Abstract

“Ximera begins where \TeX\ ends.” The ximera class aids in the creation of handouts, worksheets, exercises, and sections of textbooks to be used either individually or “glued” together via a xourse file. All ximera documents can be deployed in an online interactive form via xake See: Ximera Project and the source code on GitHub.

1 Introduction

Ximera, pronounced “chimera,” (Ximera: Interactive, Mathematics, EResources, for All) is an open-source platform that provides tools for authoring and publishing (PDF and Online), open-source, interactive educational content, such as textbooks, assessments, and online courses. The Ximera document class provides the following features:

Formatting for different domains The Ximera document class provides built-in support for formatting documents in both PDF and online formats, which can be a big time-saver for authors. Additionally, it allows for the simultaneous creation of solution manuals and teaching editions, which can be especially useful for educators.

Compiling individually or as a whole With the Ximera document class, authors can easily compile individual documents or an entire collection of documents. This flexibility can be helpful when making changes to specific documents without having to re-compile the entire collection. Moreover, this allows an author to share large portions of a text with another, with minimal changes.

Interactive content The Ximera document class allows for the inclusion of interactive content, such as answer boxes that are validated by a client-side computer algebra system. Additionally, it allows for the embedding of YouTube videos, Desmos graphs, and GeoGebra interactives.

All content displayed By default, the Ximera document class displays all content to the author. This means the author see what the students see, along with answers and solutions, and links (that can be checked) to various interactive elements (when deployed, the interactive elements are truly embedded). This can be especially helpful for catching errors or inconsistencies in the content.

Online examples can be found at

https://go.osu.edu/ximera-examples

*This file describes version v1.5.1, last revised 2024/05/12.
2 ximera.cls

2.1 Options for the class

We start by listing the options for the ximera document class. Note, since the xourse class is based on the ximera class, all listed options are available there too.

\begin{quote}
\verb|\langle* |classXimera\rangle|
\end{quote}

\textbf{handout} The default behavior of the class is to display all content. This means that if any questions are asked, all answers are shown. Moreover, some content will only have a meaningful presentation when displayed online. When compiled without any options, this content will be shown too. This option will supress such content and generate a reasonable printable “handout.”

\begin{verbatim}
\newif\ifhandout
\handoutfalse
\DeclareOption{handout}{\handouttrue}
\end{verbatim}

\textbf{noauthor} By default, authors are listed at the bottom of the first page of a document. This option will supress the listing of the authors.

\begin{verbatim}
\newif\ifnoauthor
\noauthorfalse
\DeclareOption{noauthor}{\noautortrue}
\end{verbatim}

\textbf{nooutcomes} By default, learning outcomes are listed at the bottom of the first page of a document. This option will supress the listing of the learning outcomes.

\begin{verbatim}
\newif\ifnooutcomes
\nooutcomesfalse
\DeclareOption{nooutcomes}{\nooutcomestrue}
\end{verbatim}

\textbf{instructornotes} This option will turn on (and off) notes written for the instructor.

\begin{verbatim}
\newif\ infringestornotes
\instructornotesfalse
\DeclareOption{instructornotes}{\instructornotestrue}
\end{verbatim}

\textbf{noinstructornotes} This option will turn off (and on) notes written for the instructor.

\begin{verbatim}
\DeclareOption{noinstructornotes}{\instructornotestrue}
\end{verbatim}

\textbf{hints} When the handout options is used, hints are not shown. This option will make hints visible in handout mode.

\begin{verbatim}
\newif\ifthints
\hintsfalse
\DeclareOption{hints}{\hintstrue}
\end{verbatim}

\textbf{newpage} This option will start each problem-like environment (exercise, question, problem, and exploration) start on a new page.

\begin{verbatim}
\newif\ifnewpage
\newpagefalse
\DeclareOption{newpage}{\newpagetrue}
\end{verbatim}

\textbf{numbers} This option will number the titles of the activity. By default the activities are unnumbered.

\begin{verbatim}
\newif\ifnumbers
\numbersfalse
\DeclareOption{numbers}{\numberstrue}
\end{verbatim}

\textbf{wordchoicegiven} This option will replace the choices shown by \texttt{wordChoice} with the correct choice. No indication of the \texttt{wordChoice} environment will be shown.

\begin{verbatim}
\newif\ifwordchoicegiven
\wordchoicegivenfalse
\DeclareOption{wordchoicegiven}{\wordchoicegiventrue}
\end{verbatim}

\textbf{firstinlinechoice}\% Support for other wordchoice command contents.

\begin{verbatim}
\DeclareOption{firstinlinechoice}{\firstinlinechoicetrue}
\end{verbatim}
2.2 Loading packages

Since we want \cancel to work, we load it here to avoid polluting the .jax output.

\RequirePackage{cancel}

Quite a few packages are required by the document class. This is a list of required packages. As packages are added to this list, we should include a comment as to where they are being utilized. This will help keep this list from being redundant and/or outdated.

\RequirePackage{enumitem}
\RequirePackage{titlesec}
\RequirePackage{titletoc}
\RequirePackage{titling}
\RequirePackage{url}
\RequirePackage{amssymb}
\RequirePackage{amsmath}
\RequirePackage{amsthm}
\RequirePackage{xifthen}
\usepgfplotslibrary{groupplots}
\usetikzlibrary{calc}
\RequirePackage{fancyvrb}

Load forloop for the problem environment dynamic naming and building.

\RequirePackage{forloop}

Now we load even more packages.

\RequirePackage{environ}% Included to allow saving of environment contents. This does *not* play well with various ... and would be great to find a way around utilizing this package. One option is the "Suppress" command included below.
\RequirePackage{amssymb}% Included to have access to math typeset.
\RequirePackage{amsmath}% Included to have access to math typeset.
\RequirePackage{amsthm}% Included to have access to math typeset.
\RequirePackage{xifthen}% http://ctan.org/pkg/xifthen
Various packages must be loaded early to avoid polluting the .jax file.

