edutex

D. P. Story
Email: dpstory@acrotex.net
processed August 28, 2009

Contents

1 Introduction 1
2 Declare Options 1
3 Development of the \edutex-type commands 2
  3.1 Text Entry ........................................... 3
  3.2 Multiple Choice ....................................... 4
  3.3 Multiple Selection .................................... 6
  3.4 Dropdown Menu ........................................ 8
  3.5 JavaScript support for processing questions .......... 10
  3.6 Some Web links ....................................... 11

1 ⟨∗package⟩

1 Introduction

The resource for this package is the EduTeX Wiki page.

2 Declare Options

\newif\ifedutex@preview
\newif\ifedutex@practice

preview

Preview Mode. The preview mode uses only the functionality available through LaTeX with the hyperref package.

• small, quickly generated files;
• ordinary links should work, but:
• Buttons, text entry areas, etc. are typeset as boxes or images; they look real but don’t do anything.
Potential users would use this to browse available content; see EduTeX Explorer. Course or test developers would use it in the development process. We would use it to impress potential funding sources of the importance of our project. None of these applications need advanced functionality.

Another objective is to give us experience working with the syntax. With a preview mode working we can start converting content display and postpone functionality issues.

4 \DeclareOption{preview}{\edutex@previewtrue\edutex@practicetrue}

**Practice Mode.** Roughly speaking this mode should provide all functionality that is easily achievable and stable enough to be part of core code. This will change over time.

It should **not** include:

- functions used to parse a response
- linking or interaction to specific external programs.

It should include:

- Processing of multiple-choice or multiple-selection problems
- dropdown menus
- numerical entry

There are two motivations:

- Educational effectiveness of practice materials depends much more on content than document functionality. Even when extensive functionality is available it may make sense to disable it to reduce file size.
- In the startup period it is more important to get something up and running than to wait for functionality.

5 \DeclareOption{practice}{\edutex@previewfalse\edutex@practicetrue}

We now process the options. If no option is passed, we are in for **For Credit Mode.**

6 \ProcessOptions

We use some standard options for hyperref. We hard-wire in the use of pdftex.

7 \RequirePackage[pdftex,bookmarks,bookmarksnumbered,bookmarksopen,\%
8 colorlinks,pagebackref,pdfpagemode=UseNone,\%
9 pdfstartview=Fit,pdfpagelayout=SinglePage]{hyperref}\%
10 \AtBeginDocument{\begin{Form}}
11 \AtEndDocument{\end{Form}}
3 Development of the \edutex-type commands

We begin the construction of the \edutex command, it takes three arguments, the first of them determines how the command is expanded. The permissible values of the first argument are listed below.

\begin{itemize}
  \item \texttt{\def\edutex@entryarea{entryarea}}
  \item \texttt{\def\edutex@button{button}}
  \item \texttt{\def\edutex@JS{JS}}
  \item \texttt{\def\edutex@mchoice{multiplechoice}}
  \item \texttt{\def\edutex@mselection{multipleselection}}
  \item \texttt{\def\edutex@dropdown{dropdownmenu}}
  \item \texttt{\def\edutex@weblink{weblink}}
  \item \texttt{\def\edutex@launchurl{launchurl}}
\end{itemize}

Scratch box for this package.

\newbox\edutex@boxa
\edutex The central command for generating Edu\TeX commands. The general syntax of this command is \texttt{\edutex{<type>}{<label>}{<various>}}

This command has a lot of conditionals, it can be streamlined considerably.

\begin{verbatim}
\newcommand{\edutex}[1]{\def\edutex@argi{#1}\
  \ifx\edutex@argi\edutex@entryarea
    \def\edutex@continue{\edutex@set@entryarea}\
  \else\ifx\edutex@argi\edutex@JS
    \def\edutex@continue{\edutex@setJS}\
  \else\ifx\edutex@argi\edutex@button
    \def\edutex@continue{\edutex@setSubmitButton}\
  \else\ifx\edutex@argi\edutex@mchoice
    \def\edutex@continue{\edutex@setMC}\
  \else\ifx\edutex@argi\edutex@mselection
    \def\edutex@continue{\edutex@setMS}\
  \else\ifx\edutex@argi\edutex@dropdown
    \def\edutex@continue{\edutex@setDD}\
  \else\ifx\edutex@argi\edutex@weblink
    \def\edutex@continue{\edutex@setWebLink}\
  \else\ifx\edutex@argi\edutex@launchurl
    \def\edutex@continue{\edutex@setLaunchURL}\
  \fi\fi\fi\fi\fi\fi\fi\fi\edutex@continue
}\end{verbatim}

