

Using PreTeXt with TeXShop

Richard Koch

August 2, 2022

1 PreTeXt and XML

This document was originally written in 2019. It became out of date due to changes in PreTeXt. Both this document and the PreTeXt engines were updated in August, 2022.

Below are two identical documents. Here is the LaTeX version:

```
\section{Introduction}
  Portland has many attractions.

\section{Details}
  The important attractions are the Oyster Bar in the downtown area
  and Reed college where Steve Jobs learned about fonts.

  Reed also teaches Greek, starting with  $\alpha$ ,  $\beta$ ,  $\gamma$ .
```

and here is the PreTeXt version:

```
<section>
  <title>Introduction</title>
  <p> Portland has many attractions.</p>
</section>

<section>
  <title>Details</title>
  <p>The important attractions are the Oyster Bar in the downtown area
    and Reed college where Steve Jobs learned about fonts. </p>

  <p>Reed also teaches Greek, starting with  $\alpha$ ,  $\beta$ ,  $\gamma$ .</p>
</section>
```

The obvious difference is that TeX has commands like “\section” while PreTeXt has tags like “<section>”. The more significant difference is that PreTeXt and XML rigidly insist that each opening tag have an associated closing tag. This makes it easy to parse XML files, a very significant advantage.

The acronym XML means “extensible markup language”. It codifies a general tag-based schema for encoding information. Many document formats in use today are extensions of XML, and PreTeXt is one of them, defining an extensive set of tag pairs to structure documents.

The most familiar tag-based data structure is HTML for web documents, but it was defined before XML existed and follows the XML rules “rather loosely”. In the HTML world, browsers exist to convert the data into display form for humans, and CSS style files are often used to describe how to display various HTML elements. The notion of a style file was extended to the XML world, where XSL style sheets describe how to display various XML elements, or more generally, how to convert these elements into other formats. PreTeXt contains many such XSL files, including one to convert PreTeXt documents to html and one to convert PreTeXt documents to LaTeX.

There is an open source program called xsltproc which accepts an XML file and an XSL style sheet and uses this information to output the data in converted form. This program is already in Mac OS X.

Therefore, TeXShop can “typeset” PreTeXt documents using xsltproc as soon as it has access to the PreTeXt XSL files. This is easy because the PreTeXt folks package these files, together with examples, documentation, and everything else needed to use PreTeXt, in a single folder named “mathbook”. So the first step in using PreTeXt on Mac OS is to retrieve this folder.

2 Getting mathbook into ~/Documents

Inside the folder ~/Library/TeXShop/Engines/Inactive where you found this document, there is a file named “updatemathbook.sh.” Drag a copy of this file into ~/Documents, that is, the Documents folder in your home directory.

This shell script calls the command “git clone https://github.com/rbeezer/mathbook” which tells your Macintosh to download a copy of mathbook from Robert Beezer’s server. We’ll first explain what to do if you are familiar with Apple’s Terminal program; otherwise read to the end of this section. To get the mathbook folder, open Terminal and change directory to ~/Documents. Execute the shellscript command below and you are done:

```
sh updatemathbook.sh
```

It is useful to remember this command, because PreTeXt is a moving target at the moment and users are currently urged to update the folder once a week or so. Eventually as the project matures this updating can happen less frequently. To update, use the previous command again. The command will refuse to overwrite an existing folder, so rename “mathbook” to “mathbook-old” before issuing the command and keep this backup copy until you are certain that the new version works.

It is possible that the Macintosh will not run the command and instead print a dialog claiming that your Macintosh does not have the command line tools. The dialog will offer to download and install them. Accept the offer. After a couple of minutes, repeat the command.

If this explanation was too brief, we’ll give details. Find Terminal in /Applications/Utilities. Drag a copy of this program to your dock. Double click the Terminal icon to run Terminal. In the Terminal window, type “cd ~/Documents” without the quotation marks and push RETURN. This causes Terminal to change to the directory ~/Documents.