To avoid weird margins in 2-sided mode, change the margins.

On the HTML side, there is more complicated page setup to perform.

Disable certain ligatures in HTML.

I am not sure what this does.
2.4 Structure

2.4.1 Macros

Makes everymath display style even when inline, could be optional.
\begin{verbatim}
\everymath{\displaystyle}
\end{verbatim}

Ok not everything, we also need to configure “display style” limits.
\begin{verbatim}
\let\prelim\lim
\renewcommand{\lim}{\displaystyle\prelim}
\end{verbatim}

2.4.2 Theorem and theorem-like environments

On the web, a theorem is emitted as a special <div>.
\begin{verbatim}
\newcommand{\ConfigureTheoremEnv}{\refstepcounter{problem}}\ifthenelse{\equal{##1}{}}{}{\HCode{<span class="theorem-like-title">}##1\HCode{</span>}}{}\ConfigureEnv{\#1}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="theorem-like problem-environment \#1" id="problem\arabic{identification}">}}{\HCode{</div>}}\IgnoreIndent}{}
\end{verbatim}

The key is to make sure that the theorem environments are defined in a corresponding fashion on the web and on paper.

\begin{verbatim}
\newtheorem{theorem}{Theorem}
\ConfigureTheoremEnv{theorem}
\newtheorem{algorithm}{Algorithm}
\ConfigureTheoremEnv{algorithm}
\newtheorem{axiom}{Axiom}
\ConfigureTheoremEnv{axiom}
\newtheorem{claim}{Claim}
\ConfigureTheoremEnv{claim}
\newtheorem{conclusion}{Conclusion}
\ConfigureTheoremEnv{conclusion}
\newtheorem{condition}{Condition}
\ConfigureTheoremEnv{condition}
\newtheorem{conjecture}{Conjecture}
\ConfigureTheoremEnv{conjecture}
\newtheorem{corollary}{Corollary}
\ConfigureTheoremEnv{corollary}
\newtheorem{criterion}{Criterion}
\ConfigureTheoremEnv{criterion}
\end{verbatim}
2.4.3 Enumerate fixes

Make enumerate use a letter

\begin{verbatim}
\renewcommand{\theenumi}{\textup{(\alph{enumi})}}
\renewcommand{\labelenumi}{\theenumi}
\renewcommand{\theenumii}{\textup{(\roman{enumii})}}
\renewcommand{\labelenumii}{\theenumii}
\end{verbatim}

2.4.4 Proofs

proof A mathematical proof environment.

\begin{verbatim}
\renewcommand{\qedsymbol}{$\blacksquare$}
\renewenvironment{proof}[1][\proofname]{\begin{trivlist} \item\hskip \labelsep \itshape \bfseries #1\hspace{2ex}}{\qed\end{trivlist}}
\end{verbatim}

2.4.5 Problem environments

These are problem environment decorations (these should be user invoked, not default). The decoration for these environments were inspired by http://tex.stackexchange.com/questions/11098/nice-formatting-for-theorems

\begin{verbatim}
\providecommand{\latexProblemContent}{#1}
\Make@Counter{Iteration@probCnt}
\newcommand{\hang}{% top theorem decoration
\begin{picture}(0,0)(1.5,0)
\linethickness{1pt} \color{black!50}%
\put(-3,2){\line(1,0){206}}% Top line
\multido{\iA=2+-1,\iB=50+-10}{5}{% Top hangs
\color{black!\iB}%
\put(-3,\iA){\line(0,-1){1}}% Top left hang
%\put(203,\iA){\line(0,-1){1}}% Top right hang
}
\end{picture}%
\endgroup%}
\newcommand{\hung}{% bottom theorem decoration
\begingroup%
\setlength{\unitlength}{.005\linewidth}% \linewidth/200
\begin{picture}(0,0)(1.5,0)
\linethickness{1pt} \color{black!50}%
\put(60,0){\line(1,0){143}}% Bottom line
\multido{\iA=0+1,\iB=50+-10}{5}{% Bottom hangs
\color{black!\iB}%
\put(-3,\iA){\line(0,1){1}}% Bottom left hang
\put(203,\iA){\line(0,1){1}}% Bottom right hang
\put(\iB,0){\line(60,0){10}}% Left fade out
}
\end{picture}%
\endgroup%}
\end{verbatim}

latexProblemContent Added for those that want to use UF problems without using the problem filter code. This command is renewed into something meaningful in the 'ProblemSelector.sty'.

\begin{verbatim}
\providecommand{\latexProblemContent}{#1}
\Make@Counter{Iteration@probCnt}
\newcommand{\hang}{% top theorem decoration
\begin{picture}(0,0)(1.5,0)
\linethickness{1pt} \color{black!50}%
\put(-3,2){\line(1,0){206}}% Top line
\multido{\iA=2+-1,\iB=50+-10}{5}{% Top hangs
\color{black!\iB}%
\put(-3,\iA){\line(0,-1){1}}% Top left hang
%\put(203,\iA){\line(0,-1){1}}% Top right hang
}
\end{picture}%
\endgroup%}
\newcommand{\hung}{% bottom theorem decoration
\begingroup%
\setlength{\unitlength}{.005\linewidth}% \linewidth/200
\begin{picture}(0,0)(1.5,0)
\linethickness{1pt} \color{black!50}%
\put(60,0){\line(1,0){143}}% Bottom line
\multido{\iA=0+1,\iB=50+-10}{5}{% Bottom hangs
\color{black!\iB}%
\put(-3,\iA){\line(0,1){1}}% Bottom left hang
\put(203,\iA){\line(0,1){1}}% Bottom right hang
\put(\iB,0){\line(60,0){10}}% Left fade out
}
\end{picture}%
\endgroup%}
\end{verbatim}
Configure environment configuration commands