3.1 Text Entry

The command that sets a text field, the syntax is

\texttt{\edutex@set@entryarea{<label>}{xxxxxxx}}

or, from above,

\texttt{\edutex(entryarea){<label>}{xxxxxxx}}
Field Name. The name of the field generated by this command is $P\arabic{enumi}$-$#1$ where $#1$ is the label passed in the first argument of this command.

Proposed JS function to process this question type. The function to be used will be a modification of the exerquiz command $\text{\RespBoxMath}$. 

$$\texttt{\edutex{JS}{<label>}{NumValEq({ans:1});}}$$

$$\texttt{\edutex{JS}{<label>}{FuncValEq({ans:"x-x^2",interval:[0,1]});}}$$

The first of these is used for numerical answers. $\text{NumValEq}$ basically would use $\text{\RespBoxMath}$ for the case of numerical responses, where only one sample point from the interval $[0,1]$ is used.

\begin{verbatim}
41 $\edef\edutex@previewreadonly{\ifedutex@preview readonly,\fi}\%$  
42 $\newcommand{\edutex@set@entryarea}[2]{%$  
43 $\setbox\edutex@boxa=\hbox{#2}\%$  
44 $\edef\rectwidth{\the\wd\edutex@boxa}\%$  
45 $\expandafter\TextField\expandafter[\edutex@previewreadonly$  
46 $width=\rectwidth,name={P\arabic{enumi}-#1}]{}%$  
47 $}$
\end{verbatim}

3.2 Multiple Choice

The syntax is

$$\texttt{\edutex{multiplechoice}{<label>}{%}$

\item First item
\item Second item
\item Third item
\item ...

\correctanswer{2}
}$

This expands to

\begin{verbatim}
\begin{enumerate}
\item[RB] First item
\item[RB] Second item
\item[RB] Third item
...
\end{enumerate}
\end{verbatim}

Where $\texttt{\edutex@correctanswer}$ is the answer passed as a parameter of the $\text{\correctanswer}$ command.

This command is used within the argument of the commands for $\text{multiplechoice}$ and $\text{multipleselection}$ to indicate which of the alternatives are correct. For multiple choice, it is a single number $\texttt{\correctanswer{2}}$ indicates that the second
item is the correct one. For multiples election, \texttt{\correctanswer{2,4}} indicates that the second and fourth items are the correct ones.

48 \newcommand{\correctanswer}[1]{\gdef{\edutex@correctanswer}{#1}}
49 \let{\edutex@correctanswer=\empty}

This command was created by combining parts of \texttt{\ChoiceMenu} and \texttt{\Radio}. The format for the radio buttons provided by \texttt{\ChoiceMenu} are not appropriate within an \texttt{itemize/enumerate} environment, as is desired here. So, it was necessary to create a radio button command, that obeyed the usual key-value pairs of hyperref.

50 \newcommand{\edutexRadio}[2]{%
51 \bgroup
52 \let{\Fld@name=\relax}
53 \let{\Fld@default=\relax}
54 \let{\Fld@value=\relax}
55 \def{\@currValue}{#2}%
56 \def{\Fld@width}{\DefaultWidthofChoiceMenu}%
57 \def{\Fld@height}{\DefaultHeightofChoiceMenu}%
58 \setkeys{Field}{#1}%
59 \if\Fld@hidden\def{\Fld@width}{1sp}\fi
60 \if\Fld@value\let{\Fld@value=\Fld@default}\fi
61 \leavevmode
62 \Hy@escapeform{\edutex@PDFForm@Radio
63 \pdfstartlink user {%
64 \edutex@PDFForm@Radio
65 /AP</N<</\@currValue\space <<>>>>>>%
66 }%
67 \relax
68 \MakeRadioField{\Fld@width}{\Fld@height}\pdfendlink
69 \egroup
70 }

This command is a modified version of the hyperref command \texttt{\PDFForm@Radio}. Added here is the \texttt{readonly} option: the ding type used for this radio button was changed to a circle, the boundary of the field (in Adobe Reader) is also a circle, giving it a distinctive appearance from the appearance of the multiple selection field.