In Terminal, type “sh updatemathbook.sh” without the quotation marks and push RETURN. Terminal will print some information as the folder downloads. Done.

3 Activating the PreTeXt Engines

Return to the folder ~/Library/TeXShop/Engines/Inactive/PreTeXt where the document you are reading resides. Inside this folder you will find five engine files named PreTeXt-LaTeX.engine, PreTeXt-HTML.engine, PreTeXt-Both.engine, PreTeX-LaTeX-Includes.engine, and PreTeXt-Validate.engine. Drag these files or copies of the files two levels up to the folder ~/Library/TeXShop/Engines. The engines are now active. The first will convert a PreTeXt source file to a pdf file and display the result in the Preview window. The second will convert a PreTeXt source file to an html file and display the result in the HTML window. The third will perform both conversions and show the results in both the Preview and HTML windows. The fourth will be explained later. The fifth will check your XML source for tag placement errors, but requires further setup before use. See section 19.

4 Updating /Library/TeXShop

This step is not essential, and if you find it confusing you can skip the entire section.

When TeXShop first runs, it creates a folder in the user’s Library folder in their home directory, ~/Library/TeXShop. Inside this “TeXShop” folder, TeXShop writes various subfolders holding individualized Templates, Stationery, Macros, CommandCompletion files, Engine files, and Themes, i.e., color choices for the editor. TeXShop has a menu command “Open ~/Library/TeXShop” taking users to this folder; the menu is important

because Apple generally hides the `~/Library` folder. Users can modify the items in the various folders. For instance, they may edit or replace templates and stationery; they extend and edit the command completion file, and so forth.

Since these folders contain user information, TeXShop cannot update them. But sometimes TeXShop provides new default information for these folders and users may want to get updated files. How can this be done?

If you just installed TeXShop for the first time, you have the latest files. Stop reading and go to the next section.

If you previously installed TeXShop, but seldom used it, and in particular if you never modified files in `~/Library/TeXShop`, then it is easy to get the latest updates. Use TeXShop to open `~/Library/TeXShop`. Then quit TeXShop. Throw the “TeXShop” folder in the trash. Restart TeXShop. It will create a brand new TeXShop folder and fill it with up to date subfolders. Stop reading and go to the next section.

If, however, you use TeXShop extensively and have created your own macros, or edited your own templates, then you have to obtain the new material more carefully. Go to `~/Library/TeXShop` and open the “New” folder. It has a subfolder named “Version-4.31” with four subfolders. The “Stationery” folder contains two files; drag copies of these files to `~/Library/TeXShop/Stationery`. The “CommandCompletion” folder contains one file; drag a copy of this file to `~/Library/TeXShop/CommandCompletion`. The “Themes” folder can be ignored.

The “Macros” folder contains two files, but the important file is XML-Macros.plist, which contains two optional new macros. It is tricky to deal with this; feel free to stop here. Otherwise, if you have edited the Macros in TeXShop, you will know how to open this file and add the macros there to the existing LaTeX macros. If you never edited the Macros in TeXShop, it will be easier to quit TeXShop, and move the folder `~/Library/TeXShop/Macros` to your desktop or the trash. The next time TeXShop runs, it will recreate this folder with the latest version of the Macros and you are done.

5 A Very Simple Example

In `~/Documents` there is now a subfolder named “mathbooks”. This is the folder you downloaded from the PreTeXt site. We’ll often refer to this mathbooks folder in the rest of this report. The folder has a subfolder named “examples” and one of these examples is in a subfolder named “hello-world”. Duplicate this folder and drag the copy to your home directory or other location where you store LaTeX projects. Inside the folder is a file titled “hello-world.xml”. This is the source for a very simple PreTeXt document. You’ll soon discover that PreTeXt source files either have extension `.xml` or extension `.ptx`.