The command `\problemNumber` contains all the format code to determine the number (and the format of the number) for any of the problem environments.

```
\newcommand{\problemNumber}{% % First we determine if we have a counter for this question depth level.\ifcsname c@depth\Roman{problem@Depth}Count\endcsname% Check to see if counter exists\%If so, do nothing.\else% If not, create it.\\expandafter\newcounter{depth\Roman{problem@Depth}Count}\expandafter\setcounter{depth\Roman{problem@Depth}Count}{0}\fi\\expandafter\stepcounter{depth\Roman{problem@Depth}Count}\arabic{depthICount}% The first problem depth, what use to be |\theproblem|. 258 \forloop{Iteration@probCnt}{2}{\arabic{Iteration@probCnt} < \numexpr \value{problem@Depth} + 1 \relax}{% .\expandafter\arabic{depth\Roman{Iteration@probCnt}Count}% Get the problem number of the next depth level and append it with a ".".\}\@ifpackageloaded{shuffle}{<true>}{<false>}% Check if Shuffle has been added. If so, add special numbering. Currently commented out while we decide what that should look like (compared to the normal nested problems).\ifhandout% Currently handout mode doesn’t allow hints. Putting this code in place in case that changes.\% \theproblem\%\else\% \theproblem\\%\fi\}}}}
```

```
\newcommand{\problemEnvironmentStart}[2]{% This takes in 2 arguments.\% The first is optional and is the old optional argument from existing environments.\% This is passed down to the associated problem environment name in case you want a global value.\% The second argument is mandatory and is the name of the ‘problem’ environment,\% such as problem, question, exercise, etc.\% It then configures everything needed at the start of that environment.\stepcounter{problem@Depth}% Started a problem, so we’ve sunk another problem layer.\def\spaceatend{#1}% \begin{trivlist}% \item% \hskip\labelsep\sffamily\bfseries\#2 %\arabic{problemNumber}% Determine the correct number of the problem, and the format of that number.\%\}(\spaceatend\unskip)\%}% %\%\% Configure environments start content \newcommand{\problemEnvironmentEnd}{% This configures all the end content for a problem.\% First we need to see if we’ve dropped fully out of a depth level,\% so we can reset that counter back to zero for the next time we enter that depth level.
```
Use an “identification” counter to assign IDs to the various problem-related DOM elements

Use an “identification” counter to assign IDs to the various problem-related DOM elements

\newcommand{\ConfigureQuestionEnv}[2]{
\refstepcounter{problem}\%
\renewenvironment{#1}{\refstepcounter{identification}}{}%
\ConfigureEnv{#1}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div role="article" class="problem-environment #2" id="problem\arabic{identification}">}}{\ifvmode \IgnorePar\fi \EndP\HCode{</div>}\IgnoreIndent}{}{}}%
\ConfigureQuestionEnv{problem}{problem}
\ConfigureQuestionEnv{exercise}{exercise}
\ConfigureQuestionEnv{question}{question}
\ConfigureQuestionEnv{exploration}{exploration}
\ConfigureQuestionEnv{hint}{hint}
%%%%\ConfigureQuestionEnv{shuffle}{shuffle}

2.4.6 Hints

Hint environments can be embedded inside problems.

Create a counter that will track how deeply nested the current hint is
\newcounter{hintLevel}
\setcounter{hintLevel}{0}
Create an empty shell to renew
\newenvironment{hint}{}{}

Now we renew the environment as needed, this should allow support for any transition code that treats some parts as a "handout" and some parts as non-handout. renewing the environment on the fly is a bit hacky.
\setcounter{hintLevel}{0}
\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}
\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

Step up hint level to track the nested level of the hint. This will be used for problem numbering.
\stepcounter{hintLevel}
\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

Detract from hint level counter to track hint nested level
\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}

\ifhints\renewenvironment{hint}{\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]}{\end{trivlist}}\fi

\ifhandout\egroup\ignorespacesafterend\else\end{trivlist}\fi

\addtocounter{hintLevel}{-1}
2.4.7 Solution

The solution to a problem.

\begin{solution}
\end{solution}

2.4.8 Code listing environments

A code answer environment You cannot use Environ with the fancyvrb/listings package if you want nested environments.

\begin{code}
\end{code}

A python answer environment You cannot use Environ with the fancyvrb/listings package if you want nested environments

\begin{python}
\end{python}

A JavaScript answer environment Unfortunately the name javascript is already used for the actual, executed (!) JavaScript interactive.

\begin{javascriptCode}
\end{javascriptCode}

On the web, translate verbatim and lstlisting blocks into \texttt{<pre>} elements.

\begin{dialogue}
\end{dialogue}
On the web, the resulting <dl> should have an appropriate class set.

2.4.10 Instructor notes
2.4.11 Only

prompt The prompt part for mathmode

(prompt \classXimera)
2.4.12 Foldable

The package `mdframed` is used to make pretty foldable, but the `amsthm/mdframed` conflict also messes up the `.jax` file so we don’t load `mdframed` when performing the `xake` step. But even the below isn’t enough to fix this.

```
\iftikzexport\else\RequirePackage[framemethod=TikZ]{mdframed}\fi
```

Does it fold?

```
⟨∗classXimera⟩
```

```
\colorlet{textColor}{black} % since textColor is referenced below
\colorlet{background}{white} % since background is referenced below

% The core environments. Find results in 4ht file.
% pretty-foldable
%\iftikzexport
%\newenvironment{foldable}{%}
%\else
%\renewmdenv[
%  font=\upshape,
%  outerlinewidth=3,
%  topline=false,
%  bottomline=false,
%  leftline=true,
%  rightline=false,
%  leftrimargin=0,
%  innertopmargin=0pt,
%  innerbottommargin=0pt,
%  skipbelow=\baselineskip,
%  linecolor=textColor!20!white,
%  fontcolor=textColor,
```
On the web, these foldable elements could be HTML5 details and summary.

2.4.13 Leashes

*leash*  Put content inside a scrollable box.

2.5 Document metadata

2.5.1 Metadata

To encourage authors to include relevant parseable metadata in the preamble, we define some currently ignored commands.

