Chapter 1

Author's Guide to Diderot

Let's start with creating and uploading chapters. Let's use the "Slides" book that I created (you can create as many books as you want, see below). Go to "Manage Books" from the "Links" on the center right, to the left of search bar on top. Now you can create a chapter.

1.1 Course Home Page

The Zones. Course home page, shown below, consists of four zones.

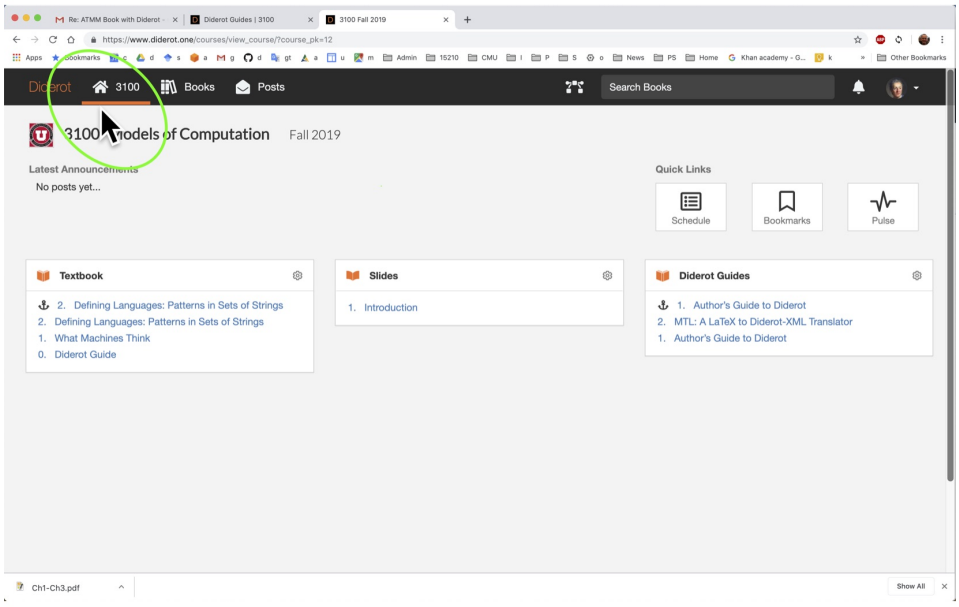
1. A toolbar, at the top

2. A "news" or "Latest Announcements" area on the top left quadrant.

3. Quick links on the right top right quadrant.

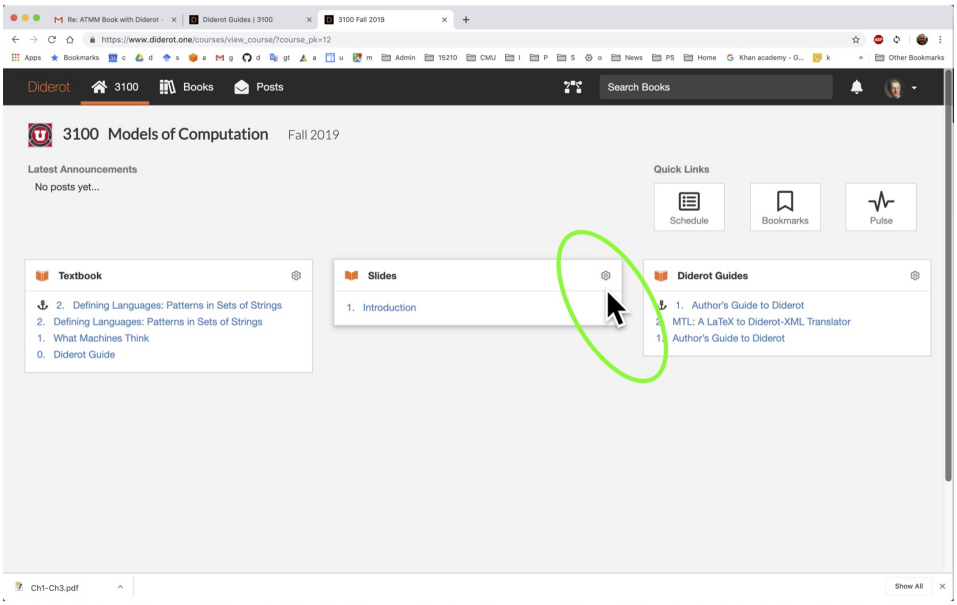
4. The books, an the lower half.