TeXShop understands that both extensions are PreTeXt files. We'll call them xml-files in this document. Open hello-world.xml in the new TeXShop.

Select the PreTeXt-Both engine in the pull-down menu next to the Typeset button on the toolbar. Then typeset. This engine generates both a pdf file and an html file, and opens them in two windows. Congratulations. You have typeset your first PreTeXt document.

You may forget to select the PreTeXt engine and get strange errors typesetting with pdflatex. Therefore it is useful to add a magic line to the source telling TeXShop to typeset with the PreTeXt-Both engine. The correct place for this line is in the blank second line of the source, just after an initial xml line and before a comment line with a long row of stars. TeXShop has a macro which makes this easy. Click once in this blank line to tell the Macro where to insert something and then choose the macro titled "PreTeXt Header".

The Macro actually inserts the following:

```
<!--  
% !TEX TS-program = PreTeXt-LaTeX  
-->
```

Note that % is a LaTeX comment symbol, but this does not denote a comment in xml. So PreTeXt would try to interpret the magic line. We avoid this by surrounding the line with xml comment markers.

There is only a macro for the PreTeXt-LaTeX engine, but it is easy enough to reach up and by hand change "LaTeX" to "HTML" or "Both" or "LaTeX-Includes" if you want to use one of the other engines.

6 The Minimal Example

It is time to try a more substantial example. Return to the mathbooks folder and its subfolder of examples. Select "minimal" and duplicate that folder. Drag the duplicate to your home directory or other location where you store LaTeX projects.

The source for the minimal example is in the subfolder named source, which has a single file named "main.ptx". Open that file in TeXShop. Note that this source file has a lot of documentation at the top explaining how to typeset it from the command line. One surprising thing this documentation reveals is that the output files will be named "minimal.tex", "minimal.pdf", and "minimal.html" even though the source file is named "main". This is an unpleasant quirk of PreTeXt, which allows authors to give their output files different names than the source file. Indeed in the xml just below, notice that the article tag assigns this new name. This quirk will also show up in other examples, but without documentation so it is up to you to decipher what is happening.

If you are converting xml to tex, this quirk causes no problems because the command line converter program has a flag allowing us to give the output any name we want. Consequently the engine PreTeXt-LaTeX will just ignore the quirk and work fine. Sadly the command line program which converts xml to html has some sort of bug. It accepts the same flag and outputs an html file with the correct name, but that file is empty. It also outputs a correct html file, but that has the name assigned by PreTeXt.

There may be several ways to get around this problem, but the trick which works for me is just to rename the source file to the name that PreTeXt has chosen for the output. Consequently, rename “main.ptx” to “minimal.ptx”.

Next use the macro “PreTeX Header” to insert a magic line at the top of the source, as in our earlier example.

Finally we are ready to typeset the sample xml file. Typeset first with the PreTeXt-LaTeX engine. We get a two-page document containing mathematics and a table.

Change the magic line and typeset with the PreTeXt-Both engine. We get two pages, and this time we see that the HTML version looks quite a lot different than the pdf version. If you are beginning to understand PreTeXt code, try to add material to the source and typeset again. Both windows will update.

7 Other Examples

It is time to confess that many Sample files in the Examples folder aren’t as easy to process. One sample illustrates font handling in unusual languages. Our PreTeXt-LaTeX engine doesn’t process that example because the tex output file must be typeset with XeLaTeX rather than pdflatex. Of course it is easy enough to modify the engine file and make that happen. But then we run into a second problem. A virtue of XeLaTeX is that it can use system fonts on the host computer and isn’t restricted to fonts in TeX Live. The downside is that such documents are not easily ported to different platforms. And sure enough, the PreTeXt authors seem to have used Linux fonts which XeLaTeX on the Macintosh doesn’t understand.

The PreTeXt project has developed an elaborate build system for documents composed of many different source files, anticipating that authors will typeset from the command line. It isn’t easy to convert these Samples for TeXShop use. But after all, if you decide to use PreTeXt, you get to decide how your source files are handled and you can follow the style of the examples we are discussing here.