*license*  In the preamble, use \license with an SPDX license expression.
\acknowledgement In the preamble, use \acknowledgement to credit others who contributed to the
intellectual content beside the author.

\tag In the preamble, a \tag provides a free-form taxonomy.

2.5.2 Abstract

abstract Every activity should include a short abstract.

\author Activities have authors. Warn the user if no author is provided.

2.5.3 Titles and authors

2.5.4 Authors

\title Activities have titles.
In a ximera document, redefine \maketitle and put them in a table of contents. The \phantomsection is to fix the hrefs.

\renewcommand{\maketitle}{%
\addtocounter{titlenumber}{1}%
\{\flushleft\large\bfseries \@pretitle\par\vspace{-1em}}%
\{\flushleft\LARGE\bfseries \ifnumbers\thetitlenumber\hspace{1em}\fi\@title \par} %
\vskip .6em\noindent\textit\theabstract\setcounter{problem}{0}\setcounter{section}{0}}%
\phantomsection%
\ifnumbers\addcontentsline{toc}{section}{\thetitlenumber~\@title}\
\else\addcontentsline{toc}{section}{\@title}\
\vskip .6em
\aftergroup\@afterindentfalse
\aftergroup\@afterheading}

\def\activitystyle{}%
2.5.6 Learning Outcomes

Specify a learning outcome, either at the level of a problem or an entire document in the preamble.

These can appear in either the preamble or in problem environments. With pdflatex, we produce the .oc file which includes ALL the outcomes; in the tex4ht world, we just produce spans for the specific outcomes.

2.5.7 Labels and references

Labels and refs both generate anchors. A \label can be referenced from any file in the xourse.
A \ref command can connect one TeX file to another if they are in the same course.

\begin{macro}{\ref}
\def\ref#1\if@filesw \immediate \write \@auxout {\string \ximera{\ref{#1}}} \fi}{\begingroup \setcounter {refcount}{0} \if@filesw \let \tempa = \ximera \let \ximera = \ximera@ \fi \ximera{\ref{#1}}\if@filesw \let \ximera = \tempa \fi}
\end{macro}

\newcommand{\ref}{\ximera{\ref}}
\newcommand{\alt}{\ximera{\alt}}

\section{Images}
\subsection{Images}

Place images inside an \texttt{image} environment. On paper, this centers the image. On the web, this provides additional benefits.

\begin{verbatim}
\newenvironment{image}[1][\textwidth]{\begin{center}\resizebox{#1}{!}{\BODY}\\}{\end{center}}
\end{verbatim}

Inside an \texttt{image} environment, \texttt{alt} provides alt-text for assistive technology like screen-readers.

\begin{verbatim}
\newcounter{imagealt}
\setcounter{imagealt}{0}
\renewenvironment{image}[1][\textwidth]{\stepcounter{imagealt}\\\ifvmode \IgnorePar \fi \EndP\\\HCode{<div class="image-environment" role="img" aria-labelledby="image-alt-\arabic{imagealt}">}\\}{\HCode{</div>}}
\renewcommand{\alt}[1]{\HCode{<div style="display: none;" id="image-alt-\arabic{imagealt}">}#1\HCode{</div>}}
\end{verbatim}

Although we accept many formats, SVG is preferred on the web. Since we have a different mechanism for producing alt text, we want to ignore tex4ht's own method for producing alt text.

\begin{verbatim}
\DeclareGraphicsExtensions{.jpg,.png,.gif,.svg}
\Configure{graphics*}{svg}{\Configure{Needs}{File: \Gin@base.svg}\Needs{}\\\Picture{}{\csname Gin@base\endcsname.svg \csname a:Gin-dim\endcsname\}\\}{}
\end{verbatim}

This is a hack to kill \texttt{includegraphics} commands in \texttt{documentclass{standalone}} files

\begin{verbatim}
\ifcsname ifstandalone\endcsname\ifstandalone\renewcommand{\includegraphics}[2][]{}\fi\fi
\end{verbatim}

PGF sometimes causes trouble, but we simply don't care in tex4ht mode.

\begin{verbatim}
\newcommand{\pgfsyspdfmark}[3]{}
\end{verbatim}
2.6.2 TikZ export

We generate SVGs and PNGs for any TikZ images, via the “externalize” feature of TikZ. Currently TikZ doesn’t compile natively into the website because of how the xake bake compilation works. In order to make Tikz work, you need to get the tool mutool on the machine that is performing xake bake.

\ifdefined\HCode
\tikzexporttrue
\fi
\iftikzexport
\usetikzlibrary{external}
\ifdefined\HCode
% in htlatex, just include the svg files
\def\pgfsys@imagesuffixlist{.svg}
\else
\tikzexternalize[prefix=./,mode=graphics if exists]
\else
% in pdflatex, actually generate the svg files
\tikzset{
/tikz/external/system call={
  pdflatex \tikzexternalcheckshellescape
  --halt-on-error --interaction=batchmode
  --jobname "\image" "\PassOptionsToClass{tikzexport}{ximera}\texsource";
  mutool draw -F svg \image.pdf > \image.svg ; % mutool adds "1" to filename ????
  mutool draw -o \image.svg \image.pdf ;
  mutool draw -r 150 -c rgbalpha -o \image.png \image.pdf ;
  ebb -x \image.png
}
\tikzexternalize[optimize=false,prefix=./]
\fi
\fi
\fi

2.6.3 XKCD

\xkcd Reference an XKCD cartoon.
\ifdefined\HCode
\newcommand{\xkcd}[1]{#1}
\fi

On the web, this should be an image linked to the actual XKCD website.
\ifhtXimera
\renewcommand{\xkcd}[1]{\ifvmode \IgnorePar\fi \EndP\HCode{<img src="https://imgs.xkcd.com/comics/#1.png">}}
\fi

2.7 Links

We put hyperref after all other packages because that is better.
\ifdefined\HCode
\RequirePackage{hyperref}
\else
\RequirePackage[pdfpagelabels,colorlinks=true,allcolors=blue!30!black]{hyperref}
\pdfstringdefDisableCommands{\def\hskip{}}%% quiets warning
\fi

\end{document}
2.8 Interactives

2.8.1 Including widgets
\includeinteractive
Cognate to \texttt{\texttt{includegraphics}} but instead of a graphics file, accepts a .js file which will be loaded as an interactive widget.

