The thorshammer Package

D. P. Story*
Email: `dpstory@acrotex.net`

processed November 30, 2019

Contents

1 Introduction **2**

2 Preliminaries **3**
2.1 Options for this package 3
2.2 Required packages 4

3 Setting the initial view **4**

4 Declaring instructor and class information **5**

5 Running headers **5**

6 Declaring a cover page **6**

7 Basic methods **7**
7.1 Configuring the basic methods experience 7
7.2 Post creation document assembly 8

8 Form field commands **9**
8.1 Controls above the `quiz` environment 9
8.2 Commands that usually follow the quiz 13
8.3 Controls inside the `quiz` environment 15

9 Field level JavaScript for form field commands **16**

[0]The author acknowledges Thorsten G. (a.k.a., Thor) who proposed this workflow and who contributed many ideas; Jürgen G. (a.k.a., Loki) also contributed many good ideas, enthusiasm, questions, bug detecting, and motivation. High regards and respect to both. D. P. Story (a.k.a., Odon).

10 Modifications and redefinitions of AeB **21**
10.1 Quiz components modified 21
10.2 Modify margin points markup 25
10.3 Modifications to the summary table 25
10.4 Boom! Thor's thunders: "Thor needs solutions!" 25
10.5 Modifications to the `web` package 27

11 The `useclass` option and above **27**
11.1 Declaring class members 27
11.2 Some process controls 28
11.3 Working with multiple quizzes in one source 29
11.4 Building quizzes with `makeClassFiles` & `\sadQuizzes` 30

12 Batch support files **36**
12.1 The container file for Thor's way 36
12.2 The batch termination file 38

13 Configuration files **39**
13.1 Class configureation 39
13.2 Load the configuration file 41

14 Index **42**

15 Change History **47**

```
1 ⟨*package⟩
```

1 Introduction

Thorsten G. has asked me to assist him in creating a quiz system, based on AcroTEX, to be delivered to his classes. His workflow for this assessment system is a follows:[1]

1. The `quiz` environment is used to pose the questions, which consist of MC, numerical fill-in the blank, and extended response questions.

   Though the `quiz` environment is used, the student does not get his/her score reported back immediately upon finishing the quiz.

2. When the student finishes the exam, taken in AR, he/she presses the End Quiz control and saves the document as `\jobname-ID.pdf`, where `ID` is a student identification. I am informed the ID is the student name.

3. At some point, the instructor's script moves the document to the instructors folder.

[1] As my occassional friend Jürgen says, this workflow is a real *hammer*, so I titled this package 'Thors(ten) hammer', or simply thorshammer.

4. The instructor opens the PDF and finishes marking the extended response questions and assigns a grade.

This package supports the Thorsten's workflow by providing the necessary form elements and JavaScript to carry out his(her) plan. What happens to the quiz after that, I do not know.

2 Preliminaries

```
\RequirePackage{xkeyval}
\edef\th@dquoteCat{\the\catcode`\"}
\catcode`\"=12\relax
```

2.1 Options for this package

`nocfg` If this option is taken, `thorshammer.cfg` is not input.

```
\DeclareOptionX{nocfg}{\let\th@loadCFG\dl@NO}
\let\th@loadCFG\dl@YES
```

`testmode` If this option is taken, quizzes can be used in the normal way.

```
\newif\ifthtestmode\thtestmodefalse
\DeclareOptionX{testmode}{\thtestmodetrue}
\DeclareOptionX{!testmode}{\thtestmodefalse}
```

`ordinary` This is an experimental option to see if we can produce a regular quiz with selected features of a thorshammer workflow.

```
\newif\ifthordinary \thordinaryfalse
\DeclareOptionX{ordinary}{\thtestmodetrue\thordinarytrue}
```

`useclass` Use this option to bring in addition code to declare each member of the class, to automatically build a quiz for each class member, to distribute these quizzes to a designated folder of the instructor, and to distribute the quizzes the respective class folder.

```
\newif\ifbasicmethods\basicmethodstrue
\newif\ifuseclassOpt\useclassOptfalse
\def\bUseClass{false}
\DeclareOptionX{useclass}{\useclassOpttrue
  \def\bUseClass{true}\basicmethodsfalse
}
```