8 The Sample-Article

One of the samples is titled Sample-Article. This is a very interesting example because every time a new feature is added to PreTeXt, a sample of that feature is created in Sample-Article. So the sample has almost 214 pages and illustrates a vast number of PreTeXt techniques. Luckily, this sample does typeset in TeXShop. The only problem is that a fair number of illustrations are missing, but pdf_latex runs continuously and handles these errors without intervention.

Copy the entire folder “sample-article” to your home directory. The source file is named “sample-article.xml”. But this is one of those examples where the output file has a different name, “derivatives”. So start by renaming “sample-article.xml” to “derivatives.xml”. Then add the magic line near the top of the file, and by hand change the typesetting engine to PreTeXt-Both.

Typeset. This time typesetting takes a long time, so do not get impatient. Ultimately two windows will appear showing the resulting pdf file and the corresponding html file.

9 The PreTeXt Guide

Go to `~/Documents/mathbook/doc` and find the folder “guide”. Make a copy of this folder and drag it to your home directory. Inside you will find a file `guide.xml`. Open this file in TeXShop. Add the three magic lines to the top as before. Then change the engine to “LaTeX-Includes”. Typeset. There will be a delay with no output in the console, but be patient.

The output is a pdf document, *The PreTeXt Guide*, giving a very readable systematic introduction to PreTeXt, followed by precise details of the language features. This is a good place to learn a little PreTeXt, start using it, and then read more as needed. You need to typeset again to get the table of contents. In 2019, this new engine was not necessary, but the xml source for the Guide has lots of include statements, and an extra flag, `-includes`, is now required to force the conversion engine to accept those includes. That is why there is a different engine.

10 Using TeXShop with PreTeXt

The remaining sections of this document explain features of TeXShop which have been added to improve the experience of creating PreTeXt source files.

One example is syntax-coloring. TeXShop uses different algorithms to syntax-color xml files and to syntax-color other files. How does it know the difference? It uses its old algorithm unless the file extension is xml or ptx or a few others, and in those cases it uses

its new xml syntax-coloring. The same method is used to switch other TeXShop behaviors to be described shortly.

When the user brings a new window to the foreground, TeXShop checks its extension. So a user can edit tex files and xml files at the same time. When a tex file is in front, TeXShop will use its old methods. When an xml file is in front, it will use new methods.

11 Starting a New Document

When TeXShop first starts, it displays a blank window waiting for input. This also happens when the user chooses “New” in the File menu. By default, this window is waiting for TeX or LaTeX input, which it will display using Latex syntax coloring. But the TeXShop “Source” menu has a new command labeled “Convert to XML.” Select this command to switch to PreTeXt syntax coloring in the window.

If you have several windows open, some for editing LaTeX sources and some for editing PreTeXt sources, the “Convert to XML” menu will be checked if the top window is in XML mode, and otherwise will be unchecked.

The first time you typeset a New file, TeXShop will present a Save dialog and ask you to name the file and select its location. Toward the bottom of this dialog, there is a popup menu listing the extension of the file to be created. The default item chosen in this menu will be “tex” for TeX or LaTeX sources, and “xml” for PreTeXt sources, as determined by the “Convert to XML” menu item. When the item is “xml”, it is necessary to “click” this item before saving, since otherwise Apple’s Cocoa frameworks will not recognize the unusual choice and you will end up with a file which will not typeset. Once the file is saved with the “xml” extension, all future save operations with that file will work correctly without further help.

If by mistake you save with the wrong extension, close the source window in TeXShop, switch to the Finder to manually change the file’s extension from “tex” to “xml”, and then switch back to TeXShop and reopen the file.

Another way to create a new PreTeXt document is to use the File menu item “New from Stationery.” This shows a choice of stationery, and one type offered is a generic PreTeXt document. This document will already be set to accept xml source.

