ximera — Simultaneously write print and online interactive materials.[*]

Jim Fowler   Jeramiah Hocutt   Oscar Levin   Jason Nowell
Hans Parshall   Bart Snapp

Released ?

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

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.

```
⟨*classXimera⟩
```

`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 printiable "handout."

```
\newif\ifhandout
\handoutfalse
\DeclareOption{handout}{\handouttrue}
```

`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.

```
\newif\ifnoauthor
\noauthorfalse
\DeclareOption{noauthor}{\noauthortrue}
```

`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.

```
\newif\ifnooutcomes
\nooutcomesfalse
\DeclareOption{nooutcomes}{\nooutcomestrue}
```

`instructornotes` This option will turn on (and off) notes written for the instructor.

```
\newif\ifinstructornotes
\instructornotesfalse
\DeclareOption{instructornotes}{\instructornotestrue}
```

[*]This file describes version ?, last revised ?.

`noinstructornotes` This option will turn off (and on) notes written for the instructor.

```
\DeclareOption{noinstructornotes}{\instructornotestrue}
```

`hints` When the `handout` options is used, hints are not shown. This option will make hints visible in handout mode.

```
\newif\ifhints
\hintsfalse
\DeclareOption{hints}{\hintstrue}
```

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

```
\newif\ifnewpage
\newpagefalse
\DeclareOption{newpage}{\newpagetrue}
```

`numbers` This option will number the titles of the activity. By default the activities are unnumbered.

```
\newif\ifnumbers
\numbersfalse
\DeclareOption{numbers}{\numberstrue}
```

`wordchoicegiven` This option will replace the choices shown by `wordChoice` with the correct choice. No indication of the `wordChoice` environment will be shown.

```
\newif\ifwordchoicegiven
\wordchoicegivenfalse
\DeclareOption{wordchoicegiven}{\wordchoicegiventrue}
\newif\iffirstinlinechoice% Support for other wordchoice command contents.
\firstinlinechoicetrue

\newif\ifxake
\xakefalse
\DeclareOption{xake}{\xaketrue}

\newif\iftikzexport
\tikzexportfalse
\DeclareOption{tikzexport}{%
  \tikzexporttrue%
  \handoutfalse%
  \numbersfalse%
  \newpagefalse%
  \hintsfalse%
  \nooutcomesfalse%
}

\DeclareOption*{%
  \PassOptionsToClass{\CurrentOption}{article}%
}
\ProcessOptions\relax
\LoadClass{article}
⟨/classXimera⟩
⟨*classXimera⟩
```

2.2 Loading packages

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

```
\RequirePackage[makeroom]{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[inline]{enumitem}
```

```
\RequirePackage[pagestyles]{titlesec}
\RequirePackage{titletoc}
\RequirePackage{titling}
\RequirePackage{url}
\RequirePackage[table]{xcolor}
\RequirePackage{tikz}
\RequirePackage{pgfplots}
\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* p
\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
\RequirePackage{multido}% http://ctan.org/pkg/multido
\RequirePackage{listings} %% is this required???

\RequirePackage{xkeyval}

\RequirePackage{comment}
⟨/classXimera⟩
```

Various packages must be loaded early to avoid polluting the `.jax` file.

```
⟨*classXimera⟩
\RequirePackage{gettitlestring}
\RequirePackage{nameref}
\RequirePackage{epstopdf}
⟨/classXimera⟩
```

2.3 Page setup

We want non-indented spaced-out paragraphs.

```
⟨*classXimera⟩
\setlength{\parindent}{0pt}
\setlength{\parskip}{5pt}
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
\oddsidemargin 62pt
\evensidemargin 62pt
\textwidth 345pt
\headheight 14pt
⟨/classXimera⟩
```

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

```
⟨*cfgXimera⟩
\Preamble{xhtml}

% We don't want to translate font suggestions with ugly wrappers like
% <span class="cmti-10"> for italic text
\NoFonts

% Don't output xml version tag
\Configure{VERSION}{}

% Output HTML5 doctype instead of the default for HTML4
\Configure{DOCTYPE}{\HCode{<!doctype html>\Hnewline}}

```

```
% Custom page opening
\Configure{HTML}{\HCode{<html lang="en">\Hnewline}}{\HCode{\Hnewline</html>}}

% Reset <head>, aka delete all default boilerplate; alternatively set up new content
\Configure{@HEAD}{\HCode{<meta name="generator" content="TeX4ht (http://www.cse.ohio-state.e
\Configure{@HEAD}{\HCode{<meta name="ximera" content="version 0.0.1" />\Hnewline}}
\Configure{@HEAD}{\HCode{<link href="https://ximera.osu.edu/public/stylesheets/standalone.cs
\Configure{@HEAD}{\HCode{<script type="text/javascript" async src="https://ximera.osu.edu/pu
⟨/cfgXimera⟩
```

Disable certain ligatures in HTML.

```
⟨*htXimera⟩
\usepackage{microtype}
\DisableLigatures[f]{encoding=*}
⟨/htXimera⟩
```

I am not sure what this does.

```
⟨*htXimera⟩
\RenewEnviron{html}{\HCode{\BODY}}
⟨/htXimera⟩
```

2.4 Structure

2.4.1 Macros

Makes everymath display style even when inline, could be optional.

```
⟨*classXimera⟩
\everymath{\displaystyle}
⟨/classXimera⟩
```

Ok not everything, we also need to configure "display style" limits.

```
⟨*classXimera⟩
\let\prelim\lim
\renewcommand{\lim}{\displaystyle\prelim}
⟨/classXimera⟩
```

2.4.2 Theorem and theorem-like environments

On the web, a theorem is emitted as a special `<div>`.

```
⟨*htXimera⟩
\newcommand{\ConfigureTheoremEnv}[1]{%
\renewenvironment{#1}[1][]{\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="
}
⟨/htXimera⟩
```

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

`theorem` Theorem

```
⟨classXimera⟩      \newtheorem{theorem}{Theorem}
⟨htXimera⟩      \ConfigureTheoremEnv{theorem}
```

`algorithm` Algorithm

```
⟨classXimera⟩      \newtheorem{algorithm}{Algorithm}
⟨htXimera⟩      \ConfigureTheoremEnv{algorithm}
```

`axiom` Axiom

```
⟨classXimera⟩      \newtheorem{axiom}{Axiom}
⟨htXimera⟩      \ConfigureTheoremEnv{axiom}
```

`claim` Claim

```
⟨classXimera⟩      \newtheorem{claim}{Claim}
⟨htXimera⟩      \ConfigureTheoremEnv{claim}
```

`conclusion` Conclusion

```
⟨classXimera⟩  \newtheorem{conclusion}{Conclusion}
⟨htXimera⟩   \ConfigureTheoremEnv{conclusion}
```

`condition` Condition

```
⟨classXimera⟩  \newtheorem{condition}{Condition}
⟨htXimera⟩   \ConfigureTheoremEnv{condition}
```

`conjecture` Conjecture

```
⟨classXimera⟩  \newtheorem{conjecture}{Conjecture}
⟨htXimera⟩   \ConfigureTheoremEnv{conjecture}
```

`corollary` Corollary

```
⟨classXimera⟩  \newtheorem{corollary}{Corollary}
⟨htXimera⟩   \ConfigureTheoremEnv{corollary}
```

`criterion` Criterion

```
⟨classXimera⟩  \newtheorem{criterion}{Criterion}
⟨htXimera⟩   \ConfigureTheoremEnv{criterion}
```

`definition` Definition

```
⟨classXimera⟩  \newtheorem{definition}{Definition}
⟨htXimera⟩   \ConfigureTheoremEnv{definition}
```

`example` Example

```
⟨classXimera⟩  \newtheorem{example}{Example}
⟨htXimera⟩   \ConfigureTheoremEnv{example}
```

`explanation` Explanation

```
⟨classXimera⟩  \newtheorem*{explanation}{Explanation}
⟨htXimera⟩   \ConfigureTheoremEnv{explanation}
```

