTeXt elements in order to produce the "Graphing a Derivative" subsection of OpenStax Calculus Volume 1 Section 3.2[1].

```
<caption>The derivative <m>f'(x)</m> is positive everywhere
    because the function <m>f(x)</m> is
    increasing.</caption>
<caption>The derivative <m>f'(x)\lt 0</m> where the
    function <m>f(x)</m> is decreasing and <m>f'(x)\gt
    0</m> where <m>f(x)</m> is increasing...</caption>
<description>The function f(x) = the square root of x is
    graphed as is its derivative f'(x) = 1/(2 times the
    square root of x).</description>
<description>Two functions are graphed here: f(x) and
    f'(x). The function f(x) is the same as the above
    graph, that is, roughly sinusoidal, starting at (-4,
    3), decreasing to a local minimum at (-2, 2), then
    increasing to a local maximum at (3, 6), and getting
    cut off at (7, 2). The function f'(x) is an
    downward-facing parabola with vertex near (0.5, 1.75),
    y-intercept (0, 1.5), and x-intercepts (-1.9, 0) and
    (3, 0).</description>
<description>The function f(x) = x squared - 2x is graphed
    as is its derivative f'(x) = 2x - 2.</description>
<description>The function f(x) is roughly sinusoidal,
    starting at (-4, 3), decreasing to a local minimum at
    (-2, 2), then increasing to a local maximum at (3, 6),
    and getting cut off at (7, 2).</description>
<example></example>
<exercise></exercise>
<figure></figure>
<figure></figure>
<image source="solution-image.jpg"></image>
<image source="fx-sqrt-x.jpg"></image>
<image source="exercise-image.jpg"></image>
<image source="fx-x2-2x.jpg"></image>
<p>We have already discussed how to graph a function...</p>
<p>In Example 3.12 we found that for
    <m>f(x)=x^2-2x,f'(x)=2x-2</m>...</p>
<p>Sketch the graph of <m>f(x)=x^2-4</m>...</p>
<p>In Example 3.11 we found that for
    <m>f(x)=\sqrt{x},f'(x)=1/2\sqrt{x}</m>...</p>
<p>The solution is shown in the following graph. Observe
    that...</p>
<p>Use the following graph of <m>f(x)</m> to sketch a graph
    of <m>f'(x)</m>.</p>
<solution></solution>
<statement></statement>
<statement></statement>
<subsection></subsection>
<title>Graphing a Derivative</title>
<title>Sketching a Derivative Using a Function</title>
```

Activity 4.4.2 Consider the following concerning the first two pages (leading up until the "Units" header) of Elements of Abstract and Linear Algebra[2].

(a) Consider how this content might be scaffolded in terms of paragraphs, theorems, statements, proofs, and so on (not necessarily in PreTeXt terms).

(b) Nest the following PreTeXt elements in order to reproduce this portion of .

[1] openstax.org/books/calculus-volume-1/pages/3-2-the-derivative-as-a-function#fs-id1169737770972

```
<definition></definition>
<example></example>
<introduction></introduction>
<li><p>If <m>a,b,c\in R</m>... (associative)</p></li>
<li><p><m>(-a)\cdot b=a\cdot(-b)=-(a\cdot
    b)</m>.</p></li>
<li><p>If <m>a,b,c\in R</m>... (distribuative)</p></li>
<li><p><m>(na)\cdot(mb)=(nm)(a\cdot b)</m>...</p></li>
<li><p><m>a\cdot
    \underline{0}=\underline{0}\cdot=\underline{0}</m>...</p></li>
<li><p><m>R</m> has a multiplicative
    identity...</p></li>
<li><p>Let <m>\underline{n}=n\underline{1}</m>. For
    example...</p></li>
<li><p>If <m>a,b\in R</m>... (commutative)</p></li>
<p>Suppose <m>R</m> is an additive abelian group...</p>
<p>The next two theorems show that ring
    mulitplication...</p>
<p>Suppose <m>R</m> is a ring and <m>a,b\in R</m>.</p>
<p>Suppose <m>a,b\in R</m> and <m>n,m\in \mathbf
    Z</m>.</p>
<p>Rings are additive abelian groups with a second
    operation...</p>
<p>The basic commutative rings in mathematics are the
    integers <m>\mathbf Z</m>...</p>
<p>Recall that, since <m>R</m> is an additive abelien
    group...</p>
<p>If 1,2,3 are satisfied...</p>
<p><ol></ol></p>
<p><ol></ol></p>
<p><ol></ol></p>
<section></section>
<statement></statement>
<statement></statement>
<statement></statement>
<theorem></theorem>
<title>Rings</title>
<theorem></theorem>
<statement></statement>
```

[2] www.math.miami.edu/~ec/book/ch03.pdf

Chapter 5

Publishing

By **publishing**, we mean here the process of finalizing the presentation and look of the output you get from PreTeXt. We stress again the philosophical distinction between **author** and **publisher** in PreTeXt, since there is great value in focusing on just one of these two types of tasks at a time.