`usebatch` This option should be used with the `useclass` option. When this option is taken, the no freeze quiz button is created. The batch sequence Thor's way does that, and the presence of the freeze button, the instructor may press it without thinking. We don't want that to happen.

```
\newif\ifth@allowfreeze \th@allowfreezetrue
\DeclareOptionX{usebatch}{\th@allowfreezefalse
  \ExecuteOptionsX{useclass}}
```

`batchdistr` This option declares the instructor's intention of using Acrobat to apply security using 'Thor distributes.sequ' or 'Thor protects and distributes.sequ' to apply security and to distribute the files to the students's folders. This option simply expands

`\distrToStudentsOff`, and redefines the two commands so the author can't use them. This option has no effect on writing quizzes to the instructor's folder.

```
\DeclareOptionX{batchdistr}{\ExecuteOptionsX{usebatch}%
  \AtEndOfPackage{\distrToStudentsOff
  \let\distrToStudentsOff\relax\let\distrToStudentsOff\relax}}
```

Process the options

```
\ProcessOptionsX
\edef\thOrdQz{\ifthordinary true\else false\fi}
```

2.2 Required packages

```
\RequirePackage{insdljs}[2019/08/06]
```

We use the `usealtadobe` option of insdljs, but not directly. If `\inputAltAdbFncs` is `\relax` than the functions have not already been input above thorshammer.

```
\ifx\usedAdbFuncs\dl@NO
  \def\inputAltAdbFncs{\InputIfFileExists{altadbfncs.def}%
    {\PackageInfo{insdljs}{Inputting code for usealtadobe option}}%
    {\PackageWarning{insdljs}{Cannot find altadbfncs.def.\MessageBreak
     Reinstall or refresh your file name database.}}%
     \let\usedAdbFuncs\dl@YES
\else
  \let\inputAltAdbFncs\relax
\fi
\inputAltAdbFncs
\RequirePackage{exerquiz}[2019/08/13]
\RequirePackage{eq-save}[2019/08/07]
\let\execjs\dl@YES
\@ifundefined{CommentStream}{\newwrite\CommentStream}{}
\def\csarg#1#2{\expandafter#1\csname#2\endcsname}
\providecommand{\eqSP}{\string\040}
\def\thPageOne{\setcounter{page}{1}}
```

3 Setting the initial view

We require the document to be opened on the first page, but the initial magnification is under the control of the document author.

`\setInitMag{fitpage|actualsize|fitwidth|fitheight|fitvisible|inheritzoom}`
This command determines the initial magnification. There are a choice of six values for the argument; the default is `fitpage`

```
\def\setInitMag#1{\setkeys{thim}{mag=#1}}
\define@choicekey+{thim}{mag}[\val\nr]%
  {fitpage,actualsize,fitwidth,fitheight,%
   fitvisible,inheritzoom}[fitpage]%
  {\edef\th@initmag{\@nameuse{dl@\val}}}
  {\PackageWarning{thorshammer}{%
    Bad choice for initial magnification,\MessageBreak
    permissible values are fitpage, actualsize,\MessageBreak
    fitwidth, fitheight, fitvisible, and\MessageBreak
    inheritzoom. Try again}}
\def\th@initmag{\dl@fitpage}
```

The `\addToDocOpen` is a command from insdljs. We turn off calculations as the student does not need to see the calculation icon each time s/he enters a response. When the instructor presses the Mark It button, calculations are turned on again. In the second line below, we set the initial view to page 1 and the magnification set by the `\setInitMag` command above.

```
\addToDocOpen{\JS{%
  var stmot=app.setTimeOut("this.calculate=false;",100);}}
\addToDocOpen{\GoToD[\Page{1}\th@initmag]}
```

4 Declaring instructor and class information

`\instrPath*{⟨path⟩}` `\instrPathIsCHTTP` The path to the instructor's folder. It is assumed that this ⟨*path*⟩ is an absolute path. If the star option is taken, then the path is relative to the current folder. The `\instrPathIsCHTTP` declaration is available if the path to the instructor is a WebDAV address. This info is passed on the a JavaScript method.

```
\def\instrPathIsCHTTP{\def\thInstrFS{CHTTP}}
\let\thInstrFS\@empty
\newcommand\instrPath{\@ifstar
  {\gdef\InstrPathFull{false}\instrPath@i}
  {\gdef\InstrPathFull{true}\instrPath@i}}
\def\instrPath@i#1{\gdef\InstrPath{"#1"}}
\def\InstrPathFull{true}
%\let\InstrPath\@empty
\def\InstrPath{this.path.replace(reRmFn,"")}
```