`fact` Fact

```
⟨classXimera⟩  \newtheorem{fact}{Fact}
⟨htXimera⟩   \ConfigureTheoremEnv{fact}
```

`lemma` Lemma

```
⟨classXimera⟩  \newtheorem{lemma}{Lemma}
⟨htXimera⟩   \ConfigureTheoremEnv{lemma}
```

`formula` Formula

```
⟨classXimera⟩  \newtheorem{formula}{Formula}
⟨htXimera⟩   \ConfigureTheoremEnv{formula}
```

`idea` Idea

```
⟨classXimera⟩  \newtheorem{idea}{Idea}
⟨htXimera⟩   \ConfigureTheoremEnv{idea}
```

`notation` Notation

```
⟨classXimera⟩  \newtheorem{notation}{Notation}
⟨htXimera⟩   \ConfigureTheoremEnv{notation}
```

`model` Model

```
⟨classXimera⟩  \newtheorem{model}{Model}
⟨htXimera⟩   \ConfigureTheoremEnv{model}
```

`observation` Observation

```
⟨classXimera⟩  \newtheorem{observation}{Observation}
⟨htXimera⟩   \ConfigureTheoremEnv{observation}
```

`proposition` Proposition

```
⟨classXimera⟩  \newtheorem{proposition}{Proposition}
⟨htXimera⟩   \ConfigureTheoremEnv{proposition}
```

`paradox` Paradox

```
⟨classXimera⟩  \newtheorem{paradox}{Paradox}
⟨htXimera⟩   \ConfigureTheoremEnv{paradox}
```

`procedure` Procedure

```
⟨classXimera⟩  \newtheorem{procedure}{Procedure}
⟨htXimera⟩   \ConfigureTheoremEnv{procedure}
```

remark Remark

```
⟨classXimera⟩     \newtheorem{remark}{Remark}
⟨htXimera⟩      \ConfigureTheoremEnv{remark}
```

summary Summary

```
⟨classXimera⟩     \newtheorem{summary}{Summary}
⟨htXimera⟩      \ConfigureTheoremEnv{summary}
```

template Template

```
⟨classXimera⟩     \newtheorem{template}{Template}
⟨htXimera⟩      \ConfigureTheoremEnv{template}
```

warning Warning

```
⟨classXimera⟩     \newtheorem{warning}{Warning}
⟨htXimera⟩      \ConfigureTheoremEnv{warning}
```

2.4.3 Enumerate fixes

Make enumerate use a letter

```
⟨*classXimera⟩
\renewcommand{\theenumi}{\textup{(\alph{enumi})}}
\renewcommand{\labelenumi}{\theenumi}
\renewcommand{\theenumii}{\textup{(\roman{enumii})}}
\renewcommand{\labelenumii}{\theenumii}
⟨/classXimera⟩
```

2.4.4 Proofs

proof A mathematical proof environment.

```
⟨*classXimera⟩
\renewcommand{\qedsymbol}{$\blacksquare$}
\renewenvironment{proof}[1][\proofname]
  {\begin{trivlist}\item[\hskip \labelsep \itshape \bfseries #1{}\hspace{2ex}]}
{\qed\end{trivlist}}
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
```

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'.

```
\providecommand{\latexProblemContent}[1]{#1}
% Iterate count for problem counts.
\Make@Counter{Iteration@probCnt}

\newcommand{\hang}{% top theorem decoration
  \begingroup%
  \setlength{\unitlength}{.005\linewidth}% \linewidth/200
    \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
  \nobreak
  \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%
}%
```

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.

```
\MakeCounter{problem}
\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|.

\forloop{Iteration@probCnt}{2}{\arabic{Iteration@probCnt} < \numexpr \value{problem@Depth} +
    .\expandafter\arabic{depth\Roman{Iteration@probCnt}Count}% Get the problem number of the
}
%\@ifpackageloaded{shuffle}{<true>}{<false>}% Check if Shuffle has been added. If so, add spe
%\ifhandout % Currently handout mode doesn't allow hints. Putting this code in place in case
% \theproblem
%\else
% \theproblem
%\fi
}


%%%%%% Configure various problem environment commands
\Make@Counter{problem@Depth}



%%%% Configure environments start content

\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 va
% 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 \problemNumber% Determine the correct number of the problem, and the format of that nu
]%
\slshape
}



%%%%% Configure environments end 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.
\stepcounter{problem@Depth}
\ifcsname c@depth\Roman{problem@Depth}Count\endcsname
\expandafter\ifnum\expandafter\value{depth\Roman{problem@Depth}Count}>0
\expandafter\setcounter{depth\Roman{problem@Depth}Count}{0}
\fi
\fi

\addtocounter{problem@Depth}{-2}% Exited a problem so we've exited a problem layer. Need -2 b

\par\addvspace{.5ex}\nobreak\noindent\hung %% line at the bottom

\ifhandout
\ifnewpage
\newpage
\fi
\fi
\end{trivlist}
}



%%%% Now populate the old environment names
%
% Old environments were "problem", "exercise", "exploration", and "question".
% Note that you can add content to the start/end code on top of these base code pieces if you


\newenvironment{problem}[1][2in]%
{%Env start code
\problemEnvironmentStart{#1}{Problem}
}
{%Env end code
\problemEnvironmentEnd
}

\newenvironment{exercise}[1][2in]%
{%Env start code
\problemEnvironmentStart{#1}{Exercise}
}
{%Env end code
\problemEnvironmentEnd
}

\newenvironment{exploration}[1][2in]%
{%Env start code
\problemEnvironmentStart{#1}{Exploration}
}
```

```
{%Env end code
\problemEnvironmentEnd
}

\newenvironment{question}[1][2in]%
{%Env start code
\problemEnvironmentStart{#1}{Question}
}
{%Env end code
\problemEnvironmentEnd
}
⟨/classXimera⟩
```

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

```
⟨*htXimera⟩
\newcounter{identification}
\setcounter{identification}{0}

\newcommand{\ConfigureQuestionEnv}[2]{%
% refstepcounter ensures that labels get updated within these environments
\renewenvironment{#1}{\refstepcounter{problem}}{}%
\ConfigureEnv{#1}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div role="a
}

\ConfigureQuestionEnv{problem}{problem}
\ConfigureQuestionEnv{exercise}{exercise}
\ConfigureQuestionEnv{question}{question}
\ConfigureQuestionEnv{exploration}{exploration}
\ConfigureQuestionEnv{xarmaBoost}{xarma-boost}
\ConfigureQuestionEnv{hint}{hint}
\ConfigureQuestionEnv{shuffle}{shuffle}
⟨/htXimera⟩
```

2.4.6 Hints

`hint` Hint environments can be embedded inside problems.

```
⟨*classXimera⟩
```

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.

```
\renewenvironment{hint}
{
\ifhandout
\setbox0\vbox\bgroup
\else
\begin{trivlist}\item[\hskip \labelsep\small\slshape\bfseries Hint:\hspace{2ex}]
\small\slshape
\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}]
\small\slshape}
{\end{trivlist}}
\fi

⟨/classXimera⟩
```

2.4.7 Solution

`solution` The solution to a problem.

```
⟨*classXimera⟩
%% solution environment
\ifhandout % what follows is handout behavior
\newenvironment{solution}%
      {%
 \setbox0\vbox\bgroup
      }
              {%
 \egroup
      }
\else
\newenvironment{solution}%
      {%
 \begin{trivlist}
 \item[\hskip \labelsep\bfseries Solution:\hspace{2ex}]
      }
      % %% line at the bottom}
      {
\end{trivlist}
 \par\addvspace{.5ex}\nobreak\noindent\hung
      }
\fi



⟨/classXimera⟩
```