After creating a new document, a typical TeXShop user would use the Templates item in the toolbar to fill it with boilerplate starting data. Users can create their own templates, which are just ordinary .tex files, and store them in `~/Library/TeXShop/Templates`. We haven’t created any new templates specifically for PreTeXt users, but this would be an easy task if you feel the need for some. Note that templates have extension .tex, but would work perfectly well as PreTeXt templates.

12 Syntax Coloring

Examine the “minimal” and “sample-article” files to see how syntax coloring works. Notice that TeXShop uses new methods for the tags in xml, but continues to use its old methods for mathematical sections of the document.

The syntax colors TeXShop uses can be adjusted using the Themes tab of TeXShop Preferences. Five colors are used. The “Comment” color is for xml comments and the “Tag” color is for xml tags. Some tags contain extra information; for instance, the minimal.xml file contains

```
<section xml:id="section-textual">
```

In this example, xml:id receives the “Attribute” color and section-textual receives the “Value” color. Finally, some rare commands have the form < and & and in these cases, lt and amp are given the “Escape” color. Here < represents < and & represents &.

The TeXShop Themes pane in Preferences, used to change colors, is rather complicated and readers can learn about it in the Changes document in the TeXShop Help Menu. Read about changes for version 4.08, when it was introduced. Let me list some general principles here. A collection of color choices is called a “theme” and TeXShop can create and use as many themes as desired. These themes are stored as small files in `~/Library/TeXShop/Themes`. If you like a theme you have created, you can retrieve it from this location and send the file to other TeXShop users. Similarly, if someone else creates a theme you like, get their file and put it in this location.

There aren’t special themes for Lite mode and Dark mode. Instead, any theme can be used for either of these modes. At the top of the panel, the user can choose which theme to use for Lite mode and which theme to use for Dark mode.

When the Themes panel first opens, it is set to edit the theme that is active at the time. So if the computer is in Dark mode, and Dark mode uses the Manteuffel theme, then the panel is set up to edit Manteuffel. However, the third button on top left can change to a different theme to edit.

During editing, all windows use the Editing theme, and any color change made to a specific color will be instantly changed in all of these windows.

Incidentally, a new theme called “PreTeXt-Dark” is provided for dark mode in TeXShop. It is not the default dark theme, but can be selected in TeXShop Preferences. This scheme should be close to the Sublime Text syntax coloring which many people were using at the Portland Conference.

13 The Tags Item on the Toolbar

In LaTeX mode, the Tags menu lists all `\chapter`, `\section`, and `\subsection` commands in a source file, identified by associated text. Selecting an item takes the user to that location in the source. The Tags menu works the same way in xml mode, but this time it lists all `<chapter>`, `<section>`, `<subsection>`, etc. locations, with associated text. Indeed, the command recognizes eleven distinct tags. Since this is a large number, it can cause very long scrollable Tags menus. So under the Misc tab in TeXShop Preferences, an item lists all eleven possibilities and allows the user to select the items that create entries in the menu.

It is possible to mark your own important spots in the source using the command

```
<!--!mytext-->
```

Here “mytext” can be replaced by any desired string. These locations will be listed as tags, with accompanying text. It is not necessary to remember the syntax; just select a location in the source and choose the Macro “PreTeXt Personal Mark” to insert appropriate text.

14 Some Keyboard Tricks

Two new keystroke commands are available to aid in text entry. Here is the first. Suppose `<hello>` is an xml tag. Double click in the “hello” while holding down the option key to select all text between this tag and its associated `</hello>`. This action understands xml comments and will ignore text inside these comments. This technique also works in reverse; double click on “hello” in `</hello>` while holding down the option key and TeXShop will find the associated key `<hello>` and select all text between these tags.