Another aspect of the publishing process is that of creating the physical book and making it available in bookstores, or making the the web version available on your own website. The latter can be done automatically through GitHub pages using the `pretext deploy` function, but perhaps you also want to have a separate landing page that links to a version of the book you are celling on Amazon.com. However, these steps are beyond the scope of this guide (as for suggestions on the PreTeXt-support group if you would like some assistance getting started with these steps).

For this chapter we will assume you are trying to produce a single **version** of your book and want to control the presentation and look of that version. You can still produce this version in multiple **formats** (web, pdf, epub), and which you want to produce might change what you do slightly.

5.1 The publication file

PreTeXt controls publisher options in a **publication file**. The default location of this file is in the folder `publication` and is itself called `publication.ptx`. Open this file and take a look.

What you will see is more XML, but a different set of tags than what you would write in your PreTeXt source. The version of the publication file that came with your codespace was (at the one time) the almost complete list of options with default values provided. This should make it easy to modify values and experiment with different settings.

As new settings are introduced, you can find documentation ins the Publication File Reference[1] part of the guide.

Remark 5.1.1 Eventually you may want to have different versions of your document, and you can do this with different publication files. In Chapter 6 we will consider how to manage multiple publication files easily.

Next we will dive deeper into the different sorts of things that can be configured with this file.

[1] `pretextbook.org/doc/guide/html/publication-file-reference.html`

5.2 Showing and Hiding things

We have seen that some **blocks**, like proofs, solutions, examples, are sometimes *hidden* behind a **knowl**: when you click on the title, the content expands to reveal itself. Most of the blocks can be "knowled" or not.

To edit these settings, look for the element `<html>`, and inside that `<knowl/>`. This element has a number of **attributes** (the PreTeXt guide marks attributes with the `@` symbol, by the way). You can specify whether elements should be knowled or not by changing the value of the attributes between yes and no.

Whether exercises are knowled depends on the type of exercises. PreTeXt distinguishes five types of exercise-like elements, since they have hints/answers/solutions that you might want to hide:

- inline, which show up as "checkpoints"; these are exercises mixed with other content in a divisions.
- divisional; these are exercises that belong to an `<exercises>` division.
- project; these are the entire project-like element like `<activity>`, `<investigation>` etc.
- worksheet; essentially a project but its a division itself, and can contain spacing for students to work in when printed.
- readingquestions; these are questions in a `<reading-questions>` division. These were exercises designed to get a text box that students can type short answers to questions at the end of a section.

You can control whether the entire exercise-like element is knowled, but their hints/answers/solutions are always knowled (no spoilers!). The only solution you can can choose to not knowl is that of an example, and that is controlled by these knowl switches.

You can, however, control which parts of exercises show up at all. This is set in a different section of the publication file: `publication/common/exercise-inline` (that is, in the `<common>` block in the `<exercise-inline>` element). You can replace `exercise-inline` by any of the five exercise types. Then as attributes, specify yes or no for the `@statement`, `@hint`, `@answer`, `@solution`.

This controls what shows up in the output, in all formats, at the position that the exercise was authored. There are also ways to redisplay any parts of these exercises later, to give a list of all solutions in the back of the book, say. See Exercises and Solutions[1] from the guide.

5.3 Numbering

There are a few settings that control how things are numbered. Most of these live inside the `<numbering>` element of the publication file.

- The `<division/>` element and its attributes says how "deep" the numbering goes (i.e., do you put numbers on subsubsection?). This is set as a natural number as the value of the `@level` attribute.
- Also in this element you can specify whether parts are **structural** or **decorative**. This essentially says whether chapter number should restart in each part.

[1] `pretextbook.org/doc/guide/html/topic-exercises-solutions.html#topic-solutions-division`

- You can also set the number of the first chapter here using the `@chapter-start` attribute.
- There are four different types of elements that can be numbered independently: **blocks**, **projects**, **equations**, and **footnotes**. You can control how specific the numbering are for each of these by giving the corresponding element a `@level` attribute.