2.4.8 Code listing environments

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

```
⟨*classXimera⟩
\DefineVerbatimEnvironment{code}{Verbatim}{numbers=left,frame=lines,label=Code,labelposition=
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
\DefineVerbatimEnvironment{python}{Verbatim}{numbers=left,frame=lines,label=Python,labelposit
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
```

```
\DefineVerbatimEnvironment{javascriptCode}{Verbatim}{numbers=left,frame=lines,label=JavaScrip
⟨/classXimera⟩
⟨*cfgXimera⟩
\renewenvironment{javascriptCode}{\NoFonts}{\EndNoFonts}
\ScriptEnv{javascriptCode}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<d
⟨/cfgXimera⟩
```

On the web, translate verbatim and lstlisting blocks into `<pre>` elements.

```
⟨*cfgXimera⟩
\ConfigureEnv{verbatim}{\HCode{<pre>}}{\HCode{</pre>}}{}{}
\ConfigureEnv{lstlisting}{\HCode{<pre>}}{\HCode{</pre>}}{}{}
⟨/cfgXimera⟩
```

2.4.9 Dialogues

`dialogue` A dialogue between people.

```
⟨*classXimera⟩
\newenvironment{dialogue}{%
   \renewcommand\descriptionlabel[1]{\hspace{\labelsep}\textbf{##1:}}
   \begin{description}%
}{%
   \end{description}%
}
⟨/classXimera⟩
```

On the web, the resulting `<dl>` should have an appropriate `class` set.

```
⟨*htXimera⟩
\renewenvironment{dialogue}{\begin{description}}{\end{description}}

\ConfigureList{dialogue}%
   {\EndP\HCode{<dl \a:LRdir class="dialogue">}%
      \PushMacro\end:itm
\global\let\end:itm=\empty}
   {\PopMacro\end:itm \global\let\end:itm \end:itm
\EndP\HCode{</dd></dl>}\ShowPar}
   {\end:itm \global\def\end:itm{\EndP\Tg</dd>}\HCode{<dt
        class="actor">}\bgroup \bf}
   {\egroup\EndP\HCode{</dt><dd\Hnewline class="speech">}}
⟨/htXimera⟩
```

2.4.10 Instructor notes

```
⟨*classXimera⟩

%% instructor intro/instructor notes
%%
\ifhandout % what follows is handout behavior
\ifinstructornotes
\newenvironment{instructorIntro}%
       {%
 \begin{trivlist}
 \item[\hskip \labelsep\bfseries Instructor Introduction:\hspace{2ex}]
}
       % %% line at the bottom}
       {
 \end{trivlist}
 \par\addvspace{.5ex}\nobreak\noindent\hung
       }
\else
\newenvironment{instructorIntro}%
       {%
 \setbox0\vbox\bgroup
       }
       {%If this mysteriously starts breaking
```

```
                       % remove \ignorespacesafterend
\egroup\ignorespacesafterend
      }
              \fi
\else% for handout, so what follows is default
\ifinstructornotes
\newenvironment{instructorIntro}%
      {%
        \setbox0\vbox\bgroup
      }
{%
  \egroup
}
               \else
       \newenvironment{instructorIntro}%
{%
  \begin{trivlist}
  \item[\hskip \labelsep\bfseries Instructor Introduction:\hspace{2ex}]
}
% %% line at the bottom}
{
  \end{trivlist}
  \par\addvspace{.5ex}\nobreak\noindent\hung
}
               \fi
\fi




%% instructorNotes environment
\ifhandout % what follows is handout behavior
\ifinstructornotes
\newenvironment{instructorNotes}%
      {%
 \begin{trivlist}
 \item[\hskip \labelsep\bfseries Instructor Notes:\hspace{2ex}]
      }
      % %% line at the bottom}
      {
\end{trivlist}
 \par\addvspace{.5ex}\nobreak\noindent\hung
      }
      \else
\newenvironment{instructorNotes}%
      {%
        \setbox0\vbox\bgroup
      }
{%
  \egroup
}
               \fi
\else% for handout, so what follows is default
\ifinstructornotes
\newenvironment{instructorNotes}%
      {%
 \setbox0\vbox\bgroup
      }
      {%
 \egroup
      }
      \else
      \newenvironment{instructorNotes}%
```

```
              {%
        \begin{trivlist}
        \item[\hskip \labelsep\bfseries Instructor Notes:\hspace{2ex}]
              }
              % %% line at the bottom}
              {
        \end{trivlist}
        \par\addvspace{.5ex}\nobreak\noindent\hung
              }
                      \fi
                              \fi

⟨/classXimera⟩
```

2.4.11 Only

prompt The prompt part for mathmode

```
⟨*classXimera⟩
\ifhandout
\NewEnviron{prompt}{}
% Currently breaks when put in mathmode!
% \newenvironment{prompt}{\suppress}{\endsuppress}
\else
\newenvironment{prompt}
     {\bgroup\color{gray!50!black}}
        {\egroup}
\fi
```

onlineOnly Only display it online

```
\ifhandout
\NewEnviron{onlineOnly}{
\iftikzexport
\BODY
\else
\fi
}
\else
\newenvironment{onlineOnly}
     {\bgroup\color{red!50!black}}
{\egroup}
\fi

\newcommand{\pdfOnly}[1]{\iftikzexport\else #1\fi}
⟨/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
```

foldable 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,
%  leftmargin=0,
%  innertopmargin=0pt,
%  innerbottommargin=0pt,
%  skipbelow=\baselineskip,
%  linecolor=textColor!20!white,
%  fontcolor=textColor,
%  backgroundcolor=background
%]{foldable}%
%\fi

%% pretty-expandable
%\iftikzexport
\newenvironment{expandable}{%
}{%
}
%\else
%\newmdenv[
%  font=\upshape,
%  outerlinewidth=3,
%  topline=false,
%  bottomline=false,
%  leftline=true,
%  rightline=false,
%  leftmargin=0,
%  innertopmargin=0pt,
%  innerbottommargin=0pt,
%  skipbelow=\baselineskip,
%  linecolor=black,
%]{expandable}%
%\fi

\newcommand{\unfoldable}[1]{#1}

⟨/classXimera⟩
```

On the web, these foldable elements could be HTML5 details and summary.

```
⟨*htXimera⟩
\renewenvironment{foldable}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<

\renewenvironment{expandable}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode

}{\HCode{</div>}\IgnoreIndent}

\renewcommand{\unfoldable}[1]{\HCode{<span class="unfoldable">}#1\HCode{</span>}}
⟨/htXimera⟩
```

2.4.13 Leashes

`leash` Put content inside a scrollable box.

```
⟨*classXimera⟩

\newenvironment{leash}[1]{%
}{%
}


```

```
⟨/classXimera⟩
⟨*htXimera⟩
\renewenvironment{leash}[1]{\ifvmode \IgnorePar\fi \EndP\HCode{<div style="overflow: auto; h
⟨/htXimera⟩
```

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.

```
⟨*classXimera⟩
\newcommand{\license}{\excludecomment}
⟨/classXimera⟩
```

`\acknowledgement` In the preamble, use `\acknowledgement` to credit others who contributed to the intellectual content beside the author.

```
⟨*classXimera⟩
\newcommand{\acknowledgement}{\excludecomment}
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
\renewcommand{\tag}{\excludecomment}
⟨/classXimera⟩
```

On the HTML side, we mark the file as the appropriate kind of object—either activity or xourse.

```
⟨*htXourse⟩
% Mark this as a xourse file
\Configure{@HEAD}{\HCode{<meta name="description" content="xourse" />\Hnewline}}
⟨/htXourse⟩
```

2.5.2 Abstract

`abstract` Every activity should include a short abstract.

```
⟨*classXimera⟩
\let\abstract\relax
\let\endabstract\relax
% Use of environ package, may want to find a better way.
\NewEnviron{abstract}{\protected@xdef\theabstract{\BODY}}
⟨/classXimera⟩
```

The abstract has been stored in `\theabstract` and should be emitted as a div, but confusingly I guess `<div class="abstract">` is defined somewhere deeper inside tex4ht, so the code below is probably unnecessary.