73 \def{\edutex@PDFForm@Radio}{%
74 /Subtype/Widget%
75 \if\Fld@hidden /F 6\else /F 4\fi
76 /T{\Hy@escapestring{\Fld@name}}%
77 /FT/Btn%
78 \if\Fld@readonly/Ff 49153\else/Ff 49152\fi % modified
79 /H/P%
80 /BS<</W \Fld@borderwidth/S/\Fld@borderstyle>>%
81 /MK<<%
82 \ifnum\Fld@rotation=\z@
This command is the target of the command \edutex{multiplechoice}{{..}}{{..}}. The second two arguments are what \edutex@setMC defines below. The first argument is a label, the second is a \item list of alternatives. The lists is placed in an enumerate environment, and the \item is replaced by \item[radiobutton]. Because the list is given as a parameter of a command, verbatim text cannot appear in the list. A better choice would be to create an environment instead.

Field names generated. The field names of the radio button fields follow the following pattern:

\texttt{P\textarabic{enumi}-#1}

where \texttt{#1} is the label.

Proposed JS to process multiple choice.

\begin{verbatim}
processMC((ans:\edutex@correctanswer))
\end{verbatim}
3.3 Multiple Selection

This is a modified version of \texttt{PDFForm@Check} from \texttt{hyperref}. Changes include the \texttt{readonly} option, and the ability to give an export value to the check box, other than a export value of \texttt{Yes}.

\begin{verbatim}
def\edutex@PDFForm@Check{%
/Subtype/Widget%
/ifFld@hidden /F 6\else /F 4\fi
/ifFld@readonly/Ff 1\fi % modified
/T(\\Hy@escapestring{Fld@name})%
/Q \Fld@align
/BS<</W \Fld@borderwidth /S/\Fld@borderstyle>>%
/AP<H<<\@currValue\space <<>>>>% 
/MK<<%
/num\Fld@rotation=\z@
/else
/R \Fld@rotation
/fi
/ifx\Fld@rotation\relax
/else
/BC[\Fld@bordercolor]\relax
/else
/ifx\Fld@bordercolor]\relax
/else
/BG[\Fld@bcolor]\relax
/else
/CA(\\Hy@escapestring{\Fld@cbsymbol})% 
>>%
/DA(\ZaDb \strip@pt\Fld@charsize\space Tf%
/ifx\Fld@color@empty\else\space\Fld@color\fi)% 
/FT/Btn%
/H/Y%
/if\Fld@checked /V/\@currValue\else /V/Off\fi
}
\end{verbatim}
A special version of \CheckBox, the one argument is the base name for the field. Currently, the name of the field becomes \texttt{P\arabic{enumi}-#1.@currValue}, where \texttt{\arabic{enumi}} is the current problem number, and \texttt{@currValue} is the item number within the \texttt{enumerate} list created in \texttt{edutex@setMS}.

\begin{verbatim}
\newcommand{\edutexCheckBox}[1]{
  \edef\@currValue{\arabic{\edutex@enumctr}}\expandafter\CheckBox\expandafter\[\edutex@previewreadonly
  name=\texttt{P\arabic{enumi}-#1.@currValue},value=\@currValue\]{}}
\end{verbatim}

This is the target of the command \texttt{edutex{multipleselection}{...}{...}}. Again, we set up an \texttt{enumerate} environment to list the alternatives, each alternative begins with \texttt{\item[checkbox]}. The second argument cannot contain any verbatim commands.

\textbf{Field names generated.} The field names of the radio button fields follow the following pattern:

\texttt{P\arabic{enumi}-#1.@currValue}

where \#1 is the label, and \texttt{@currValue} is the value of the counter that enumerates this item, usually, this will be \texttt{\arabic{enumii}}. Each checkbox will have a different name, based on the label passed, the problem number, and the item number. The export values of the fields are number sequentially, 1, 2, 3, \ldots, n.