5.4 Formatting

While there are some ways to control the look of the pdf, that is beyond the scope of this guide. Web output can be controlled more easily, selecting from a small number of predefined styles.

This is a feature that is currently under active development, so we will not say much here, but instead refer you again to the Publication File Reference[1] part of the guide.

[1] `pretextbook.org/doc/guide/html/publication-file-reference.html`

Chapter 6

Managing Your Project

6.1 File and directory structure

Here we describe the file and directory structure of a project. PreTeXt allows a fair amount of flexibility in how you structure the project, but we believe the following are best practices.

`source` folder	Contains all your `.ptx` files that hold the content of your document. A new book starts with just a single file, but later we will see how to modularize the source to make organizing it easier.
`publication` folder	Contains your `publication.ptx` file or files, which define the publication-specific information about your document, as we saw in Chapter 5
`assets` folder	Put any images or other static files that you will include in your document here. This does not include images that you describe inside your source (like `<latex-image>`). You can have subfolders as you like, and if you refer to these files in your source, you do ***not*** use "assets" as part of the file name (PreTeXt knows where to look, since this is specified in the `publication.ptx` file.)
`generated-assets` folder	This is a folder that PreTeXt automatically creates and fills with assets that it generates from your source. You shouldn't edit anything in this folder. It is not tracked by git by default.
`output` folder	Another folder created by PreTeXt. It will contain the results of `pretext build`. In general, you should not touch anything in this folder. Not tracked by git by default.
`project.ptx` file	This is the **project manifest** file, which you use to manage the different builds of your project. We will describe how to use this in more detail below.

There are a few other files that you might see in a project, such as `requirements.txt` and `.gitignore`. Don't worry about these for now.

When your project grows, you will likely want to separate your `main.ptx` source file (inside the `source` folder) into multiple `.ptx` files. For example, you might want a single file per chapter, and even a separate file for each section. You can do this by using the `<xi:include>` tag. For example, if you have a

file source/chapter1.ptx that contains the first chapter of your book, you can include it in your main.ptx file like this:

```
<xi:include href="./chapter1.ptx" />
```

In the top level tag of the file in which you use <xi:include> (in this case, main.ptx), you must add the following attribute:

```
xmlns:xi="http://www.w3.org/2001/XInclude"
```

Then in the chapter1.ptx file, you would start with the standard <?xml version="1.0" encoding="UTF-8"?> followed by the top-level tag <chapter>. There can only be one top level tag in this file. A second chapter would need to be its own file and <xi:include> it separately.

More information can be found in the Modular Source Files[1] section of the guide.

6.2 Project.ptx

You will likely want to build your source file into multiple output formats. How do you tell PreTeXt what outputs you want? You keep track of this, and some other information, in your project.ptx file.

The project.ptx file is a **project manifest** file. It is used to manage the different builds of your project. It is a .ptx file, but it is not part of the content of your document. It is used to tell PreTeXt how to build your document.

For now, take a look at the project.ptx file in this project. You will see that there are multiple **targets** each with a **name**, and a specified <format>, <source>, <publication> (file), and <output-dir> (directory). The <source> and <publication> are probably the same for all targets, but they don't have to be.

For a complete description of this file and its use, see https://pretextbook.org/doc/guide/html/processing-CLI.html#cli-project-manifest.

6.3 Versions

6.4 Building subsets

6.5 Codespaces, GitHub, and GitHub Pages

[1] pretextbook.org/doc/guide/html/processing-modular.html

Chapter 7

Converting To PreTeXt

Suppose you have materials written in another format, such as MS Word or LaTeX, and would like to convert these to PreTeXt. Alas, there is not a perfect way to make this happen, but what follows is some general advice to get you started. We would love to hear what sort of workflow you would like to see so we can improve this experience.

Warning 7.0.1 Your goal in doing a conversion should be to move all your materials into PreTeXt and then always work from the PreTeXt source, converting back to other formats as needed. It is simply not feasible to keep converting to PreTeXt, since this process is not automatic.

7.1 Using Pandoc

If you have a number of individual documents that you would like to convert, from pretty much any format, consider trying Pandoc[1]. This command-line tool can read in lots of different formats and output in lots of different formats. Unfortunately, PreTeXt is not a default output format, so you need to use a custom **writer**.