```
⟨*cfgXimera⟩
\let\abstract\relax
\let\endabstract\relax
⟨/cfgXimera⟩
```

2.5.3 Titles and authors

2.5.4 Authors

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

```
⟨*classXimera⟩
\let\@emptyauthor\@author
\def\author#1{\gdef\@author{#1}}
\def\@author{\@latex@warning@no@line{No \noexpand\author given}}
⟨/classXimera⟩
```

Include author name in meta tags

```
⟨*htXimera⟩
\Configure{@HEAD}{\HCode{<meta name="author" content="}\@author\HCode{" />\Hnewline}}
⟨/htXimera⟩
```

The `\and` command would emit tabular environments which really should not appear in a meta tag.

```
⟨htXimera | classXimera⟩\def\and{and }
```

2.5.5 Title

`\title` Activities have titles.

```
⟨*classXimera⟩
\let\title\relax
\newcommand{\title}[1][]{{\protected@xdef\@pretitle{#1}}\protected@xdef\@title}

\title{}

\newcounter{titlenumber}
\renewcommand{\thetitlenumber}{\arabic{titlenumber}}
%\renewcommand{\thesection}{\arabic{titlenumber}} %% Makes section numbers work
\setcounter{titlenumber}{0}

\newpagestyle{main}{
\sethead[\textsl{\ifnumbers\thetitlenumber\quad\fi\@title}][][] % even
{}{}{\textsl{\ifnumbers\thetitlenumber\quad\fi\@title}} % odd
\setfoot[\thepage][][] % even
{}{}{\thepage} % odd
}
\pagestyle{main}
```

`\maketitle` 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\fi}{\ifnumbers\hspace{1em}\else\hspac
  \phantomsection%
  \ifnumbers\addcontentsline{toc}{section}{\thetitlenumber~\@title}\else\addcontentsline{toc
  \vskip .6em\noindent\textit\theabstract\setcounter{problem}{0}\setcounter{section}{0}\setc
  \ifnooutcomes\else\let\thefootnote\relax\footnote{Learning outcomes: \theoutcomes}\fi
  \ifnoauthor\else\let\thefootnote\relax\footnote{Author(s):~\@author}\fi
  \aftergroup\@afterindentfalse
  \aftergroup\@afterheading}

\ifnumbers
\setcounter{secnumdepth}{2}
\renewcommand{\thesection}{\arabic{titlenumber}.\arabic{section}~}
\else
\setcounter{secnumdepth}{-2}
\fi

\def\activitystyle{}
\newcounter{sectiontitlenumber}
\setcounter{secnumdepth}{0}
\newcommand\chapterstyle{%
  \def\activitystyle{activity-chapter}
  \def\maketitle{%
    \addtocounter{titlenumber}{1}%
                    {\flushleft\small\sffamily\bfseries\@pretitle\par\vspace{-1.5em}}%
                    {\flushleft\LARGE\sffamily\bfseries\thetitlenumber\quad\@title \par }%
                    {\vskip .6em\noindent\textit\theabstract\setcounter{problem}{0}\setcounte
                    \par\vspace{2em}
                    \phantomsection\addcontentsline{toc}{section}{\textbf{\thetitlenumber\qua
```

```
                  }}

\newcommand\sectionstyle{%
  \def\activitystyle{activity-section}
  \def\maketitle{%
    \addtocounter{sectiontitlenumber}{1}
    {\flushleft\small\sffamily\bfseries\@pretitle\par\vspace{-1.5em}}%
    {\flushleft\Large\sffamily\bfseries\thetitlenumber.\thesectiontitlenumber\quad\@title \pa
    {\vskip .6em\noindent\textit\theabstract}%
    \par\vspace{2em}
    \phantomsection\addcontentsline{toc}{subsection}{\thetitlenumber.\thesectiontitlenumber\c
  }}


\iftikzexport%% allows xake to handle \chapterstyle and \sectionstye
\renewcommand\chapterstyle{\def\activitystyle{chapter}}
\renewcommand\sectionstyle{\def\activitystyle{section}}
\else
\fi

⟨/classXimera⟩
```

Eliminate some formatting that we'll handle later with CSS

```
⟨*htXimera⟩
\renewcommand{\maketitle}{}
⟨/htXimera⟩
```

2.5.6 Learning Outcomes

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

```
⟨*classXimera⟩
\def\theoutcomes{}

\ifdefined\HCode%
  \newcommand{\outcome}[1]{}
\else%
  \newwrite\outcomefile
  \immediate\openout\outcomefile=\jobname.oc

  \newcommand{\outcome}[1]{\edef\theoutcomes{\theoutcomes #1~}%
  \immediate\write\outcomefile{\unexpanded{\outcome}{#1}}}
  \fi%
⟨/classXimera⟩
```

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.

```
⟨*cfgXimera⟩
\renewcommand{\outcome}[1]{
  \Configure{@HEAD}{\HCode{<meta name="learning-outcome" content="#1"/>\Hnewline}}
}
% Sometimes there are no outcomes at all
\IfFileExists{\jobname.oc}{\input{\jobname.oc}}{}

\renewcommand{\outcome}[1]{%
  \HCode{<span class="learning-outcome">#1</span>}
}
⟨/cfgXimera⟩
```

2.5.7 Labels and references

`\label` Labels and refs both generate anchors. A `\label` can be referenced from any file in the xourse.

```
⟨*htXimera⟩
\renewcommand{\label}[1]{\HCode{<a class="ximera-label" id="#1"></a>}}
⟨/htXimera⟩
```

`\ref` A `\ref` can connect one TEX file to another if they are in the same xourse.

```
⟨*htXimera⟩
\renewcommand{\ref}[1]{\HCode{<a class="reference" href="\##1">#1</a>}}
⟨/htXimera⟩
```

2.6 Images

2.6.1 Images

`image` Place images inside an `image` environment. On paper, this centers the image. On the web, this provides additional benefits.

```
⟨*classXimera⟩
%\newenvironment{image}[1][]{\begin{center}}{\end{center}}
\NewEnviron{image}[1][3in]{%
  \begin{center}\resizebox{#1}{!}{\BODY}\end{center}% resize and center
}
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
\newcommand{\alt}[1]{}
⟨/classXimera⟩
```

The `image` environment doesn't actually work in tex4ht as defined with NewEnviron; so this renewenvironment is needed. image-environment also gets formatted in a well, and when the user clicks on the image, it zooms in.

```
⟨*htXimera⟩
\newcounter{imagealt}
\setcounter{imagealt}{0}
\renewenvironment{image}[1][]{\stepcounter{imagealt}%
  \ifvmode \IgnorePar\fi \EndP%
  \HCode{<div class="image-environment" role="img" aria-labelledby="image-alt-\arabic{imageal
}{\HCode{</div>}}
\renewcommand{\alt}[1]{\HCode{<div style="display: none;" id="image-alt-\arabic{imagealt}">}
⟨/htXimera⟩
```

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 fo producing alt text.

```
⟨*cfgXimera⟩
\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}%
}
⟨/cfgXimera⟩
```

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

```
⟨*cfgXimera⟩
\ifcsname ifstandalone\endcsname
  \ifstandalone
    \renewcommand\includegraphics[2][]{}
  \fi
  \fi
⟨/cfgXimera⟩
```

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

```
⟨*htXimera⟩
```

```
\newcommand{\pgfsyspdfmark}[3]{}
⟨/htXimera⟩
```

2.6.2 TikZ export

We generate SVGs and PNGs for any TikZ images, via the “externalize” feature of TikZ.