\define@key{interactive}{id}{\def\interactive@id{#1}}
\setkeys{interactive}{id=}
\newcommand{\includeinteractive}[2]{% \setkeys*{interactive}{#1}\
ifthenelse{\equal{\interactive@id}{}}{}{\recordvariable{\interactive@id}}
Interactive
}
\end{verbatim}
\end{code}

2.8.2 Google Sheet
\googleSheet
googleSheet command. Requires id, width, and height as arguments. Optional arguments are gid for sheet ID and range for cell range. Command definition
\begin{verbatim}
\define@key{googleSheet}{rc}{false, sdz=false, smb=false, stb=false, stbh=false, ld=false, sri=false}
\setkeys{googleSheet}{rc=false, sdz=false, smb=false, stb=false, stbh=false, ld=false, sri=false}
\renewcommand{\includegoogleSheet}[5]{% ifthenelse{\equal{#5}{}}% \HCode{<iframe width="#2px" height="#3px" src="https://docs.google.com/spreadsheets/d/#1/htmlembed?single=true&amp;gid=#4&amp;range=#5&amp;widget=false"/>% This browser does not support embedded elements.</script>}}%
\end{verbatim}

2.8.3 Geogebra
\geogebra
Geogebra command. Requires id, width, and height as arguments.
\begin{verbatim}
\define@key{geogebra}{rc}{true}\def\geo@rc{#1}
\define@key{geogebra}{sdz}{true}\def\geo@sdz{#1}
\define@key{geogebra}{smb}{true}\def\geo@smb{#1}
\define@key{geogebra}{stb}{true}\def\geo@stb{#1}
\define@key{geogebra}{stbh}{true}\def\geo@stbh{#1}
\define@key{geogebra}{ld}{false, sri=true}\def\geo@ld{#1}
\setkeys{geogebra}{rc=false, sdz=false, smb=true, stb=false, stbh=false, ld=false, sri=false}
\renewcommand{\includegeogebra}[4]{% ifthenelse{\equal{#4}{}}% \HCode{<iframe scrolling="no" src="https://www.geogebra.org/material/iframe/id/#2/width/#3/height=#4"/>}
\end{verbatim}

2.8.4 Desmos
\desmos Desmos command. Requires id, width, and height as arguments.
\newcommand{\desmos}[3]{{Desmos link: \url{https://www.desmos.com/calculator/#1}}}
\newcommand{\desmosThreeD}[3]{{Desmos3D link: \url{https://www.desmos.com/3d/#1}}}
\renewcommand{\desmos}[3]{{\HCode{<iframe src="https://www.desmos.com/calculator/#1" width="100\%" height="#3px" style="border: 1px solid \#ccc" frameborder=0>This browser does not support embedded elements.</iframe>}}}
\renewcommand{\desmosThreeD}[3]{{\HCode{<iframe src="https://www.desmos.com/3d/#1" width="#2px" height="#3px" style="border: 1px solid \#ccc" frameborder=0>This browser does not support embedded elements.</iframe>}}}

2.8.5 Graphs
\graph An embedded graph (in math mode).
\newcommand{\graph}[2]{{\text{Graph of $#2$}}}
\renewcommand{\graph}[2]{{\HCode{<div class="graph" data-options="#1">}#2\HCode{</div>}}}

2.8.6 Video
\youtube Youtube command. Requires id.
\newcommand{\youtube}[1]{{YouTube link: \url{https://www.youtube.com/watch?v=#1}}}
\renewcommand{\youtube}[1]{{\ifvmode \IgnorePar\fi \EndP\HCode{<div class="video youtube-player" data-youtube="#1">_</div>}}}

Video commands are also emitted, slightly differently, when placed at top-level in a xourse file.
\renewcommand{\youtube}[1]{{\ifvmode \IgnorePar\fi \EndP\HCode{<a class="youtube" href="https://www.youtube.com/watch?v=#1">#1</a>}\IgnoreIndent}}

2.8.7 JavaScript
\javascript Code inside a javascript environment is printed on paper, but executed on the web.
\DefineVerbatimEnvironment{javascript}{Verbatim}{numbers=left,frame=lines,label=JavaScript,labelposition=topline}
\renewenvironment{javascript}{\NoFonts}{\EndNoFonts}
\ScriptEnv{javascript}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="javascript" id="javascript\arabic{identification}"><script type="text/javascript">}
{\HtmlParOff}{\HtmlParOn\HCode{</script></div>}}
\def\js#1{\mbox{\texttt{\detokenize{#1}}}}
\def\js#1{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<span class="inline-javascript" ... function javascript\arabic{identification}() }
The \js macro is evaluated and replaced with its value.
2.9 SageMath support

Load SageTeX if it exists.

\begin{verbatim}
\IfFileExists{sagetex.sty}{\RequirePackage{sagetex}}{}
\end{verbatim}

Create an interactive SageMath widget.

\begin{verbatim}
\DefineVerbatimEnvironment{sageCell}{Verbatim}{numbers=left,frame=lines,label=SAGE,labelposition=topline}
\end{verbatim}

Execute SageMath code and output the result.

\begin{verbatim}
\DefineVerbatimEnvironment{sageOutput}{Verbatim}{numbers=left,frame=lines,label=SAGE-Output,labelposition=topline}
\end{verbatim}

Used for setting numeric answer tolerance for online student input.

\begin{verbatim}
\Define@key{answer}{tolerance}{\def\ans@tol{#1}}
\end{verbatim}

Used to run dynamic js code on student provided answers. Note: currently pdf outputs the validator code itself.