The pretext-tools VS Code extension has a shortcut for using the custom writer and pandoc, assuming you have pandoc installed (which you will if you use a codespace). You will need to upload the file you want to convert, either by dragging it to the Explorer menu in VS Code. Then open the Command Pallet with `CTRL+SHIFT+P` and start typing "pretext: convert file to pretext" and select the item to shows up. You will be prompted for the name of the file you want to convert. The file will be converted and opened in a new tab. You can then copy/paste the contents into your main documents.

You can also use Pandoc outside of VS Code. Install pandoc from their website. Then download the this writer[2] and put it somewhere you will remember. Documentation for how to use this tool is available on its GitHub repository[3].

7.2 Using the community

If you have a mostly complete project, written entirely in not-too-customized LaTeX, David Farmer has volunteered to help you convert the entire project to

[1] `pandoc.org/`
[2] `raw.githubusercontent.com/oscarlevin/pandoc-pretext/master/pretext.lua`
[3] `github.com/oscarlevin/pandoc-pretext`

likely 90% correct working PreTeXt. See https://pretextbook.org/conversions.html for details.

7.3 By hand

If you have some familiarity with Regex, you can use VS Code's find/replace tools to clean up text that you copied/pasted into your document. For example, if you entered `\$(.*?)\$` in the search bar (with the ".*" setting selected), and put `<m>$1</m>` in the replace box, you can easily replace all math in your document with the correct syntax.

Appendix A

Getting Help

Here we collect a number of useful resources to help you when you are stuck. The official PreTeXt site[1] has lots of resources, but we understand it can be overwhelming.

Official documentation. Note that the official PreTeXt Guide[2] can be hard to use because there is so much stuff in it. Additionally, some of the documentation is out of date. Still, if you know where to look, it is a great resource.

Here are some sections that we find especially helpful:

- Basics Reference[3]: A listing of the main elements of PreTeXt including snippets of the code that create them. This is one of the few places in the guide that has examples of the markup.
- Publication File Reference[4]: When you are ready to start changing how your output looks, you can use the **publication file**, which is described in this part of the guide.
- PreTeXt Schema[5]: The official list of elements and where they can go is given in the PreTeXt Schema, which is described here. Also you can check out the schema browser[6] to actually view the schema.
- Getting PreTeXt[7]: If you want to install PreTeXt on your own computer, this early part of the guide gives you directions. It should be updated with information on CodeSpaces soon as well, if you need a refresher.

Finally, note that the search in PreTeXt now works really well, and searching for a feature will usually get you pointed in the right spot.

Examples. The Examples[8] page on the PreTeXt site contains a number of useful live examples. Links are provided to web, pdf, and source (on GitHub). For some of the examples, there is also an *annotated* version available. We find these especially helpful since you can "view source" to see exactly how each bit of the example was marked up in code.

Here are some of the most useful such examples:

[1] pretextbook.org/
[2] pretextbook.org/doc/guide/html/guide-toc.html
[3] pretextbook.org/doc/guide/html/part-basics.html
[4] pretextbook.org/doc/guide/html/publication-file-reference.html
[5] pretextbook.org/doc/guide/html/schema.html
[6] pretextbook.org/doc/schema/
[7] pretextbook.org/doc/guide/html/quickstart-getting-pretext.html
[8] pretextbook.org/examples.html

- Sample Book[9]: This annotated sample book contains a section on interactive exercises[10]. The PreTeXt developers use this book for testing, so you can see the latest (sometimes experimental) features available.
- Sample Article[11]: Not particularly well organized (it is also a proving ground for developers) but this contains almost every variation of every feature of PreTeXt. Using the search and "view source" makes this an invaluable resource.

Community Support. There is a very active google group for support: pretext-support[12]. You should also subscribe to the low-traffic pretext-announce[13] to get updates.

This spring we will host daily virtual "drop-in" sessions to support authoring and devlopement of PreTeXt. Information will be posted to the pretext-announce google group.

[9] pretextbook.org/examples/sample-book/annotated/
[10] pretextbook.org/examples/sample-book/annotated/interactive-exercises.html
[11] pretextbook.org/examples/sample-article/annotated
[12] groups.google.com/g/pretext-support
[13] groups.google.com/g/pretext-announce

Appendix B

Copyright and Licensing

[1] creativecommons.org/licenses/by-sa/4.0

Appendix C

Acknowledgement

We would like to thank the American Institute of Mathematics[1] for sponsoring us to present this Professional Enhancement Program at the 2023 Joint Mathematics Meeting.

[1] aimath.org