The second keystroke command is called “Close Tag” and is initiated by a TeXShop menu item of the same name. The command has a keyboard shortcut `option-command-period` for easy entry. This command will search for an unclosed tag like `<hello>` and close it with `</hello>` at the cursor spot. The command understands xml comments and ignores text in such comments.

This command is often used by PreTeXt authors, who insert a new tag and then click on the appropriate closing spot and type `option-command-period` to insert its close.

Close Tab works by searching backward until it finds an open tag that has not been closed. Note that “the rules of xml” limit the spots where commands can be legally closed. If a user tries to close a command at an illegal spot, a different closing tag will almost always be entered at that spot. When that happens, back up and think carefully about what you want to do.

For example, consider the text below, where `<hello>` has no associated closing tag. Notice that the initial `<hello>` can be completed immediately after it occurs, or after the ending `</sage>`, or after the following `</p>`, but not elsewhere. If you try to close it after `A.rref()`, Close Tag backs up and finds that `<input>` has not been closed, so it closes that rather than `<hello>`.

```

<section>
  <hello>
    <sage>
      <input>
      A = matrix(4,5, srange(20))
      A.rref()
      </input>
      <output>
      [ 1  0 -1 -2 -3]
      [ 0  1  2  3  4]
      [ 0  0  0  0  0]
      [ 0  0  0  0  0]
      </output>
    </sage>

    <p>This is extra text.</p>

</section>

```

15 Command Completion

Finally a new completion dictionary is provided for xml files, listing phrases likely to occur in a PreTeXt document. When an xml file is opened, TeXShop will automatically switch to this dictionary, and the menu command “Edit Command Completion File” will open this dictionary.

Command completion works as it already does for LaTeX files. Recall the idea. Type a few letters like `<sec` and push a magic key. This can be either the Escape key or the Tab key, as selected in TeXShop Preferences. I’ll suppose the Tab key has been selected. TeXShop will then complete the command using one of the selections which matches `<sec`, like `<section>` or `<section xml-id=“●”>`. Pushing Tab again will cycle through all possible completions, including the original `<sec`.

Interruption: Sometimes Apple makes mistakes and valiantly stands their ground as users complain. The following two paragraphs of the original document have been kept to illustrate this point. They are about the machine I owned at the time, a 2016 Macbook Pro

with Touchbar and the Butterfly Keyboard. Last fall I replaced this machine with a 2021 16" Arm Macbook Pro, the best machine I have ever owned by a long shot, and all these problems went away.

Aside: Command completion originally used the Escape key. A few users asked to use Tab instead, so the choice was added to TeXShop Preferences. I didn't understand the request, but I made the change anyway.

Recent events changed my mind. My main work machine is a 2016 Macbook Pro with 2 TB of flash memory. It has great features, not so great features, and blah features. The flash memory is very fast, and there is enough of it that I have all the operating systems MacTeX currently supports on the machine: Sierra, High Sierra, Mojave, and Catalina. When I travel, I unplug the external display (which also supplies power) and take my entire computing life with me. Wonderful.

The not so great feature is the butterfly keyboard, and if Apple would release a portable without it, I'd snap it up in a minute. Grrrr.

The blah feature is the Touch Bar, and to get right down to the nitty-gritty, the Escape key is now not a key at all, but a touch spot on the Touch Bar. It isn't nearly as stable as an actual key, even if that key uses the butterfly mechanism. So if you have such a portable, I recommend using the Tab (actual) key, rather than the Escape (fake) key, for command completion. End of Aside.

16 More on Command Completion

If course it would be a shame to lose the ability to enter tabs, so completion only occurs if certain rules are followed. The material to be completed, say <sec, can start at the left margin or follow a whitespace character like a space. Moreover, this material must have at least one possible completion. So typing " sec" and pushing Tab gives a Tab. In practice Tab almost always gives a tab if you want a tab, and gives a completion if you want a completion.