\begin{verbatim}
\Define@key{answer}{validator}{\def\ans@validator{}}
\end{verbatim}

2.10 Answerables

2.10.1 Answers

\begin{verbatim}
\answer
\end{verbatim}

A math answer

\begin{verbatim}
\ifdefined\HCode
\newcommand{\recordvariable}[1]{}
\else
\newwrite{idfile}
\immediate\openout{idfile}=\jobname.ids
\newcommand{\recordvariable}[1]{\ifthenelse{\equal{#1}{}}{}{\immediate\write{idfile}{var #1;}}}
\fi
\end{verbatim}

Determines if answer is shown in handout mode. When \texttt{given=true}, show answer in handout mode, show answer in “given box” outside handout mode. When \texttt{given=false}, do not show answer in handout mode, show answer outside handout mode

\begin{verbatim}
\define@key{answer}{given}{\def\ans@given{#1}}
\end{verbatim}

Used for setting numeric answer tolerance for online student input.

\begin{verbatim}
\define@key{answer}{tolerance}{\def\ans@tol{#1}}
\end{verbatim}

Used to run dynamic js code on student provided answers. Note: currently pdf outputs the validator code itself.

\begin{verbatim}
\define@key{answer}{validator}{\def\ans@validator{}}
\end{verbatim}

Used for assigning a js ID to answer for dynamic code (eg validators).

\begin{verbatim}
\define@key{answer}{id}{\def\ans@id{#1}}
\end{verbatim}

Used to set anticipated input format; eg "string".

\begin{verbatim}
\define@key{answer}{format}{\def\ans@format{}}
\end{verbatim}
Used to hide the answer input box on the web.

\define@key{answer}{onlinenoinput}[false]{}

Used to add a 'show answer' button to the answer blank.

\define@key{answer}{onlineshowanswerbutton}[false]{}

Set default values for \answer command key=value pairs. Default values are \texttt{given = false}.

\setkeys{answer}{id=, \texttt{given=false}, onlinenoinput=false, onlineshowanswerbutton=false}

Basic code for \answer.

\let\handoutAnswerFormat\answerFormatDots
\let\defaultAnswerFormat\answerFormatBlue
\let\givenAnswerFormat\answerFormatBoxedGiven

\newcommand{\answer}[2][false]{\ifmmode\setkeys{answer}{#1}\recordvariable{\ans@id}\ifthenelse{\boolean{\ans@given}}{\handoutAnswerFormat{#2} %% in case the argument helps formatting}{\defaultAnswerFormat{#2} %% in case the argument helps formatting}}{\GenericError{\space\space\space\space}% Throw an error based on... something? -- Jason\texttt{\{Attempt to use \backslash\char\answer outside of math mode\}}\texttt{\{See https://github.com/ximeraProject/ximeraLatex for explanation.\}}\texttt{\{Need to use either inline or display math.\}}\ifthenelse{\boolean{#2}}{\texttt{\fi}}{\texttt{\else\texttt{\fi}}}}

On the HTML side, \answer emits spans—but it is usually just handled directly by MathJax.

\newcommand{\answer}[2][false]{\HCode{<span class="answer respondable">}#2\HCode{</span>}}

\def\validator[#1]{\stepcounter{identification}\HCode{<div class="validator" id="validator\arabic{identification}"><script type="text/javascript">function validator\arabic{identification}() \{ \HCode{ return \detokenize{#1}; } \}\HCode{</script>}}
\def\endvalidator{\HCode{</div>}}

24
2.10.2 Multiple choice and the like

\begin{verbatim}
\newcommand\multipleChoice[1]{
  \if\ischoice\else\item\fi
  \if\iscorrect\else\\checkmark\fi
  \item{#1}
}\newcommand[2]{\choice{#1}{#2}}
\end{verbatim}

2.10.3 Options

\begin{verbatim}
\define@key{choice}{value}{\def\choice@value{#1}}
\define@key{multipleChoice}{id}{\def\mc@id{#1}}
\define@key{otherchoice}{value}{\def\otherchoice@value{#1}}
\define@key{otherchoice}{correct}{true}
\end{verbatim}

2.10.4 Choices

\begin{verbatim}
\item Like \item but for choice environments. choice command denotes a possible answer choice for the multiple choice question.
\end{verbatim}
On the HTML side, \choice emits <span>s.

\begin{enumerate}
\item \choice\emph{emits} <span>s.
\end{enumerate}

\newenvironment{multipleChoice}[1]\[
\begin{trivlist}
\item \[
\begin{enumerate}
\end{enumerate}
\end{trivlist}
\end{enumerate}

2.10.5 Environment(s)

The environment \texttt{multipleChoice@} is for internal use only. Wrap \choices in a \texttt{multipleChoice} environment to make a multiple choice question.
\newenvironment{multipleChoice@}{\refstepcounter{problem}}{}%\ConfigureEnv{multipleChoice@}{\stepcounter{identification}\IgnorePar\HCode{<span class="word-choice" id="word-choice\arabic{identification}">}}{\HCode{</span>}\IgnoreIndent}{}

\newenvironment{selectAll}{\begin{trivlist}\item\hskip \labelsep\small\bfseries Select All Correct Answers:\hfil\begin{enumerate}}{\end{enumerate}\end{trivlist}}

2.11 Word choice

An in-line version of multipleChoice: uses enumitem package note, it is coded as a single line to avoid unwanted spaces in “given” mode.