\textbf{Proposed JS to process multiple choice.}

\begin{verbatim}
processMC({ans:\edutex@correctanswer})
\end{verbatim}

Where \texttt{\edutex@correctanswer} is a text macro that expands to the argument of the \texttt{\correctanswer} command.

\begin{verbatim}
\newcommand{\edutex@setMS}[2]{% 
  \begin{enumerate}\let\edutex@save@item\item 
  \edef\edutex@enumctr{enum\romannumeral\the\@enumdepth}%
  \def\item{\@ifnextchar[{{\edutex@save@item}{% 
    \refstepcounter{\edutex@enumctr}%
    \edutex@save@item{\edutexCheckBox{#1}}\]}}#2
  \end{enumerate}\edef\edutex@tmp@exp{% 
    \noexpand\edutex{JS}{#1}{processMS({ans:\texttt{[\edutex@correctanswer]})}}}}% 
  \edef\edutex@tmp@exp{% 
    \noexpand\edutex{JS}{#1}{processMS({ans:\texttt{[\edutex@correctanswer]})}}
  \end{enumerate}\edef\edutex@tmp@exp{% 
    \noexpand\edutex{JS}{#1}{processMS({ans:\texttt{[\edutex@correctanswer]})}}% 
  \end{enumerate}\edef\edutex@tmp@exp{% 
    \noexpand\edutex{JS}{#1}{processMS({ans:\texttt{[\edutex@correctanswer]})}}}
\end{verbatim}

3.4 Dropdown Menu

This is a modified version of \texttt{\ChoiceMenu}, this was needed to support the \texttt{readonly} option.
\def\edutex@ChoiceMenu[#1][#2][#3]{% parameters, label, choices
  \def\Fld@name{#2}\
  \let\Fld@default\relax
  \def\Fld@value\relax
  \def\Fld@width{\DefaultWidthofChoiceMenu}\n  \def\Fld@height{\DefaultHeightofChoiceMenu}\n  \bgroup
    \Fld@menulength=0 %
    \@tempdima\z@\n    \@for\@curropt:=#3\do{%
      \expandafter\Fld@checkequals\@curropt==\%
      \Hy@StepCount\Fld@menulength
      \settowidth{\@tempdimb}{\@currDisplay} %
      \ifdim\@tempdimb>\@tempdima\@tempdima\@tempdimb\fi %
      \advance\@tempdima by 15\p@\n    }%\n
    \Field@toks={ }\n    \setkeys{Field}{#1} %
    \if\Fld@hidden\def\Fld@width{1sp}\fi\n    \if\Fld@value\relax\let\Fld@value\Fld@default\fi\n    \if\Fld@width<\@tempdima\if\dim<1\cm\@tempdima1\cm\fi\fi\n    \if\Fld@flags\@empty\fi %
    \edef\Fld@height{\the\Fld@menulength\Fld@charsize}\n    \if\Fld@readonly\def\Fld@flags{1}\fi %
    \@@Listbox{#3}\n  \egroup
}\n\@AddToEduTeXDD uses a core \LaTeX\ command, \g@addto@macro, to build a list of tokens that are to be saved in the command \edutex@DD. \@AddToEduTeXDD is
used in \edutex@setDD below.

\newcommand{\@AddToEduTeXDD}{\g@addto@macro\edutex@DD}

\edutex@setDD

The command is the target when \edutex{dropdownmenu}{...}{...} is executed. The full syntax is given below.

\edutex{dropdownmenu}{DDProb}{First Choice,Second Choice,Third Choice}

For the drop down menu, the items listed in the comma-delimited list are the appearance or face values of the menu. The export value of each are set to the numbers 1, 2, ..., n, in the order that they are listed.

Field Names. The field names for the check boxes being created use the following formula: P\arabic{enumi}-#1, where #1 is the first argument of \edutex@setDD, this is the label of the problem.

Proposed JS Method to Process this question. At this time, subject to change, we propose writing a function processDD, syntax is

\edutex{JS}{DDProb}{processDD({ans:2})}

Using an object as an argument allows easy to add parameters, as needed. In this example, the second listed item in the dropdown menu is the correct answer; hence, we see if the export value of the menu has a value of 2.