You can go to the course home page by pressing the "Home" button at the top right corner.



1.2 Book Management

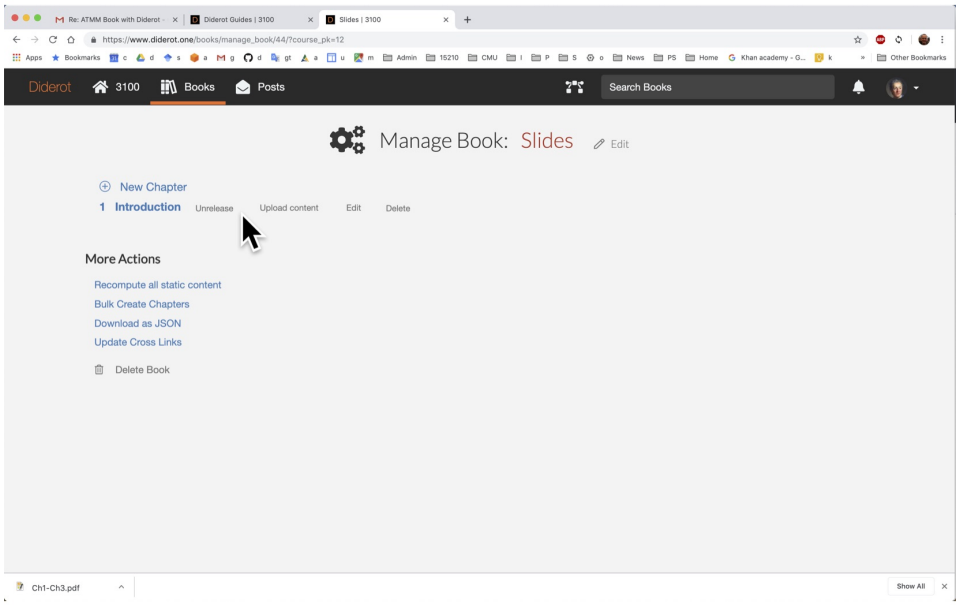
Finding Book Management. As an author, you will spend much of your time in the "Book Management" page. A quick way to get to this page is to go to the course home page and to press the settings ("wheel") icon for the book that you want to manage.



The book management page offers functionality for

- creating chapters,
- uploading content onto chapters,
- realeasing/unreleasing chapters,
- deleting chapters,
- additional more advanced operations, which we will skip for now.

To see the actions that you can perform on a chapter options, move the mouse over to that chapter.