This is actually just word choice

2.12 Select all

A multiple-multiple choice question

In the future we need this to (optionally) be displayed in the problem, while the actual code lives in the solution. Here is how this could be implemented: Like the title/maketitle commands, the multiple-choice could be stored in themultiplechoice, flip a boolean, and execute \makemultiplechoice at the end of the problem. We should also make a command called \showchoices that will show choices in the handout.
On the web, selectAll is handled just like multipleChoice.

```latex
\renewenvironment{selectAll}{\refstepcounter{problem}}{}%
\ConfigureEnv{selectAll}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="select-all" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
```

2.12.1 Free response

A freeform input box.

```latex
\renewenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
```

```latex
\renewenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
```

```latex
\renewenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="free-response" id="problem\arabic{identification}"}}{\HCode{</div>\IgnoreIndent}{\ifhandout
\newenvironment{freeResponse}{\refstepcounter{problem}}{}%
```

2.12.2 Feedback

**feedback** An initially hidden environment that uncovers itself at an appropriate time. New Validator rewrite code added by Jason Nowell. Original code or provided by Jim Fowler. Validator is an environment designed to run a custom check on answers (usually) using javascript code.

Define a placeholder command for validator and feedback.

\begin{verbatim}
\newcommand{\PH@Command}{}
\def\feedback{
@ifnextchar[\@feedbackcode]{
@feedbackattempt}
\def\@feedbackcode[#1]{
\stepcounter{identification}\
\ifvmode\IgnorePar\fi \EndP\
\ifthenelse{\equal{#1}{attempt}}{<div class="feedback" data-feedback="attempt" id="feedback\arabic{identification}">}{
\ifthenelse{\equal{#1}{correct}}{<div class="feedback" data-feedback="correct" id="feedback\arabic{identification}">}{\HCode{<div class="feedback" data-feedback="script" id="feedback\arabic{identification}"><script type="text/javascript">function feedback\arabic{identification}()\{
return \detokenize{#1};\}\</script>}}}}
\def\endfeedback{\HCode{</div>\IgnoreIndent}
\end{verbatim}

Feedback environments take an optional parameter (which describes when the feedback is to be provided).

\begin{verbatim}
@htXimera\end{verbatim}
2.12.3 Ungraded activities

The *ungraded* environment is used to record that certain parts of activities should not be worth points. For example, if you want to use a multipleChoice as a survey question, you can place it inside an *ungraded* environment. On the \LaTeX{} side, the *ungraded* environment does nothing.

\begin{verbatim}
\newenvironment{ungraded}{}{}
\end{verbatim}

But on the html side, *ungraded* wraps the activities in a div in order to assign some weight to them for grading.

\begin{verbatim}
\renewenvironment{ungraded}{% \ifvmode \IgnorePar\fi \EndP\HCode{<div class="ungraded">}\IgnoreIndent% }{% \ifvmode \IgnorePar\fi \EndP\HCode{</div>}\IgnoreIndent% }
\end{verbatim}

2.13 Support for the web

2.13.1 MathJax support

When using mathjax, dump all the \newcommand{s to a .jax file. First, create the .jax file.

\begin{verbatim}
\ifdefined\HCode
\else
\newwrite\myfile
\immediate\openout\myfile=\jobname.jax
\fi
\end{verbatim}

From only.dtx we must also create prompt on the MathJax side.

\begin{verbatim}
\ifdefined\HCode
\else
\immediate\write\myfile{\unexpanded{\newenvironment}{\unexpanded{\prompt}}{}{}}
\fi
\end{verbatim}

Redefine newcommand appropriately.

\begin{verbatim}
\let\@OldDeclareMathOperator\DeclareMathOperator
\renewcommand\DeclareMathOperator[2]{\@OldDeclareMathOperator{#1}{#2}\immediate\write\myfile{\unexpanded{\DeclareMathOperator}{\unexpanded{#1}}{\unexpanded{#2}}}}
\end{verbatim}

Include the jax’ed newcommands

\begin{verbatim}
% Remove commands that use @
\immediate\write18{sed -i "/@/d" \jobname.jax}

% Replace #1 with #1 and so forth
\immediate\write18{sed -i "s/\string#\string\string\string\string\\string\string\string\string\string\string\((0-9)\string\string\)/string#string\string\string\((0-9)\string\string\)/string#string\string\string\\string/g"
\end{verbatim}
% Instead of a nonbreaking space, use a standard space
\makeatletter
\def\FV@Space{\space}
\makeatother
% Include the mathjax newcommands in a math/tex script right at the beginning of the body
\Configure{BODY}{%}
\HCode{<body>\Hnewline}%
\Tg<div class="preamble">%
\Tg<script type="math/tex">%
\BVerbatimInput{\jobname.jax}%
\Tg</script>%
\IfFileExists{\jobname.ids}{\HCode{<script type="text/javascript">\Hnewline}%
\BVerbatimInput{\jobname.ids}%
\HCode{</script>\Hnewline}%
\Tg</div>%
}%
\HCode{</body>\Hnewline}%
}

Now I just need to add a newcommand command which outputs the appropriate new-
commands to MathJax; then this should be "good enough" for our purposes.
\newtoks\eqtoks
\def\AltMath#1${\eqtoks{#1}%
\HCode{<script type="math/tex">\the\eqtoks</script>}$}
\Configure{$}{}{}{\expandafter\AltMath}
\def\AltlMathI#1\(){\eqtoks{#1}%
\HCode{<script type="math/tex">\the\eqtoks</script>\)}
\Configure{()}{\AltlMathI}{}
\def\AltlDisplay#1\]{\eqtoks{#1}%
\HCode{<script type="math/tex; mode=display">\the\eqtoks</script>}\}]
\Configure{\[\]}{\AltlDisplay}{}
\def\AltlDisplayI#1$${\eqtoks{#1}%
\HCode{<script type="math/tex; mode=display">\the\eqtoks</script>$$}
\Configure{$$}{}{}{\expandafter\AltlDisplayI}

Need to turn off htmlpar too, as expained in http://tex.stackexchange.com/questions/204930/vertical-
spaces-in-htlatex-scriptenv
\newcommand\VerbMath[1][]{%\newline
\renewenvironment{#1}[1]{\NoFonts}{\EndNoFonts}{%\newline
_SCRIPTEnv(#1){%\ifmode \IgnorePar\fi \EndP\HCode{<script type="math/tex; mode=display"> \string#1 \strut}\newline
}\newline
}

This is a fix for the LAODE book, which uses matlabEquation as if it were an equation
\VerbMath{equation}
\VerbMath{equation*}
\VerbMath{align}
\VerbMath{align*}
\VerbMath{alignat}
\VerbMath{alignat*}
\VerbMath{eqnarray}
\VerbMath{eqnarray*}

//cfgXimera}
2.13.2 Semantic HTML

\textbf Using \textbf emits a <strong> tag.
\emph Using \emph or similar emits an <em> tag.
\texttt Using \texttt emits a <code> tag.