The completion might be just one word, but often it is a phrase, or even a collection of several lines of text. The cursor may appear after the full completion text, but it can also appear in the middle of the completion text if the author of the completion entry in the dictionary selected such a spot. Indeed, there can be several spots where additional entry is needed, and each such spot is indicated by a small black circle called a "mark". Typing control-command-F moves to the next mark, and typing control-command-G moves to the previous mark. In this manner, a completion can produce several lines of boiler-plate text which the user can easily fill in.

Possible completions are described in a dictionary named CommandCompletionXML.txt.

TeXShop has a command which opens this dictionary for editing, so if the user thinks of a new useful completion, they can immediately add it to the list.

Perhaps an example will best illustrate how this works. Suppose a PreTeXt user types <f and then a Tab. Below is one possible completion TeXShop will propose:

```
<frontmatter xml:id="●">
  <titlepage>
    <author>
      <personname>●</personname>
      <department>●</department>
      <institution>
        <line>●</line>
        <line>●</line>
      </institution>
      <email>●</email>
    </author>
    <date>●</date>
  </titlepage>
  <abstract>
  ●
  </abstract>
</frontmatter>
```

How is this completion described in the Completion Dictionary? The first thing to realize is that each entry in this dictionary is a single (possibly very long) line.

The line for the previous entry is shown below, artificially written on several lines:

```
<frontmatter xml:id="#INS#●#INS#"> #RET##T#<titlepage> #RET#
#T##T#<author> #RET##T##T##T#<personname>●</personname> #RET#
#T##T##T#<department>●</department>
#RET##T##T##T##T#<line>●</line> #RET##T##T##T#</institution>
#RET##T##T##T#<email>●</email> #RET##T##T#</author>
#RET##T##T#<date>●</date> #RET##T#</titlepage> #RET#
#T#<abstract> #RET##T# ● #RET##T#</abstract> #RET#
</frontmatter>
```

Each black dot in the completion is a spot where the author will enter information. Although not shown here, the cursor will be placed at the first black dot. When the author begins typing, this dot will be replaced by new material. When this material is complete, type control-option-F to jump to the second dot. Start typing and the dot will be replaced by new material. Etc.

In the definition of the completion, the symbols `#INS#●#INS#` determine where the cursor will appear. Material between the two symbols will be selected, and thus will vanish when the user starts typing. Incidentally, `command-8` produces that period symbol.

The symbol `#RET#` will cause a line feed. This command repeats the white space (spaces and tabs) before the initial word of the completion, but no additional initial white space created by later stages of the completion. So a completion without `#T#` would end up as a vertically aligned series of lines.

Finally, `#T#` is a new symbol introduced for PreTeXt, and it simply causes a tab. Notice that the indented nature of the completion was created by using an appropriate number of tabs following each RETURN linefeed.

The dictionary `CommandCompletionXML.txt` is rather sparse. I need help in finding appropriate completion symbols, which I'll happily add to the dictionary. Important symbols should occur several times with more and more parameters displayed. Thus for sections the following entries, and probably several others, should be included:

```
<section>
  <section xml-i="#INS#●#INS#">
    <section xml-i="#INS#●#INS#">#RET##T#<title>●</title>
```

17 Latexmk and XeLaTeX

Currently the PreTeXt-LaTeX engine uses `pdflatex` to typeset the tex file produced by conversion from the xml source. But it can easily be edited to use `xelatex` instead. Ordinary `pdflatex` will correctly process some extended Unicode characters, but `xetex` expects unicode source files by design. So any project using extensive non-Roman fonts should switch to `xelatex`.

Similarly, the PreTeXt-LaTeX engine can be edited to use `pdflatexmk` rather than `pdflatex`. `Latexmk` is a make script which keeps track of files that have changed and typesets as many times as necessary to give updated documents. For instance, it will typeset twice if there are references that require the aux file and a second pass to correctly update.