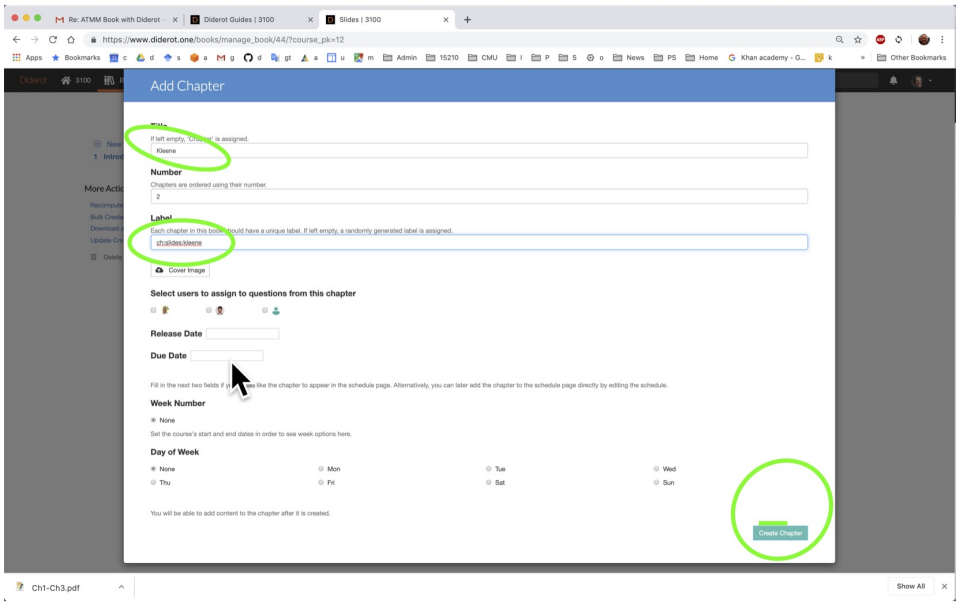


1.3 Chapters

Chapter Creation. You can add a chapter to a book.

- Go to course home page.
- Find the book that you want to edit. You can reach the book management screen from the "Manage Books" link from the "Links" button on the center right, to the left of search bar on top.
- Press "Create chapter" and follow the instructions. To create a chapter in its most basic form, you can leave all of the rest of the fields empty. The additional field are discussed below.

When creating a chapter, assign the chapter a unique number and a unique label, e.g., `ch:kleene`. The label needs to be unique only within the book—no two chapters of the same book can have the same label. If the chosen label does not meet this criteria, you will receive an error message and can choose another one.



Exercise 1.1. As an example, create now a chapter for your second slide deck from your Fall 2018 site and give it the label `ch:kleene`.

Released versus Unreleased Chapters. A chapter is either released or unreleased. ***Released*** chapters are visible to the students. ***Unreleased*** chapters are not visible to the students but are visible to course staff.

This is designed to allow for a "feedback cycle" before releasing chapters to students. For example, I typically upload my lectures a bit ahead of time and before releasing them, ask my TAs to look over them and give feedback. To give feedback, the TA's simply select the relevant atom and create a feedback, e.g., they might note a typo. I then fix these problems, reupload the chapter, and then release it.

Release Dates and Schedule. You can assign release dates to your chapters and this will automatically construct (and maintain) a schedule for you. Let's skip this step for now. So leave these fields empty. You can edit chapters to add schedule information later.

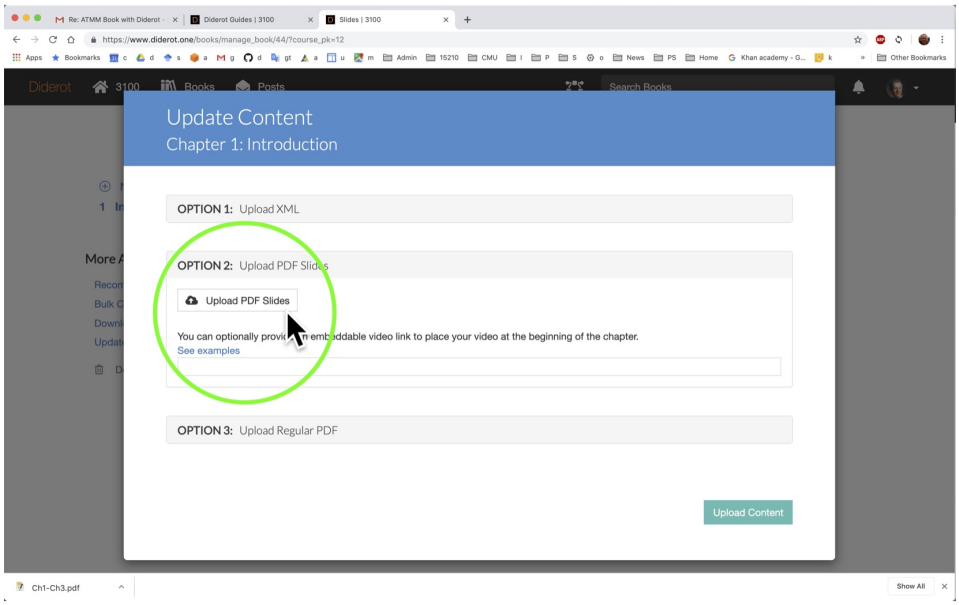
User Assignments. You can "assign" users to a chapter. The intention here is to designate ***discussion chairs***, who are usually TAs for each chapter. They will be assigned the questions asked on that content.