`\classPath*{⟨path⟩}` `\classPathIsCHTTP` The path to the class folder. It is assumed that this ⟨*path*⟩ is an absolute path. If the star option is taken, then the path is relative to the current folder. The `\instrPathIsCHTTP` declaration is available if the path to the class folders is a WebDAV address. This info is passed on the a JavaScript method.

```
\def\classPathIsCHTTP{\def\thClassFS{CHTTP}}
\let\thClassFS\@empty
\newcommand\classPath{\@ifstar
  {\gdef\ClassPathFull{false}\classPath@i}
  {\gdef\ClassPathFull{true}\classPath@i}}
\def\classPath@i#1{\gdef\ClassPath{"#1"}}
\def\ClassPath{this.path.replace(reRmFn,"")}
\def\ClassPathFull{true}
```

5 Running headers

`\thQzHeaderL` The scheme used here assumes no other LaTeX package has been used to take over the running headers (and footers). If that is the case, use the values of the commands below to design your own.{⟨*text*⟩} This is inserted into the left running header for the quiz pages.

```
\def\thQzHeaderL#1{\def\th@QzHeaderLQ{\makebox[0pt][l]{#1}}}
\def\th@QzHeaderL{\th@QzHeaderLQ}
\thQzHeaderL{Thor's Class}
\def\th@QzHeaderLS{\th@HeaderOffset\th@QzHeaderLQ}
```

`\thQzHeaderCQ{⟨text⟩}` This is inserted into the center running header for the *quiz* pages.

```
\def\thQzHeaderCQ#1{\def\th@QzHeaderCQ{\makebox[0pt][c]{#1}}}
\thQzHeaderCQ{Quiz \thQuizName}
\def\th@QzHeaderC{\th@QzHeaderCQ}
```

`\thQzHeaderCS{⟨text⟩}` This is inserted into the center running header for the *solution* pages.

```
\def\thQzHeaderCS#1{\def\th@QzHeaderCS{\makebox[0pt][c]{#1}}}
\thQzHeaderCS{Solutions: \thQuizName}
```

`\thQzHeaderR{⟨text⟩}` This is inserted into the right running header for the solution pages.

```
\def\thQzHeaderR#1{\def\t@hQzHeaderR{\makebox[0pt][r]{#1}}}
\thQzHeaderR{\thepage}
```

We apply the running headings, depending on whether web loaded by testing for the `webheadings` page style.

```
\@ifundefined{ps@webheadings}{%
  \def\th@setHeaders{%
  \renewcommand{\@oddhead}{\th@QzHeaderL\hfil\th@QzHeaderC\hfil
  \t@hQzHeaderR}\renewcommand{\@evenhead}{\@oddhead}}%
}{%
  \def\th@setHeaders{%
    \lheader{\th@QzHeaderL}%
    \cheader{\th@QzHeaderC}%
    \rheader{\t@hQzHeaderR}}%
}
```

Change the header for the solution section

```
\def\eq@normallheader{%
  \@ifundefined{ps@webheadings}{%
    \def\th@QzHeaderL{\th@QzHeaderLS}%
    \def\th@QzHeaderC{\th@QzHeaderCS}%
  }{%
    \lheader{\th@QzHeaderLS}%
    \cheader{\th@QzHeaderCS}%
  }
}
```

`\rhPgNumsOnly` Originally, this package only exhibited page numbers in the running header, expanding `\rhPgNumsOnly` in the preamble restores that original experience.

```
\def\rhPgNumsOnly{\thQzHeaderL{}\thQzHeaderCQ{}\thQzHeaderCS{}}
\AtBeginDocument{\th@setHeaders}
```

6 Declaring a cover page

`\DeclareCoverPage{⟨pgNum⟩}` A cover page, if declared, is appended to the beginning of the quiz. The page specified by ⟨*pgNum*⟩ is the cover page. The cover page is a single page and must occur prior to any quiz. Valid in the *preamble only.*

```
\newif\ifthCoverPage \thCoverPagefalse
\newcommand{\DeclareCoverPage}[1]{\thCoverPagetrue
  \def\thIsCP{true}\def\thCvrPg{#1}}
\def\thIsCP{false}\def\thCvrPg{0}
\@onlypreamble\DeclareCoverPage
```