We put hyperref after all other packages becuase that is better.

```
⟨*classXimera⟩
% Don’t use hyperref when using Tex4ht
\ifdefined\HCode
\RequirePackage{hyperref}
\else
\RequirePackage[pdfpagelabels,colorlinks=true,allcolors=blue!30!black]{hyperref}
\pdfstringdefDisableCommands{\def\hskip{}}%% quiets warning
\fi
⟨/classXimera⟩
```

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`.

```
⟨*classXimera⟩
\ifdefined\HCode
  \tikzexporttrue
\fi

\iftikzexport
  \usetikzlibrary{external}

  \ifdefined\HCode
    % in htlatex, just include the svg files
    \def\pgfsys@imagesuffixlist{.svg}

    \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 -o \image.svg \image.pdf ;
        mutool draw -r 150 -c rgbalpha -o \image.png \image.pdf ;
      }
    }
    \tikzexternalize[optimize=false,prefix=./]
  \fi

  \fi

⟨/classXimera⟩
```

2.6.3 XKCD

`\xkcd` Reference an XKCD cartoon.

```
⟨*classXimera⟩
\newcommand{\xkcd}[1]{#1}
⟨/classXimera⟩
```

On the web, this should be an image linked to the actual XKCD website.

```
⟨*htXimera⟩
\renewcommand{\xkcd}[1]{\ifvmode \IgnorePar\fi \EndP\HCode{<img src="https://imgs.xkcd.com/c
⟨/htXimera⟩
```

2.7 Interactives

2.7.1 Including widgets

`\includeinteractive` Cognate to `includegraphics` but instead of a graphics file, accepts a `.js` file which will be loaded as an interactive widget.

```
⟨*classXimera⟩
\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
}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewcommand{\includeinteractive}[2][]{\stepcounter{identification}\ifvmode \IgnorePar\fi \E
⟨/htXimera⟩
```

2.7.2 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

```
⟨*classXimera⟩
% Google Spreadsheet link (read only)
\newcommand{\googleSheet}[5]{%
  Google Spreadsheet link: \url{https://docs.google.com/spreadsheets/d/#1}%
}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewcommand{\googleSheet}[5]{%
  \ifthenelse{\equal{#4}{}}%
    {\HCode{<iframe width="#2px" height="#3px" src="https://docs.google.com/spreadsheets/d/#1
    {\ifthenelse{\equal{#5}{}}%
       {\HCode{<iframe width="#2px" height="#3px" src="https://docs.google.com/spreadsheets/d
       {\HCode{<iframe width="#2px" height="#3px" src="https://docs.google.com/spreadsheets/d
    }%
  }%
⟨/htXimera⟩
```

2.7.3 Geogebra

`\geogebra` Geogebra command. Requires id, width, and height as arguments.

```
⟨*classXimera⟩
%Geogebra link
\newcommand{\geogebra}[3]{Geogebra link: \url{https://tube.geogebra.org/m/#1}}
⟨/classXimera⟩
```

Define keys for answer geogebra key=value pairs.

```
⟨*htXimera⟩
\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}[true]{\def\geo@ld{#1}}
\define@key{geogebra}{sri}[true]{\def\geo@sri{#1}}
%set default key values
\setkeys{geogebra}{rc=false,sdz=false,smb=false,stb=false,stbh=false,ld=false,sri=false}
%command definition
\renewcommand{\geogebra}[4][]{%
  \setkeys{geogebra}{#1}% Set new keys
  \HCode{<iframe scrolling="no" src="https://tube.geogebra.org/material/iframe/id/#2/width/#3
```

```
⟨/htXimera⟩
```

2.7.4 Desmos

`\desmos` Desmos command. Requires id, width, and height as arguments.

```
⟨*classXimera⟩
\newcommand{\desmos}[3]{Desmos link: \url{https://www.desmos.com/calculator/#1}}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewcommand{\desmos}[3]{\HCode{<iframe src="https://www.desmos.com/calculator/#1" width="1C
⟨/htXimera⟩
```

2.7.5 Graphs

`\graph` An embedded graph (in math mode).

```
⟨*classXimera⟩
\newcommand{\graph}[2][]{\text{Graph of $#2$}}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewcommand{\graph}[2][]{\HCode{<div class="graph" data-options="#1">}#2\HCode{</div>}}
⟨/htXimera⟩
```

2.7.6 Video

`\youtube` Youtube command. Requires id.

```
⟨*classXimera⟩
\newcommand{\youtube}[1]{YouTube link: \url{https://www.youtube.com/watch?v=#1}}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewcommand{\youtube}[1]{\ifvmode \IgnorePar\fi \EndP\HCode{<div class="video youtube-playe
⟨/htXimera⟩
```

Video commands are also emitted, slightly differently, when placed at top-level in a xourse file.

```
⟨*htXourse⟩
\renewcommand\youtube[1]{%
\ifvmode \IgnorePar\fi \EndP\HCode{<a class="youtube" href="https://www.youtube.com/watch?v=#
}
⟨/htXourse⟩
```

2.7.7 JavaScript

`javascript` Code inside a javascript environment is printed on paper, but executed on the web.

```
⟨*classXimera⟩
\DefineVerbatimEnvironment{javascript}{Verbatim}{numbers=left,frame=lines,label=JavaScript,la
⟨/classXimera⟩
```

```
⟨*htXimera⟩
% for programming javascript
\renewenvironment{javascript}{\NoFonts}{\EndNoFonts}
\ScriptEnv{javascript}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div cl
⟨/htXimera⟩
```

`\js` Code inside a `\js` macro is evaluated and replaced with its value.

```
⟨*classXimera⟩
\def\js#1{\mbox{\texttt{\detokenize{#1}}}}
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\def\js#1{\stepcounter{identification}\HCode{<span class="inline-javascript" id="javascript\a
⟨/htXimera⟩
```

2.8 SageMath support

Load SageTEX if it exists.

```
⟨*classXimera⟩
\IfFileExists{sagetex.sty}{\RequirePackage{sagetex}}{}
⟨/classXimera⟩
```

sageCell Create an interactive SageMath widget.

```
⟨*classXimera⟩
\DefineVerbatimEnvironment{sageCell}{Verbatim}{numbers=left,frame=lines,label=SAGE,labelposit
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewenvironment{sageCell}{\NoFonts}{\EndNoFonts}
\ScriptEnv{sageCell}{\ifvmode \IgnorePar\fi \EndP\HCode{<div class="sage"><script type="text/
⟨/htXimera⟩
```

sageOutput Execute SageMath code and output the result.

```
⟨*classXimera⟩
\DefineVerbatimEnvironment{sageOutput}{Verbatim}{numbers=left,frame=lines,label=SAGE-Output,l
⟨/classXimera⟩
```

```
⟨*htXimera⟩
\renewenvironment{sageOutput}{\NoFonts}{\EndNoFonts}
\ScriptEnv{sageOutput}{\ifvmode \IgnorePar\fi \EndP\HCode{<div class="sageOutput"><script typ
⟨/htXimera⟩
```

sageSilent Execute SageMath code without outputing the result.

```
⟨*htXimera⟩
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\renewenvironment{sagesilent}{\NoFonts}{\EndNoFonts}
\ScriptEnv{sagesilent}{\ifvmode \IgnorePar\fi \EndP\HCode{<script type="text/sagemath">}\Html
⟨/htXimera⟩
```

2.9 Answerables

2.9.1 Answers

\answer A math answer

```
⟨*classXimera⟩

\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
```

Determines if answer is shown in handout mode. when `given=true`, show answer in handout mode, show answer in "given box" outside handout mode. When `given=false`, do not show answer in handout mode, show answer outside handout mode

```
\define@key{answer}{given}[true]{\def\ans@given{#1}}
```

Used for setting numeric answer tolerance for online student input.

```
\define@key{answer}{tolerance}{\def\ans@tol{#1}}
```

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

```
\define@key{answer}{validator}{}
```

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

```
\define@key{answer}{id}{\def\ans@id{#1}}
```

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

```
\define@key{answer}{format}{}
```

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

```
\setkeys{answer}{id=,given=false}
```