I don't know who I will assign the chapters to at the time of creation and therefore leave this blank. I or the TAs later edit the chapter to assign the discussion chairs.

Uploading Content. To upload contents into a chapter, move the mouse over the chapter and select upload content. Here you can upload either an XML file or a PDF. Let's start with PDF. Select the PDF from slides and select the PDF file to upload, and upload.

Now you will see a spinning wheel. This will take some time maybe a minute or two. What is going on is that the PDF is split into pages and the text is extracted from it. When it is finished, click on the chapter and you should be able to view it.

As an example, upload your second slide deck from your Fall 2018 site to the chapter that you have created above.



Important (Unreleased chapters are not searchable). When a PDF document is uploaded, its text is extracted from and stored into an index, allowing the document to be searchable via the search bar when the chapter is released. Unreleased chapters will not be searchable (so students won't see unreleased content).

Note. Currently, pdf documents themselves are shown as JPEG images. We are in the process of changing this so that they are shows an PDFs. This should be ready soon.

Interactive Chapters. Once a chapter is uploaded, it becomes interactive.

- If unreleased, course staff may send feedback and open other discussions.
- If released: all the students and staff may send feedback, ask questions, take private notes on the content, etc.

Updating a Chapter. For unreleased chapters, simply re-upload the contents. This will recreate the chapter and delete all prior discussions on it.

For released chapters, you can unrelease and re-upload, but this will delete all content, including student created content (private notes, questions, etc). Unreleasing and re-uploading could be thought as the same as deleting the chapter and recreating it. Because it deletes all related content, we discourage using this feature (after a course starts). If you don't currently have students and are developing content, you may release chapters, and re-upload them as you wish.

There are more advanced ways to upload released content. We will see these later.

1.4 Books

Create a book. You can create new books and as many as you want. I usually create one for textbook, one for recitations, one for assignments, one for additional materials (misc). Many courses create a ”Slides” book.

To create a book go to ”links” to the right of the search bar and then press manage books. You will give your book a unique label (e.g, ”textbook”, ”recitations”, ”misc”...) and a unique”rank” which is the order it will be shown but is otherwise immaterial.

By default, a book is selected to be a ***booklet***, which consists of a sequence of chapter. If you want the book to contain ”parts” as an organization element above chapters, uncheck the “booklet” box.

A book can contain PDFs or XMLs (generated by a compiler).

Example 1.1. In the algorithms (15210) course, the main textbook has parts but the rest of the books don't. So all the books except for the textbook are booklets.

Editing a Book. To edit the book, create and upload chapters, go to ”Manage Books” and select your book. Here you can edit various properties of the book, create chapters upload content

1.5 User Accounts

Creation. To create user accounts, you can upload a roster in CSV format, which will email them all. I usually wait to do that until the first day of the class or right before it but you can create accounts for your TAs a bit ahead of time so that they can start playing with things (TAs and undergrads seem to pick things up super quickly). When you create accounts, the users receive email notifications and instructions on how to log on.