\newcommand{\edutex@setDD}[2]{\bgroup
\let\edutex@DD\@empty
\@tempcnta=0
\edef\tmp@exp{,,\@curropt=\the\@tempcnta}\
expandafter\@AddToEduTeXDD\expandafter{\tmp@exp}
\@for\@curropt:=#2\do{\wit\advance\@tempcnta1\relax
\edef\tmp@exp{,,\@curropt=\the\@tempcnta}\
expandafter\@AddToEduTeXDD\expandafter{\tmp@exp}}}

We use the hyperref command \ChoiceMenu, which calls \@ChoiceMenu. Before we execute \ChoiceMenu, we first assign a new definition to \@ChoiceMenu, our own, which is defined above.

\let\@ChoiceMenu\edutex@ChoiceMenu
\edef\tmp@exp{\noexpand\ChoiceMenu[\edutex@previewreadonly
\popdown,default=1,name=P\arabic{enumi}-#1]{}}

Each item in \edutex@DD begins with a comma (,), so we gobble up the first comma.

{\expandafter\@gobble\edutex@DD}}\tmp@exp\@CM
\egroup
3.5 JavaScript support for processing questions

When we are in “for credit” mode, we pass pass the code that is to be used to evaluate the problems. The syntax is

\edutex{JS}{<label>}{<JS function>}

This commands are executed following the problem statement, and are saved in the command \edutex@@JS using the \g@addto@macro command.

\let\edutex@@JS\@empty
\newcommand{\@AddToEduTeXJS}{\g@addto@macro\edutex@@JS}

The first parameter is the <label>, the second one is the JS function. It is important that the JS code be only a function of the form MyJSFunc(...), because, this command modifies this to read

MyJSFunc(\arabic{enumi},"P\arabic{enumi}-#1",...)

Where #1 is the label. "P\arabic{enumi}-#1" is the base name of the field being generated.

\newcommand{\edutex@setJS}{%}

When this macro executes, it defines \edutex@addProbData.

\edef\edutex@addProbData##1{%
\@AddToEduTeXJS\edutex@addProbData#2\string;r%
}
\edef\edutex@submitAction{%
app.launchURL("http://www.math.uakron.edu/~dpstory/",true)
}
\def\edutex@PDFForm@Push{%
/Subtype/Widget%
\ifFld@hidden /F 6\else /F 4\fi%}
\T{\Hy@escapestring{\Fld@name}}%}
\ifFld@readonly/Ff 65541\else/Ff 65540\fi% modified%}
\if num/Fld@borderwidth/S/Fld@borderstyle>>%}
\else
/R
\Fld@rotation=\z0}
\else
/BC[Fld@bordercolor]%
\fi%}

/A<</S/JavaScript/JS(\Hy@escapestring{\Fld@onclick})>>%
3.6 Some Web links

Two web links that follow the general Edu\TeX. The first is based in \href. Within a browser, the current PDF will be replaced by another page, if the PDF has form data, the data will be lost.

This version uses a JavaScript action that guarantees that the new page will be opened in a new window.

\begin{verbatim}
\newcommand{\edutex@setWebLink}{\href}
syntax #1=url, #2=text
\end{verbatim}

\begin{verbatim}
\newcommand{\edutex@setLaunchURL}{\hyper@normalise\edutex@@setLaunchURL}
\end{verbatim}

\begin{verbatim}
\edutex@@setLaunchURL[2]{\begingroup\Hy@pstringdef\Hy@pstringF{#1}\%\hyper@chars\leavevmode\pdfstartlinkattr{\Hy@setpdfborder\ifx\@pdfhighlight\@empty\else/H[@pdfhighlight]\fi\ifx\@filebordercolor\relax\else/C[@filebordercolor]\%\fi\fi\)
user{%/Subtype/Link%/\A<<%/S/JavaScript%/\JS(app.launchURL("\Hy@pstringF",true))\>
}\relax\Hy@colorlink\@urlcolor#2\%\close@pdflink
\endgroup}
\end{verbatim}
\endgroup
}