Basic code for `\answer`.

```
\newcommand{\answer}[2][]{%
\ifmmode%
\setkeys{answer}{#1}%
\recordvariable{\ans@id}
\ifthenelse{\boolean{\ans@given}}
{% Start then statement
\ifhandout
#2
\else
\underset{\scriptstyle\mathrm{given}}{\fbox{\ensuremath{#2}}}
\fi
}% End then statement
{% Start else statement
\ifhandout
\fbox{\rm{?}}
\else% show answer in box outside handout mode
\fbox{\ensuremath{#2}}
\fi
}% End else statement
\else%
\GenericError{\space\space\space\space}% Throw an error based on... something? -- Jason
{Attempt to use \@backslashchar answer outside of math mode}
{See https://github.com/ximeraProject/ximeraLatex for explanation.}
{Need to use either inline or display math.}%
\fi
}
⟨/classXimera⟩
```

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

```
⟨*htXimera⟩
\renewcommand{\answer}[2][false]{\HCode{<span class="answer respondable">}#2\HCode{</span>}}

\def\validator[#1]{\stepcounter{identification}\HCode{<div class="validator" id="validator\a
\def\endvalidator{\HCode{</div>}}

⟨/htXimera⟩
```

2.9.2 Multiple choice and the like

`multipleChoice` Multiple choice

```
⟨*classXimera⟩
% Jim: Originally this was \renewcommand{\theenumi}{$(\mathrm{\alph{enumi}})$}
% but that breaks tex4ht because mathmode can only be processed by mathjax.
% so now I made this just italicized.
```

2.9.3 Options

```
\define@key{choice}{value}[]{\def\choice@value{#1}}
```

This flags the answer as the correct answer

```
\define@boolkey{choice}{correct}[true]{\def\choice@correct{#1}}
```

Use an ID to refer to the choice.

```
\define@key{multipleChoice}{id}{\def\mc@id{#1}}
```

`\otherchoice` outputs the item if correct and nothing if incorrect.

```
\define@key{otherchoice}{value}[]{\def\otherchoice@value{#1}}
\define@boolkey{otherchoice}{correct}[true]{\def\otherchoice@correct{#1}}
```

Default key choices for multiple choice options. Default for choice pairs. Default: answers without the option ”correct=true” is ”incorrect”.

```
\setkeys{choice}{correct=false,value=}
```

Defaults for multipleChoice pairs. Default to no id? – Jason

```
\setkeys{multipleChoice}{id=}
```

Defaults for otherchoice pairs. Default "otherchoice" to behave like "choice" for error checking.

```
\setkeys{otherchoice}{correct=false,value=}
⟨/classXimera⟩
```

2.9.4 Choices

`\choice` Like `\item` but for choice environments. choice command denotes a possible answer choice for the multiple choice question.

```
⟨*classXimera⟩
\newcommand{\choice}[2][]{%
\setkeys{choice}{#1}%
\item{#2}
\ifthenelse{\boolean{\choice@correct}}
    {% Begin then result
    \ifhandout% if it's a handout do nothing.
    \else% otherwise place a checkmark when you select the "correct choice"... maybe? -- Jas
        \,\checkmark\,\setkeys{choice}{correct=false}
    \fi
    }% End then result
    {}% Begin/End else result.
}

%Define an expandable version of choice Not really meant to be used outside this package (us
% Is there a reason we can't just always use this as default? -- Jason
\newcommand{\choiceEXP}[2][]{%
\expandafter\setkeys\expandafter{choice}{#1}%
\item{#2}
\ifthenelse{\boolean{\choice@correct}}
{% Begin then result
\ifhandout
\else
\,\checkmark\,\setkeys{choice}{correct=false}
\fi
}% End then result
{}% Begin/End else result.
} %% note all the {} are needed in case the choice has [] in it.

% \otherchoice is the \choice used in wordChoice command.
\newcommand{\otherchoice}[2][]{%
\ignorespaces%
\setkeys{otherchoice}{#1}%
\ifthenelse{\boolean{\otherchoice@correct}}%
{% Start then result
#2\ignorespaces\setkeys{otherchoice}{correct=false}\ignorespaces%
}% End then result
{}% Start/End else result
\ignorespaces%
}%
\newcommand{\inlinechoice}[2][]{%
\setkeys{choice}{#1}%
\iffirstinlinechoice
(\hspace{-.25em}
\firstinlinechoicefalse
\else
/
\fi
#2
\ifthenelse{\boolean{\choice@correct}}%
{% Start then result
\ifhandout\else\checkmark\ignorespaces\setkeys{choice}{correct=false}\ignorespaces\fi%
}% End then result
{}% Start/End else result
\hspace{-.25em}\ignorespaces%
}

⟨/classXimera⟩
```

On the HTML side, `\choice` emits `<span>`s.

```
⟨*htXimera⟩
\newcounter{choiceId}
\renewcommand{\choice}[2][]{%
\setkeys{choice}{correct=false}%
\setkeys{choice}{#1}%
\stepcounter{choiceId}\IgnorePar%
\HCode{<span class="choice }%
\ifthenelse{\boolean{\choice@correct}}{\HCode{correct}}{}
\HCode{" }
\ifthenelse{\equal{\choice@value}{}}{}{\HCode{data-value="\choice@value" }}
\HCode{id="choice\arabic{choiceId}">}%
#2\HCode{</span>}}
⟨/htXimera⟩
```

2.9.5 Environment(s)

`multipleChoice` multipleChoice@ and multipleChoice@@ are for internal use only. Wrap `\choice`s in a `multipleChoice` environment to make a multiple choice question.

```
⟨*classXimera⟩
\newenvironment{multipleChoice}[1][]
{% Environment Start Code
\setkeys{multipleChoice}{#1}%
\recordvariable{\mc@id}%
\begin{trivlist}
\item[\hskip \labelsep\small\bfseries Multiple Choice:]\hfil
\begin{enumerate}
}% Note this means that \item has to be the first line after \begin{multipleChoice}.
{% Environment End Code
\end{enumerate}
\end{trivlist}
}

%multipleChoice@ is for internal use only! (used in wordChoice)
%It displays all choices in a list separated by /
\newenvironment{multipleChoice@}[1][]
{% Environment Start Code
\setkeys{multipleChoice}{#1}%
\ifthenelse{\equal{\mc@id}{}}% Test if \mc@ID is empty. This is not a robust check, is this
{}% Begin/End then result.
{% Begin else result
\immediate\write\idfile{var \mc@id;}
}% End else result
\begin{enumerate*}[before={\upshape{(\hspace{-.25em}}}, itemjoin={/\hspace{-.25em}}, after={
}
{% Environment End Code
\end{enumerate*}
}


%multipleChoice@@ is for internal use only! (used in wordChoice)
%this is simply a wrapper for the sole showing (other)choice.
\newenvironment{multipleChoice@@}[1][]{}{}
⟨/classXimera⟩
```

On the web, you might also expect these to be "problem environments" but they aren't – they're respondables. You might expect a `\setcounter{choiceId}{0}` here — that would be wrong, because then the generated IDs would no longer be unique.

```
⟨*htXimera⟩
\renewenvironment{multipleChoice}[1][]
{\setkeys{multipleChoice}{#1}%
\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div class="multiple-choice" ]
\ifthenelse{\equal{\mc@id}{}}{}{\HCode{data-id="\mc@id" }}%
\HCode{id="problem\arabic{identification}">}%
}{\HCode{</div>}\IgnoreIndent}
\ConfigureEnv{multipleChoice}{}{}{}{}
⟨/htXimera⟩
```

2.10 Word choice

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

```
⟨*classXimera⟩
\newcommand{\wordChoice}[1]{%
\let\choicetemp\choice% Assign a "choicetemp" command to duplicate choice.
\ifwordchoicegiven% If wordchoice option is on, we need to juggle around some definitions.
\let\choice\otherchoice%
\begin{multipleChoice@@}%
#1
\end{multipleChoice@@}%
\else% If it isn't the regular "choice" command should work.
\let\choice\inlinechoice%
\begin{multipleChoice@@}%
#1)%
\end{multipleChoice@@}%
\fi%
\let\choice\choicetemp% Now that choicetmp has been manipulated to what we want, replace cho
}%


⟨/classXimera⟩
```