Chapter 2

MTL: A LaTeX to Diderot-XML Translator

2.1 Overview

Diderot is an online book system that integrates discussions with content. Diderot consists of two largely separate systems that are designed to work together. The first is the Diderot website, which provides the users (instructors and students) with an online interface for reading books and discussions. The second is the ***MTL*** (read "metal") compiler that translates LaTeX and Markdown sources to Diderot-stple XML, which can then be uploaded onto the Diderot site. In addition to XML, Diderot site accepts conventional PDF documents and slide decks for upload. This document describes MTL and its use.

Important. To run through the examples here, please clone this repository and follow the instructions in INSTALL.md.

2.2 Typesetting with LaTex for Diderot

MTL tries to remain compatible with LaTeX. If you have LaTeX sources that you are able to compile and generate PDF from, then in most cases, you can use MTL to generate XML from your LaTeX sources. Translation of LaTeX sources to XML is not perfect at this time but works reasonably well. MTL also provides a few additional syntactic elements that helps improve the interactivity of the content.

Definition 2.1 (Books and Booklets)**.** After a LaTeX document is translated to XML via MTL, it can be uploaded as a ***chapter*** to a Diderot book or Diderot booklet. A Diderot ***book*** is a collection of parts, where each ***part*** is a collection of chapters. A Diderot ***booklet*** is a collection of chapters.

Example 2.1. The example directory contains some example LaTeX sources along with additional tools for translation to Diderot XML. The directories example/book and example/booklet

contain a sample book (consisting of parts and chapters) and booklet (consisting only of chapters) respectively. Each come with a Makefile for generating PDF and Diderot-XML from the sources.

Example 2.2 (PDF Generation)**.** You can generate the PDF for the whole book by running the following.

```
$ cd examples/book
$ make book.pdf

$ cd examples/booklet
$ make book.pdf
```

Example 2.3 (XML Generation)**.** For ease of publication on Diderot, XML generation occurs on a chapter by chapter basis. You can generate the XML for each chapter by specifying the chapter path.

```
$ cd examples/book
$ make probability/expectation.xml

$ cd examples/booklet
$ make skyline/main.xml
```

2.2.1 Basic Structure of LaTeX

As with Latex, MTL requires a book to be organized into chapters, each of which then contains sections (section, subsection, subsubsection, and paragraph). Each chapter and section can in turn contain "elements" each of which is an "atom" or a "group".

Example 2.4 (Sections)**.** A chapter is typically structured into a number of sections.

```
\chapter{Introduction}
\label{ch:example}  % Chapters must have a label.

<elements>

\section{A Section}
\label{sec:example} % Optional but recommended section label.

<elements>

\subsection{A Subsection}
\label{sec:example::sub} % Optional but recommended section label.

<elements>

\subsubsection {A Subsubsection}
\label{sec:example::subsub}
<elements>
```

```
\paragraph {A paragraph}
\label{sec:example::paragraph}
<elements>
```

The term element in the example refers to an ”atom” or a ”group”.

2.2.2 Atoms

Definition 2.2 (Atom)**.** An ***atom*** is either

1. a plain paragraph, or
2. a single-standing environment of the form

   ```
   \begin{<atom>}[Optional title]
   \label{atom-label} % optional but recommended label
   <atom body>
   \end{<atom>}
   ```

   The term <atom> above can be replaced with any of the following:

   - algorithm, assumption,
   - code, corollary , \ lstinline costspec‘,
   - datastr (data structure), datatype, definition
   - example, exercise,
   - gram (non descript atom, i.e., a paragraph),
   - hint,
   - important,
   - lemma,
   - note,
   - preamble (as a first atom of chapter), problem (a problem for students to solve), proof, proposition,
   - remark (an important note), reminder,
   - solution (a solution to an exercise/problem), syntax (a piece of syntax)
   - task (a task in an assignment), theorem.

Note. Authors can currently use only these atoms defined above. We are working on a way to allow the authors to define their own atoms and expect it to be available soon. In the mean time, you can request new atoms (please send feedback here).