2.14 Tools

2.14.1 Suppress

The suppress environment is a good way to suppress output without commenting it. This way we can avoid many of the places we use environ package and this should also avoid most of the verbatim conflicts. This is code adapted from syntonly.sty.

2.14.2 The End

It seems that some of the files need to conclude with something or another.
3 xourse.cls

The default behavior of the class is to provide a table of contents listing all activities in the course. This option will suppress this table of contents.

\newif\ifnotoc
\notocfalse
\DeclareOption{notoc}{\notoctrue}
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{ximera}}
\ProcessOptions\relax
\LoadClass{ximera}

% \begin{macrocode}
⟨∗classXourse⟩
\newcommand{\skip@preamble}{%
\let\document\relax\let\enddocument\relax%
\newenvironment{document}{\let\input\otherinput}{}%
\renewcommand{\documentclass}[2][subfiles]{}}
\end{macrocode}

Note that the new command \subfile calls for \skip@preamble within a group. The changes to \document and \documentclass are undone after the inclusion of the subfile.

Numbering starts a page too soon without this:
\let\otherinput\input
\let\othermaketitle\maketitle
\maketitle

In a xourse file, \maketitle is redefined to give course packet title page and toc.
\newcommand{\maketitle}{ %
\pagestyle{empty}
\begin{center}
\vskip .25\textheight
\hrulefill\vskip 1em
\bfseries{\Huge @title} \hrulefill\vskip 3em
{\Large @author}\vskip 2em
{\large @date}\end{center}
\clearpage

When notoc option is used, we do not include a table of contents. Otherwise we include a table of contents in every course packet.
\ifnotoc
\else
\tableofcontents\clearpage
\fi
\clearpage
\fi

Switch to main pagestyle, just like a document with documentclass ximera.

\pagestyle{main}

Renew maketitle to usual definition.

\let\maketitle\othermaketitle

And we finish with our redefinition of \maketitle.

\relax

\fi

\fi

\relax

\relax

3.1.1 Regular activities

\activity Documents included with \activity will be included in the body of the xourse document. Any \input commands within included ximera documents will be ignored. Any \usepackage commands within included ximera documents will cause an error. Overlapping \newcommand definitions within multiple ximera documents included simultaneously will cause an error. The \activity command inputs the file name provided without \documentclass, without \begin{document}/\end{document} and without any inputs in the preamble of the included file.

\ifnonewpage
\newcommand{\activity}[2][{}]{%
\setkeys{activity}{#1}
\renewcommand{\input}[1]{{}
\begingroup\skip@preamble\otherinput{#2}\endgroup\par\vspace{\topsep}
\let\input\otherinput}
\else
\newcommand{\activity}[2][{}]{%
\setkeys{activity}{#1}
\renewcommand{\input}[1]{{}
\begingroup\skip@preamble\otherinput{#2}\endgroup\clearpage
\let\input\otherinput}
\fi
\relax

3.1.2 Practice activities

\practice Like \activity but not expecting a title.

\ifhandout
\newcommand{\practice}[2][{}]{
\setkeys{practice}{#1}{}
\renewcommand{\input}[1]{{}
\begingroup\skip@preamble\otherinput{#2}\endgroup
\let\input\otherinput}
\else
\relax

When running xake, we can just ignore activities

\ifxake
\renewcommand{\activity}[2][]{}
\fi
\relax

\relax

34
\newcommand{practice}[2][]{\texttt{\detokenize{#2}}} % gives file name for practice
\setkeys{practice}{#1} % !!!!!!
\renewcommand{\input}[1]{} %
\begingroup\skip@preamble\otherinput{#2}\endgroup
\let\input\otherinput}
\fi
\relax
The practice environment does nothing, but will eventually produce exercises at the end of an activity
\ifxake
\renewcommand{practice}[2][]{}
\fi
I suppose it is reasonable for practice cards to NOT have an activitystyle, since the activitystyle is basically PRACTICE.
\renewcommand{\paragraph}[1]{%
\HCode{<span class="paragraphHead">}#1%
\HCode{</span>}\par\IgnorePar}
\renewcommand{\subparagraph}[1]{%
\HCode{<span class="subparagraphHead">}#1% 
\HCode{</span>}
\relax}
\newcounter{ximera@part}
\setcounter{ximera@part}{0}
\renewcommand{\part}[1]{%
\stepcounter{ximera@part}\
\ifvmode \IgnorePar\fi \EndP%
%HCode{<h1 id="part\arabic{ximera@part}" class="card part">}#1%HCode{</h1>}% makes cards disappear?
%HCode{<h1 id="part\arabic{ximera@part}" class="card part">}#1\HCode{</h1>}%
\IgnoreIndent%
}
\paragraph
\subsection The name of a subsection inside an activity.
\part Xourse files can have parts. The name of a large part of a xourse.
\paragraph
\subparagraph An even smaller heading.

3.2 Sectioning

Makes the table of contents look a bit better. This can be redefined in the preamble if you do not like the appearance. The name of a section inside an activity.
\section
\subsection The name of a subsection inside an activity.
\part Xourse files can have parts. The name of a large part of a xourse.
\paragraph
\subparagraph An even smaller heading.
3.3 Grading by points

The graded environment does nothing in LaTeX, but in HTML, it wraps the activities in a div in order to assign some weight to them for grading.

So indeed this environment in HTML wraps the activities in a div in order to assign some number of points to them.

3.4 Logos

The xourse logo is an og:image in the opengraph taxonomy.