7 Basic methods

Thor is tormenting me with the basic methods option. The basic methods option is no options other than perhaps `nocfgs`. As a consequence, the student names are not pre-filled into the name fields. When multiple quizzes are produced, they are named differently, `\jobname-1.pdf`, `\jobname-2.pdf`, and so on. A lot of work has gone in to the basic methods so it works link the non-basic methods (option `useclass` or higher). The commands `\instrPath` and `\classPath` are supported; to use the `\classMember` command, use `useclass` or higher. In addition to `\instrPath` and `\classPath`, we define special basic method commands, as describe in the next section.

7.1 Configuring the basic methods experience

`\useNameToCustomize` (Basic methods) `\useNameToCustomize` can be used to modify the file name of the quiz to include student name; the default is to use the original quiz file name. This command is implemented through the Freeze Quiz button. This command has no effect in the non-basic setting.

```
\def\useNameToCustomize{\def\thUseNameToCustomize{true}}
\def\thUseNameToCustomize{false}
```

`\enumQuizzes{`⟨*num*⟩`}` (Basic methods) This file specifies that the quizzes should be replicated ⟨*num*⟩ times and named `\jobname-1`, `\jobname-2`, ..., `\jobname-`⟨*num*⟩. The default is not to enumerate.

```
\def\enumQuizzes#1{\def\bUseClass{true}\basicmethodsfalse
  \ClassEntriestrue\def\ClassPathFull{true}\def\InstrPathFull{true}%
  \def\ClassPath{this.path.replace(reRmFn,"")}%
  \def\InstrPath{this.path.replace(reRmFn,"")}%
  \bgroup\@tempcnta#1\relax
  \@whilenum\@tempcnta>\z@\do{\classMember{}{}{}%
    \advance\@tempcnta\m@ne}\egroup
  \def\thEnumQuizzes{#1}\def\bEnumQuizzes{true}}
\def\thEnumQuizzes{0}\def\bEnumQuizzes{false}
```

`\distrQuizzes{{`⟨$folder_1$⟩`}{`⟨$folder_2$⟩`}...{`⟨$folder_n$⟩`}}` (Basic methods) If `\distrQuizzes` is used, `\enumQuizzes` command is ignored. The quizzes are enumerated, as described above, but the number of quizzes created is the number of folders declared. The script `\sadQuizzes` also distributes the individual quizzes to the appropriate folder, on the path determined by the `\classPath` command.

```
\newcommand{\distrQuizzes}{%
  \ifuseclassOpt
    \def\th@next{\PackageWarning{thorshammer}
    {Use have specified the useclass option or higher\MessageBreak
     yet you employ \string\distrQuizzes, these are\MessageBreak
     incompatible. Assuming the specified package option}}%
  \else
    \let\th@next\th@distrQuizzes
  \fi\th@next
}
\def\th@distrQuizzes{\def\bUseClass{true}\basicmethodsfalse
  \ClassEntriestrue\bgroup\@makeother\_\th@distrQuizzes@i}
\def\rmSTAR#1*\@nil{\def\@folder{#1}}
%\def\tstForSTAR#1{\tstForSTAR@i#1**\@nil}
\def\tstForSTAR#1*#2*\@nil{\def\@rgi{#1}\ifx\@rgi\@empty
  \def\ISSTAR{*}\rmSTAR#2\@nil\else\let\ISSTAR\@empty\fi}%
\def\th@distrQuizzes@i#1{\@tempcnta\z@
  \@tfor\@folder:=#1\do{\advance\@tempcnta\@ne
```

Determine if `\@folder` begin with *, remove it and return the path as `\@folder`

```
    \expandafter\tstForSTAR\@folder**\@nil
    \edef\x{\noexpand\classMember{}{}\ISSTAR{\@folder}}\x
  }\xdef\enumQuizzes{\the\@tempcnta}%
  \gdef\bDistrQuizzes{true}\egroup
}
\def\bDistrQuizzes{false}
```