Important. Atoms are ***single standing***, that is to say surrounded by ”vertical white spaces” or empty lines on both ends. Therefore, white space matters. In the common case, this goes along with our intuition of how text is organized but is worth keeping in mind. For example, the following code will not be a definiton atom, but will be a plain paragraph atom, because definition is not single standing.

```
We can now define Kleene closure as follows.
\begin{definition}
...
\end{definition}
```

The following is a definition atom, because it is single standing.

```
\begin{definition}
...
\end{definition}
```

Note. Atoms can contain multiple paragraphs. The following text consists of a text-paragraph atom and a definition atom.

```
Paragraph 0. Sentence 1.
Sentence 2.

\begin{definition}[Definition Title]
Paragraph 1

Paragraph 2
\end{definition}
```

Controlling granularity

MTL will create an atom for each text paragraph. If a piece of text contains many small (one or two line) paragraphs, it can distracting for the user. For example, the following text consists of three small paragraphs.

```
If this then that.

If that then this.

This if and only of if that.
```

We usually don't write like this but sometimes it happens. In such a case, you may want to wrap this text into a single paragraph atom.

```
\begin{gram}[If and only If]
If this then that.

If that then this.

This if and only of if that.
\end{gram}
```

Alternatively you can wrap the text by curly braces as follows.

```
{
If this then that.

If that then this.

This if and only of if that.
}
```

Both will have no impact on the PDF but on Diderot, you will have only one atom for the three sentences.

2.2.3 Groups

Group. A ***group*** consist of a sequence of atoms. MTL currently support only one kind of group: flex. On Diderot, a flex will display its first atom and allow the user to reveal the rest of the atoms by using a simple switch. We find flex groups useful for hiding simple examples for a definition, the solution to an exercise, and sometimes tangential remarks. This has turned out be a favorite feature of Diderot for authors and students alike.

```
\begin{flex}
\begin{definition}[A Definition]
\label{def:a}

A definition
\end{definition}

\begin{example}[Simple Example I]
\label{ex:a-simple}
Simple example 1
\end{example}

\begin{example}[Simple Example II]
\label{ex:a-simple-2}
Simple example 2
\end{example}

\end{flex}
```

You can see how this flex example works below, where a definition atom has been paired flexibly with two examples. Click the drawer icon at the bottom right to open the flex.

Definition 2.3 (A Definition)**.** A definition.

Example 2.5 (Simple Example)**.** Simple example 1

Example 2.6 (Simple Example)**.** Simple example 2

2.3 Labels and References

Diderot uses labels to identify atoms uniquely. It is a good practice to try to give a label to each atom, group, and segment. All labels in a book must be unique. MTL generates labels for all segments, groups, and atoms even if you don't give them one. To help in authoring, I recommended giving each chapter a unique label, and prepending each label with that of the chapter.

Example 2.7. I recommend labeling chapters as follows.

```
\chapter{Introduction}
\label{ch:intro}

\begin{preamble}
\label{prml:intro}
...
\end{preamble}

\section{Overview}
\label{sec:intro::overview}


Here is a paragraph atom without a label.


\begin{gram}
\label{grm:intro::present}
In this  section, we present...
\end{gram}

Here is another paragraph atom, consisting of two environments:
\begin{itemize}
...
\end{itemize}
\begin{enumerate}
...
\end{enumerate}
```

References. To reference a label you can either use

- \href{label}{ref text}
- \ref{label}.

MTL replaces the former with '[]' command so that we can get proper linked refs is latex/pdf.