This is actually just word choice

```
⟨*htXimera⟩
\renewenvironment{multipleChoice@}{\refstepcounter{problem}}{}%
\ConfigureEnv{multipleChoice@}{\stepcounter{identification}\IgnorePar\HCode{<span class="word
⟨/htXimera⟩
```

2.11 Select all

`selectAll` A multiple-multiple choice question

```
⟨*classXimera⟩
\newenvironment{selectAll}[1][]
{\begin{trivlist}\item[\hskip \labelsep\small\bfseries Select All Correct Answers:]\hfil\begi
    {\end{enumerate}\end{trivlist}}
⟨/classXimera⟩
```

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`.

```
⟨*htXimera⟩
\renewenvironment{selectAll}{\refstepcounter{problem}}{}%
\ConfigureEnv{selectAll}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<div
⟨/htXimera⟩
```

2.11.1 Free response

freeResponse A freeform input box.

```
⟨*classXimera⟩
\newboolean{given} %% required for freeResponse
\setboolean{given}{true} %% could be replaced by a key=value pair later if needed

\ifhandout
\newenvironment{freeResponse}[1][false]%
{%
\def\givenatend{\boolean{#1}}
\ifthenelse{\boolean{#1}}
{% Begin then result
\begin{trivlist}
\item
}% End then result
{% Begin else result
\setbox0\vbox\bgroup
}% End else result
% {}% Don't think this is doing anything? -- Jason
}
{%
\ifthenelse{\givenatend}
{% Begin then result
\end{trivlist}
}% End then result
{% Begin else result
\egroup
}% End else result
% {}% Don't think this is doing anything? -- Jason
}
\else
\newenvironment{freeResponse}[1][false]%
{% Environment Beginning Code
  \ifthenelse{\boolean{#1}}%% Could probably change this with just putting the (given) in the
   {% Begin then result
   \begin{trivlist}
   \item[\hskip \labelsep\bfseries Free Response (Given):\hspace{2ex}]
   }% End then result
{% Begin else result
\begin{trivlist}
\item[\hskip \labelsep\bfseries Free Response:\hspace{2ex}]
}% End else result
}
{% Environment Ending Code
\end{trivlist}
}
\fi

⟨/classXimera⟩
⟨*htXimera⟩

\renewenvironment{freeResponse}{\refstepcounter{problem}}{}%
\ConfigureEnv{freeResponse}{\stepcounter{identification}\ifvmode \IgnorePar\fi \EndP\HCode{<c

⟨/htXimera⟩
```

2.11.2 Feedback

feedback An initially hidden environment that uncovers itself at an appropriate time. New Validator rewrite code added by Jason Nowell. Original code orovided 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.

```
⟨*classXimera⟩
\newcommand{\PH@Command}{}
```

Validator should take an argument and detokenize it and display it at the start of the environment. The original Validator environment had everything framed in an mbox; presumably to make the text look a bit nicer, although this seems redundant with `texttt`. It shouldn't cause any harm so I have left it in for now.

```
\newenvironment{validator}[1][]{
\def\PH@Command{#1}% Use PH@Command to hold the content and be a target for "\expandafter" to
\mbox{\texttt{\detokenize\expandafter{\PH@Command}}}% Now expand PH@Command once and then det
}{}
```

First, if it's a handout, we want feedback to eat everything and then disappear entirely. So we use `\suppress`.

```
\ifhandout%
\newenvironment{feedback}{\suppress}{\endsuppress}
```

If this isn't a handout, then we want to display the Feedback by using a label, positioned and formated as a `\item` in a trivlist. It is important that we also detokenize the content of the optional argument, as it is likely to contain javascript or other code that latex won't be able to make sense of.

```
\else

\newenvironment{feedback}[1][attempt]{

\def\PH@Command{#1}% Use PH@Command to hold the content and be a target for "\expandafter" to

\begin{trivlist}% Begin the trivlist to use formating of the "Feedback" label.
\item[\hskip \labelsep\small\slshape\bfseries Feedback% Format the "Feedback" label. Don't fo
(\texttt{\detokenize\expandafter{\PH@Command}}):% Format (and detokenize) the condition for f
\hspace{2ex}]\small\slshape% Insert some space before the actual feedback given.
}{
\end{trivlist}
}

\fi
⟨/classXimera⟩
```

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

```
⟨*htXimera⟩
\def\feedback{\@ifnextchar[{\@feedbackcode}{\@feedbackattempt}}
\def\@feedbackattempt{\@feedbackcode[attempt]}
\def\@feedbackcode[#1]{\stepcounter{identification}%
\ifvmode \IgnorePar\fi \EndP%
\ifthenelse{\equal{#1}{attempt}}{\HCode{<div class="feedback" data-feedback="attempt" id="fee
{\ifthenelse{\equal{#1}{correct}}{\HCode{<div class="feedback" data-feedback="correct" id="fe
{\HCode{<div class="feedback" data-feedback="script" id="feedback\arabic{identification}"><sc
\def\endfeedback{\HCode{</div>}\IgnoreIndent}
⟨/htXimera⟩
```

2.12 Support for the web

2.12.1 MathJax support

When using mathjax, dump all the `\newcommand`s to a `.jax` file.

First, create the `.jax` file.

```
⟨*classXimera⟩
\ifdefined\HCode
  \else
    \newwrite\myfile
    \immediate\openout\myfile=\jobname.jax
\fi
⟨/classXimera⟩
```

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

```
⟨*classXimera⟩
\ifdefined\HCode
  \else
    \immediate\write\myfile{\unexpanded{\newenvironment}{prompt}{}{}}
\fi
⟨/classXimera⟩
```

Redefine newcommand appropriately.

```
⟨*classXimera⟩
\ifdefined\HCode
  \else
\let\@oldargdef\@argdef
\long\def\@argdef#1[#2]#3{%
\immediate\write\myfile{\unexpanded{\newcommand}{\unexpanded{#1}}[\unexpanded{#2}]{\unexpande
\@oldargdef#1[#2]{#3}%
}

\let\@OldDeclareMathOperator\DeclareMathOperator
\renewcommand{\DeclareMathOperator}[2]{\@OldDeclareMathOperator{#1}{#2}\immediate\write\myfil

\fi
⟨/classXimera⟩
```

Include the jax'ed newcommands

```
⟨*cfgXimera⟩
% 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\\([0-9]\string\\)/\string#\string\\1/g"

\Configure{BVerbatimInput}{}{}{}{}

\Configure{verbatiminput}{}{}{}{}

% 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 newcommands 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]{%
\renewenvironment{#1}{\NoFonts}{\EndNoFonts}
\ScriptEnv{#1}{\ifvmode \IgnorePar\fi \EndP\HCode{<script type="math/tex; mode=display"> \st
}
```

This is a fix for the LAODE book, which uses matlabEquation as if it were an equation

```
\ScriptEnv{matlabEquation}{\ifvmode \IgnorePar\fi \EndP\HCode{<script type="math/tex; mode=d

\VerbMath{equation}
\VerbMath{equation*}
\VerbMath{align}
\VerbMath{align*}
\VerbMath{alignat}
\VerbMath{alignat*}
\VerbMath{eqnarray}
\VerbMath{eqnarray*}

⟨/cfgXimera⟩
```

2.12.2 Semantic HTML

\textbf Using \textbf emits a <strong> tag.

```
⟨*cfgXimera⟩
\Configure{textbf}{\ifvmode\ShowPar\fi\HCode{<strong>}}{\HCode{</strong>}}
⟨/cfgXimera⟩
```