18 End of Revision of 2019 Document

This ends my revision of the original document for PreTeXt 2022. At this point, the old document described using Java to activate the Validate Engine. I haven't used Java in years and don't want to install it on my new machine. Think of the remaining paragraphs as suggestions that might or might not work today. If you don't decide to use PreTeXt, these paragraphs won't make a difference. If you do decide to use it, you'll know more than I do after two or three days.

Here is the historical material:

19 The Validate Engine

There are rules governing the placement of tags in a PreTeXt document. For example, only selected tags are allowed between `<section>` and `</section>`; one of the prohibited tags is another `<section>` because it does not make sense to start a new section inside an existing section. That's what "subsections" are for.

Tools are available to check your xml source and report placement errors for tags. These tools consult a file in `mathbook`, `mathbook/schema/pretext.rng`, to learn the PreTeXt rules for tags. The PreTeXt authors recommend running a validation tool regularly, and recommend "jing", a Java program. TeXShop has a typesetting engine to run this specific tool. Reach up to the magic comment line near the top and change "LaTeX" or "HTML" to "Validate". If there are errors, this tool will print very clear descriptions of the errors in the console. If there are no errors, it will write a soothing message.

The jing applet is included in `~/Library/TeXShop/bin`. It was obtained from the web site <https://relaxng.org/jclark/jing.html>. In the middle of this web page, the following text occurs: "The latest version of Jing will always be available in the Downloads section of the project site." The word "Downloads" is a link; clicking it leads to a link to download "jing-20091111.zip". The jing in TeXShop is the applet provided by this link.

If you do not already have Java, then you must also download and install it. Apple used to supply Java, but now it is maintained by Oracle. Go to the web site https://www.java.com/en/download/mac_download.jsp. Click the red link titled "Agree and Start Free Download". This will produce a Macintosh install package. Double click to install. You have Java, but you are not yet done.

The Macintosh will not run this Java until you obtain a Java Runtime. Go to the site <https://www.oracle.com/technetwork/java/javase/downloads/jdk12-downloads-5295953.html>. Download the file `jdk-12.0.2-osx-x64_bin.dmg` from the table toward the bottom of the page. This will lead to another Macintosh install package. Install this package.

On April 16, 2019, Oracle substantially changed the Oracle Java License. Use of these files is permitted for personal and development use, but apparently not for businesses. The sites claim to have OpenJDK packages released under the GPL license. If this is a matter of concern, you probably should examine the sites more carefully to find the GPL versions, or read the new license carefully.

20 PreTeXt Projects with Several Source Files

Some PreTeXt projects, particularly books, tend to divide the source file into several files which are input by a master file. TeXShop has a magic line to handle this situation. Each source file *except the master file* should have an extra line at the top similar to the third line of the example below. In the TeXShop Macro menu, the item "Root (Short Version)" will produce the start of this line.

```
<!--  
% !TEX TS-program = PreTeXt-LaTeX  
% !TEX root = Main.xml  
-->
```

Once these lines are in place, you can edit any project source file. Typing command-T will save the file you are working on, and then typeset the Main file, and thus the entire project.

The Main file can be given any name desired in this magic line. Naming just the file is fine if it is in the same folder as the subfile being edited. TeXShop also understands constructions like "../Main.xml" or full paths.

LaTeX has a command called "includeonly" for the Main file, which reads the header information in this file, but only includes listed chapter files to typeset. This is useful when working on a particular chapter. I don't know if PreTeXt has a similar facility.

21 Miscellaneous Features

Users will discover other TeXShop features that work fine with PreTeXt. We mention two here. The TeXShop Window menu has an item named "Use One Window." If it is selected, the xml source and the pdf output will be placed side by side in a single window. A preference setting determines which is on the left and which is on the right. The command "Use Separate Windows" returns to two window mode.

If a project requires several source files, there are magic lines for TeXShop which tell it to open all of these source files as tabbed entries of a single source window. Read the Comments document in TeXShop Help and experiment to see how this works.