When auto-generating labels, MTL uses different prefixes for labels: sec for all sections, grp for groups, and the following for atoms. Atoms and their labels are shown below.

```
algorithm : "alg"
assumption : "asm"
code : "cd"
corollary : "crl"
costspec : "cst"
datastr : "dtstr"
datatype : "adt"
definition : "def"
example : "xmpl"
exercise : "xrcs"
hint : "hint"
important : "imp"
lemma : "lem"
note : "nt"
gram : "grm"
preamble : "prmbl"
problem : "prb"
proof : "prf"
proposition : "prop"
remark : "rmrk"
reminder : "rmdr"
slide : "slide"
solution : "sol"
syntax : "syn"
task : "tsk"
theorem : "thm"
```

2.3.1 Code

For code, you can use \lstinline and the lstlisting environment. The language has to be specified first (see below for an example). The Kate language highligting spec should be included in the "meta" directory and the name of the file should match that of the language. For example if language = C, then the Kate file should be meta/C.xml. If the language is a dialect, then, e.g., language = \{[Cdialect]C}, then the file should be called CdialectC. Kate highlighting definitions for most languages are available online. If you need a custom language, you can probably write one with a bit of effort by starting with the Kate specification for a suitably close language.

Example 2.8 (Python Code)**.** Specify language as python, and include line-numbers on the left of each line. Don't forget to include the Kate highlighting file for python in the meta directory.

```
\begin{lstlisting}[language = python, numbers = left]
def is_even (i):
```

```
  if i %2 = 0:
    return true
  else
    return false
\end{lstlisting}
```

The code above will render like this:

```
1 def is_even (i):
2   if i %2 = 0:
3     return true
4   else
5     return false
```

Example 2.9 (Code in C Dialect)**.** Specify the dialect preferred, and don't include line-numbers. Don't forget to include the Kate highlighting file for the dialect in the meta directory.

```
\begin{lstlisting}[language = {[C0]C}]
main () {
  return void
}
\end{lstlisting}
```

2.3.2 Images

You can include JPEG or PNG images by using the usual includegraphics command. We currently don't support PDF images (will be available shortly).

There is one point to be careful about: sizing. In principle, you could use fixed sizes, e.g.,

```
\includegraphics[width=5in]{myimage.jpg}
```

The problem with this approach is that the PDF output and the Diderot output will likely have different formats and the image that looks just fine on paper might look too big on Diderot or possibly vice versa. I therefore recommend using relative widths using the following approach

```
\includegraphics[width=0.5\textwidth]{myimage.jpg}
```

MTL will translate this to 50% width in html and 50% of textwidth in PDF and the image will look consistent with its environment in both cases.

Note. For PDF output, the author could also use \textheight. But this is not meaningful for HTML output, and will not work as expected.

Important (Scale is not Supported). For PDF output it is also possible to use scale. For example,

```
\includegraphics[scale=0.5]{myimage.jpg}
```

This means that the image should be shown at 50% of its actual dimensions. This has the same issues as the absolute measures approach. Furthermore, MTL doesn't detect attempt to detect the actual dimensions of the image and this approach is not supported.

2.3.3 Colors

You can use colors as follows

```
\textcolor{red}{my text}
```

2.3.4 Limitations

LaTeX has become rich but many author tend to use a small subset. MTL appears to work well for most uses. Here is a list of known limitations.

- For XML translation work, the chapter should be compileable to PDF.
- Do not use \input directives in your chapters.
- Each chapter must have a unique label.
- Fancy packages will not work. Stick to basic latex and AMS Math packages.
- Support for tabular environment is limited: borders don't work, neither does columnt alignmen, columns are centered. You can use the array (math/mathjax) as a substitute. This could require using \mbox for text fields.
- Center environment doesn't work.
- You can use itemize and enumerate in their basic form. Changing the label format with enumitem package and similar packages do not work. You can imitate these by using heading for your items.
- Labeling and references are limited to atoms. You can label atoms and refer to them, but you cannot label codelines, items in lists, etc.
- We use mathjax to render math environments. This works in many cases, especially for use that is consistent with AMS Math packages. There are a few important caveats.
  - Once you switch to math, try to stay in math. You can switch to text mode using \mbox but if you use macros inside the mbox, then they might not work (because mathjax doesn't know about your macros). For example, the following won't work.