Just auto-save the document - not recommended The `\executeSave()` command previously figured in importantly, as this package developed, use of `\executeSave()` cannot be recommended. This command was implemented early in the development process

```
\@ifundefined{executeSave}
  {\def\executeSave(){%
  console.println("automatically saving this file...");^^J%
  var retn=aebTrustedFunctions(this,aebDocSaveAs,%
  {cPath:this.path,bCopy:false})}}{}
```

`docassembly` The `docassembly` environment was created early in development and was meant to be used with `\executSave()`. The environment definition was updated to be equivalent to the `makeClassFiles` environment. An environment by the same name, and the same functionality, is defined in aeb_pro.

```
\@ifundefined{docassembly}
  {\newenvironment{docassembly}{%
    \execJS[\mkClFlsSpcls]{docassembly}}{\endexecJS}}
  {\renewenvironment{docassembly}{%
    \execJS[\mkClFlsSpcls]{docassembly}}{\endexecJS}}
```

7.2 Post creation document assembly

`\rasSolns` The `\sadQuizzes` command is placed within the `docassembly` or `makeClassFiles` environment, `\sadQuizzes` in the body of the environment.

```
\begin{docassembly}
\sadQuizzes
\end{docassembly}
\begin{document}
```

Originally, we defined a command `\rasSolns`, this command has been `\let` to `\sadQuizzes`, which now performs its duties.

8 Form field commands

We define two types of controls: (1) those placed outside the quiz; (2) those placed within the quiz.

8.1 Controls above the quiz environment

Commands that occur above the quiz environment

`\FirstName` `\LastName`

The student needs to sign in with his/her first and last name. The commands `\FirstName` and `\LastName` are defined for that purpose.

```
\ifbasicmethods\let\th@namePresets\@empty\else
```

If the option `useclass`, or higher, is taken, we make these fields read only and the JavaScript code of `\sadQuizzes` will fill name field in for the student.

```
\def\th@namePresets{\Ff\FfReadOnly\BC{}}\fi
\newcommand\FirstName[3][]{\th@bMrkQz\textField[%
  \presets{\th@namePresets}#1]{Name.first}{#2}{#3}}
\newcommand\LastName[3][]{\textField[%
  \presets{\th@namePresets}#1]{Name.last}{#2}{#3}}
```

The `\sadQuizzes` command uses the name fields to identify on which page a quiz begins. This worries me a little if a document designer places more than one name field for a quiz. We attempt to make the first use of the name field per quiz. We define `\th@bMrkQz`. These fields are place exactly once for each quiz and is attached to the name fields.

```
\def\th@bMrkQz{\@ifundefined{bMrkQz\currQuiz}
  {\rlap{\textField[\Ff\FfReadOnly\BC{}\BG{}]{bMrkQz}{0pt}{0pt}}%
   \@namedef{bMrkQz\currQuiz}{}}{}}
```

`\FullName[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` The first and last name fields are required; however, when `useclass` or higher is used, they are automatically filled in. The `\FullName` field uses the calculate event to extract the first and last names and displays them together in one field. The format for the this name field can be changed through the declaration `\thfullnameFmt`.

`\thfullnameFmt`

```
\def\th@fullnamePresets{\Ff\FfReadOnly\BG{}\BC{}}
\def\thfullnameFmt#1{\def\th@fullnameFmt##1##2{#1}}
\thfullnameFmt{#1+" "+#2}
\newcommand{\FullName}[3][]{\textField[%
```

```
  \presets{\th@fullnamePresets}#1\AAcalculate{%
  var fName=this.getField("Name.first").value;\r
  var lName=this.getField("Name.last").value;\r
  event.value=\th@fullnameFmt{fName}{lName};}]{FullName}{#2}{#3}}
```

`\pwdInstrFld[⟨options⟩]{⟨pwd⟩}{⟨wd⟩}{⟨ht⟩}` In this workflow, when the instructor opens a quiz file, he/she enters a password. On success, the non-extended response questions of the student's quiz is marked, and various hidden form elements are made visible.