\textit Using \textit or similar emits an <em> tag.

```
⟨*cfgXimera⟩
\Configure{textit}{\ifvmode\ShowPar\fi\HCode{<em>}}{\HCode{</em>}}
\Configure{emph}{\ifvmode\ShowPar\fi\HCode{<em>}}{\HCode{</em>}}
⟨/cfgXimera⟩
```

\texttt Using \texttt emits a <code> tag.

```
⟨*cfgXimera⟩
\Configure{texttt}{\ifvmode\ShowPar\fi\HCode{<code>}}{\HCode{</code>}}
⟨/cfgXimera⟩
```

2.13 Tools

2.13.1 Suppress

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.

```
⟨*classXimera⟩
\font\dummyft@=dummy \relax
\def\suppress{%
  \begingroup\par
  \parskip\z@
  \offinterlineskip
  \baselineskip=\z@skip
  \lineskip=\z@skip
  \lineskiplimit=\maxdimen
  \dummyft@
  \count@\sixt@@n
  \loop\ifnum\count@ >\z@
    \advance\count@\m@ne
    \textfont\count@\dummyft@
    \scriptfont\count@\dummyft@
    \scriptscriptfont\count@\dummyft@
  \repeat
  \let\selectfont\relax
  \let\mathversion\@gobble
  \let\getanddefine@fonts\@gobbletwo
  \tracinglostchars\z@
  \frenchspacing
  \hbadness\@M}
\def\endsuppress{\par\endgroup}
⟨/classXimera⟩
```

2.13.2 The End

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

```
⟨*htXimera⟩
\Hinput{ximera}
⟨/htXimera⟩
```

```
⟨*htXourse⟩
\Hinput{xourse}
⟨/htXourse⟩
```

```
⟨*cfgXimera⟩
\begin{document}
\EndPreamble
⟨/cfgXimera⟩
```

3 xourse.cls

```
⟨*classXourse⟩
```

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

```
\newif\ifnotoc
\notocfalse
\DeclareOption{notoc}{\notoctrue}
```

`nonewpage` The default behavior of the class is to start each activity on a new page. This option will start activities without making a new page.

```
\newif\ifnonewpage
\nonewpagefalse
\DeclareOption{nonewpage}{\nonewpagetrue}
```

```
\DeclareOption*{\PassOptionsToClass{\CurrentOption}{ximera}}
\ProcessOptions\relax
\LoadClass{ximera}
%    \begin{macrocode}
⟨/classXourse⟩
```

3.1 Activities

The core of the `xourse` system. It works by redefining the `document` environment, thus making the `\begin` and `\end{document}` of the subfile 'transparent' to the inclusion. The redefinition of `\documentclass` is analogous, just having a required and an optional arguments which mean nothing to `\subfile`.

```
⟨*classXourse⟩
```

```
\newcommand{\skip@preamble}{%
    \let\document\relax\let\enddocument\relax%
    \newenvironment{document}{\let\input\otherinput}{}%
    \renewcommand{\documentclass}[2][subfiles]{}}
```

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
```

Store usual `\maketitle` as `\othermaketitle`

```
\let\othermaketitle\maketitle
```

`\maketitle` In a xourse file, `\maketitle` is redefined to give course packet title page and toc.

```
\renewcommand{\maketitle}{ %
\pagestyle{empty}
\begin{center}
~\\ %puts space at top of page to move title down.
\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
  \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
⟨/classXourse⟩
```

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.

```
⟨*classXourse⟩
\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
⟨/classXourse⟩
```

```
⟨*htXourse⟩
\renewcommand\activity[2][]{%
\ifvmode \IgnorePar\fi \EndP\HCode{<a class="activity card \activitystyle" href="#2" data-opt
}
⟨/htXourse⟩
```

When running xake, we can just ignore activities

```
⟨*classXourse⟩
\ifxake
\renewcommand\activity[2][]{}
\fi
⟨/classXourse⟩
```

3.1.2 Practice activities

\practice Like \activity but not expecting a title.

```
⟨*classXourse⟩
\ifhandout
\newcommand{\practice}[2][]{
\setkeys{practice}{#1}%!!!!!
  \renewcommand{\input}[1]{}
  \begingroup\skip@preamble\otherinput{#2}\endgroup
  \let\input\otherinput}
\else
\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
⟨/classXourse⟩
```

The practice environment does nothing, but will eventually produce exercises at the end of an activity

```
⟨*classXourse⟩
\ifxake
\renewcommand\practice[2][]{}
\fi
⟨/classXourse⟩
```

I suppose it is reasonable for practice cards to NOT have an activitystyle, since the activitystyle is basically PRACTICE.

```
⟨*htXourse⟩
\renewcommand\practice[2][]{%
  \ifvmode\IgnorePar\fi\EndP%
  \HCode{<a class="activity card practice" href="#2" data-options="#1">#2</a>}%
  \IgnoreIndent%
}
⟨/htXourse⟩
```

3.2 Sectioning

Makes the table of contents look a bit better. This can be redefined in the preamble if
\section you do not like the appearance. The name of a section inside an activity.

```
⟨*classXourse⟩
\renewcommand*\l@section{\@dottedtocline{1}{1.5em}{4.2em}}
⟨/classXourse⟩
```

`\subsection` The name of a subsection inside an activity.

```
⟨*classXourse⟩
\renewcommand*\l@subsection{\@dottedtocline{2}{3.8em}{4.2em}}
⟨/classXourse⟩
```

`\part` Xourse files can have parts. The name of a large part of a xourse.

```
⟨*htXourse⟩
\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</h1>}%
\IgnoreIndent%
}
⟨/htXourse⟩
```

`\paragraph` Paragraph commands emit spans. A small heading.

```
⟨*cfgXimera⟩
\renewcommand{\paragraph}[1]{%
  \HCode{<span class="paragraphHead">}%
  #1%
  \HCode{</span>}\par\IgnorePar}
⟨/cfgXimera⟩
```

`\subparagraph` An even smaller heading.

```
⟨*cfgXimera⟩
\renewcommand{\subparagraph}[1]{%
  \HCode{<span class="subparagraphHead">}%
  #1%
  \HCode{</span>}\par\IgnorePar}
⟨/cfgXimera⟩
```

3.3 Grading by points

`graded` 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.

```
⟨*classXourse⟩
\newenvironment{graded}[1]{}{}
⟨/classXourse⟩
```

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

```
⟨*htXourse⟩
\renewenvironment{graded}[1]{%
\ifvmode \IgnorePar\fi \EndP\HCode{<div class="graded" data-weight="#1">}\IgnoreIndent%
}{
\ifvmode \IgnorePar\fi \EndP\HCode{</div>}}\IgnoreIndent%
}
⟨/htXourse⟩
```

3.4 Ungraded activities

`ungraded` 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.

```
⟨*classXimera⟩
\newenvironment{ungraded}{}{}
⟨/classXimera⟩
```

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

```
⟨*htXimera⟩
\renewenvironment{ungraded}[1]{%
\ifvmode \IgnorePar\fi \EndP\HCode{<div class="ungraded">}\IgnoreIndent%
}{
\ifvmode \IgnorePar\fi \EndP\HCode{</div>}}\IgnoreIndent%
}
⟨/htXimera⟩
```

3.5 Logos

`\logo` A logo for the xourse.

```
⟨*classXourse⟩
\newcommand*{\logo}[1]{%
  \ifx\@onlypreamble\@notprerr
    \ClassError{xourse}{logo can only be used in the preamble}
      {Move your logo command to the preamble}
  \else %
    \IfFileExists{#1}%
      {\gdef\xourse@logo{#1}}%
      {\ClassError{xourse}{logo file does not exist}
        {To use logo, make sure that the referenced image file exists}}%
  \fi%
}

⟨/classXourse⟩
```

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

```
⟨*htXourse⟩
\Configure{@HEAD}{%
  \HCode{<meta name="og:image" content="}%
  \xourse@logo%
  \HCode{" />\Hnewline}}
⟨/htXourse⟩
```