    ```
    $\lstinline 'xyz'$
    ```

- The "tabular" environment does not work in MathJax. Use "array" instead.
- The environment

```
\begin{alignat}
...
\end{alignat}
```

  should be wrapped with \htmlmath, e.g.,

```
\htmlmath{
\begin{alignat}
...
\end{alignat}
}
```

2.4 How to Compile Using MTL

The following instructions are tested on Mac OS X and Ubuntu. The binaries in 'bin' might not work on systems that are not Mac or Linux/Unix-like. Several examples are provided in examples.

The book and booklet directories contain several files.

- templates/diderot.sty

  Supplies diderot definitions needed for compiling latex to pdf's. You don't need to modify this file.

- templates/preamble.tex

  Supplies your macros that will be used by generating a pdf via pdflatex. Nearly all packages and macros should be included here. Each chapter will be compiled in the context of this file. Ideally this file should - include as few packages as possible - define no environment definitions - macros should be simple

- templates/preamble−diderot.tex

  Equivalent of preamble.tex but it is customized for XML output. This usually means that most macros will remain the same but some will be simplified to work with 'pandoc'. If you don't need to customize, you can keep just one preamble. The example in directory 'booklet' does so.

2.4.1 Structuring your books sources

I recommend structuring your book sources in a way that streamlines your workflow for PDF generation and Diderot uploads. I have found that the structure outlined below separately for booklets and books work well. The example book and booklet provided follow this structure (see directories 'book' and 'booklet').

Booklets

Booklets are books that don't have parts. For these I recommend creating one directory per chapter and placing a single main.tex file to include all contain that you want. Place all media (images, videos etc) under a media/ subdirectory.

- ch1/main.tex
- ch1/media/: all my media files, *.png *.jgp, *.graffle, etc.
- ch2/main.tex
- ch2/media/: all my media files for chapter 2, *.png *.jgp, *.graffle, etc.
- ch3/main.tex

Books

Books have parts and chapters. I recommend structuring these as follows, where 'ch1, ch2' etc can be replaced with names of your choice.

- part1/ch1.tex
- part1/ch2.tex
- part1/media−ch1/
- part1/media−ch2/
- part2/ch3.tex
- part2/ch4.tex
- part2/ch5.tex
- part2/media−ch3/
- part2/media−ch4/
- part2/media−ch5/

2.4.2 Making PDFs

You can use pdflatex to generate PDFs. See the Makefile in book or booklet as examples. For example, you can invone the Makefile as follows to make a PDF:

```
$ make book.pdf
```

2.4.3 Making PDF a Specific Chapter

To make specific chapters, I usually extend separate rules for them in the Makefile. See the Makefile in book or booklet as examples. For example, to compile the chapter probability in the book directory usen

```
$ make ch2
```

2.4.4 Making XML of a specific chapter

```
$ make ch2/main.xml
```

Error messages from the XML translator are not useful. But, if you are able to generate a PDF, then you should be able to generate an XML. If you encounter a puzzling error try the "debug" version which will give you an idea of where it blew up.

```
$ make ch2/main.xmldbg
```

Assuming that you structure your book as suggested above, then you will mostly be using the Makefile but you could also use the MTL tools directly.

texml. This tools translates the given input LaTeX file to xml.

```
$ texml -meta ./meta -preamble templates/preamble.tex input_file.tex -o output_file
```

The meta direcotry contains some files that may be used in the xml translation. You can ignore this directory to start with and then start populating it based on your needs. The main file that you might want to add are Kate highlighting specifications to be used for highlighting code.

texml.dbg. This tools is the "debug" version of the texml binary above. As you might notite, 'texml' doesn't currenty give reasonable error messages. The debug version prints out the text that it parses, so you can have some sense of where things have gone wrong. As you will likely experience, 'texml' should work if your latex sources are otherwise correct (you can run them through pdflatex), so hopefully, you will not have to use this binary much.

```
texml -meta ./meta -preamble templates/preamble.tex input_file.tex -o output_file.xm
```