`\pwdInstrFldTU{⟨pdfstr⟩}` can be redefined to provide a tool tip for this field.

```
\def\pwdInstrFldTU#1{\def\pwdInstrFld@TU{#1}}
\pwdInstrFldTU{Enter password to mark this quiz}
```

The definition of `\pwdInstrFld`

```
\newcommand{\pwdInstrFld}[4][]{% opts, pwd, wd, ht
  \@ifundefined{\currQuiz-nQs}{\def\nQs{0}}
    {\edef\nQs{\@nameuse{\currQuiz-nQs}}}%
  \textField[\cmd{\bParams{\currQuiz}{\nQs}{"#2"}\eParams}
  \Ff\FfPassword\AAkeystroke{\pwdKeyJS}
  \protect\AA\protect\Ff\TU{\pwdInstrFld@TU}#1%
]{pwdtxt}{#3}{#4}}
```

`\markQz[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` Loki suggested another idea to have the password field hidden until the instructor opens the file. Well if you are doing that, why have a password field? Instead, a push button is provided.

`\markQzFldCA⟨jsstr⟩` The caption for this button

```
\def\markQzFldCA#1{\def\markQzFld@CA{#1}}
\markQzFldCA{Mark It}
```

`\markQzFldTU⟨jsstr⟩` The tool tip for this button

```
\def\markQzFldTU#1{\def\markQzFld@TU{#1}}
\markQzFldTU{Press to mark this quiz}
```

Important! The code for `\markQz`. **Important!** For obvious reasons, we don't want the push button to be seen by the students. As a result, it is initially hidden. The key to having the push button visible when the instructor is the private JavaScript variable `_thorshammer` (this can be changed). The following code is placed in the `config.js` file of the instructor's Acrobat installation:

_thorshammer
config.js

```
    var _thorshammer=true;
```

Acrobat reads this file only once when it's opened. When the instructor opens the student's quiz PDF in Acrobat, some underlying JavaScript code tests whether the `_thorshammer` variable is defined and is `true`. If these conditions are met, JavaScript makes the `\markQuizFld` and `\freezeQz` fields visible.

Acrobat

```
\newcommand{\markQz}[3][]{%
  \@ifundefined{\currQuiz-nQs}{\def\nQs{0}}%
    {\edef\nQs{\@nameuse{\currQuiz-nQs}}}%
```

This text field is placed underneath the push button. It is the one that causes the `\markQz` field to be visible.

```
  \makebox[0pt][l]{\textField[\BC{}\BG{}\H{S}\AAformat{%
    var f=this.getField("MarkIt");\r
    var g=this.getField("freezeQz");\r
    if(typeof _thorshammer!="undefined" && _thorshammer){\r\t
    if(f!=null)f.display=display.visible;\r\t
} else{\r\t
    if(f!=null)f.display=display.hidden;\r\t
    if(g!=null)g.display=display.hidden;\r
}}]{hideTxtFldMI}{0pt}{0pt}}%
```

The push button seen by the instructor to mark the quiz.

```
  \pushButton[\cmd{\bParams{\currQuiz}{\nQs}\eParams}\F\FHidden
    \AAmouseup{\commonPassKey}\CA{\markQzFld@CA}
    \TU{\markQzFld@TU}\protect\AA\protect\F#1%
  ]{MarkIt}{#2}{#3}}
```

`\freezeQuiz[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` The `\freezeQuiz` makes all form fields readonly. After the instructor finishes marking the quiz, he/she presses the freeze quiz button before he moves it to the student's folder for review. This is done so the student cannot modify the quiz in any case and beg for more points. (They never beg for fewer points). The freeze quiz button makes itself hidden as well.

`\freezeQuizFldTU{⟨pdfstr⟩}` can be redefined to provide a tool tip for this field.

```
\def\freezeQuizFldTU#1{\def\freezeQuizFld@TU{#1}}
\freezeQuizFldTU{Make all fields readonly, cannot be undone}
```

`\freezeQuizFldCA{⟨pdfstr⟩}` can be redefined to provide a button caption for this field.

```
\def\freezeQuizFldCA#1{\def\freezeQuizFld@CA{#1}}
\freezeQuizFldCA{Freeze Quiz}
```

The definition of `\freezeQuiz`. If the `usebatch` option is taken, we do not create the push button.

```
\newcommand\freezeQuiz[3][]{\pushButton[\cmd{\let\%\defjsLB}
  \CA{\freezeQuizFld@CA}\F\FHidden
  \TU{\freezeQuizFld@TU}\AAmouseup{freezeQuizMU()}
  \protect\AA\protect\F
  #1]{freezeQz}{}{11bp}}
```

`\instrSave[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` A companion macro to `\freezeQuiz`. This macro is substituted for `\freezeQuiz` when the `usebatch` option is taken. The `\instrSave` and `\freezeQuiz` should not appear in the same document; we give them the same field name so the JavaScript treats them them the same, in terms of making them hidden and visible. `\ifth@allowfreeze`

`\instrSaveFldTU{⟨pdfstr⟩}` can be redefined to provide a tool tip for this field.

```
\def\instrSaveFldTU#1{\def\instrSaveFld@TU{#1}}
\instrSaveFldTU{Save and close this file to the current folder}
```

`\instrSaveFldCA{⟨pdfstr⟩}` can be redefined to provide a button caption for this field.

```
\def\instrSaveFldCA#1{\def\instrSaveFld@CA{#1}}
\instrSaveFldCA{Save \string& Close}
```

The definition of `\instrSave`. If the `usebatch` option is taken, we do not create the push button.

```
\newcommand\instrSave[3][]{\pushButton[%
  \CA{\instrSaveFld@CA}\F\FHidden
  \TU{\instrSaveFld@TU}\AAmouseup{%
    var f=this.getField("studentenGrade");\r
    var str=""+f.value;\r
    str=str.replace(/\string\s/g,"");\r
    if (str=="")\r\t
      app.alert("You did not award the student a final mark."
      +"\\n\\nAward the mark and then save.");\r
    else {\r\t
      aebTrustedFunctions(this,aebSaveAs);\r\t
    this.closeDoc(true);\r
  }}\protect\AA\protect\F
  #1]{freezeQz}{}{11bp}}
```

`\freezeOrSave[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` is the recommended way of inserting `\freezeQuiz` or `\instrSave`. If `usebatch` is taken, freezeQrSave expands to `\instrSave`; otherwise it expands to `\freezeQuiz`.

```
\AtEndOfPackage{\ifth@allowfreeze\let\freezeOrSave\freezeQuiz
  \else\let\freezeOrSave\instrSave\fi}
```

`\studentReport[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` In Thor's way of things, a summary report is placed at the top of the document. This readonly field shows the number of points awarded and the total points.

```
\newcommand{\studentReport}[3][]{%
  \textField[\BC{}\BG{}\F\FHidden\Ff\FfReadOnly\protect\Ff#1%
  ]{studentenReport}{#2}{#3}}
```

`\studentGrade[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` Again, in Thor's way of things, a text field is available to assign grade. This field is initially hidden, but becomes visible when instructor signs in.

```
\newcommand{\studentGrade}[3][]{\textField[\F\FHidden\protect\F
  \BC{red}\BG{}\Q1\textSize{12}\textColor{blue}
  \AAkeystroke{event.change=event.change.toUpperCase()}#1%
  ]{studentenGrade}{#2}{#3}}
```

`\thQHFirstName[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` The first name of the student

```
\def\thQHFirstName#1{\def\th@QHFirstName{\textbf{#1}\space}}
\thQHFirstName{First name:}
```

`\thQHLastName[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` The last name of the student

```
\def\thQHLastName#1{\def\th@QHLastName{\textbf{#1}\space}}
\thQHLastName{Last name:}
```

`\thQHPoints[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` Number of points, displayed in the form '10 / 20'.

```
\def\thQHPoints#1{\def\th@QHPoints{\textbf{#1}\space}}
\thQHPoints{Points:}
```

`\thQHGrade[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` Some grade mark (A, B, C, etc., or 1, 2, 3, etc.)

```
\def\thQHGrade#1{\def\th@QHGrade{\textbf{#1}\space}}
\thQHGrade{Grade:}
```

`\thQuizHeader*` The above commands typically appear above the quiz and placed in some beautiful way. We bundle these commands into a single one, according to my own happiness, but you may seek happiness some other way by redefining `\thQuizHeaderLayout`. (The command `\thQuizHeader` just picks up on the `*`-option, then expands `\thQuizHeaderLayout`.) The command is placed beneath the `\DeclareQuiz` command and above the `quiz` environment. The command automatically emits a `\newpage`, unless the `*`-option is taken.

Preferred Placement	**Alternate Placement**
`\DeclareQuiz{⟨qz-name⟩}`	`\begin{document}`
...	`\DeclareQuiz{⟨qz-name⟩}`
`\begin{document}`	`\thQuizHeader`
...	...
`\thQuizHeader`	⟨*quiz-begins*⟩
...	
⟨*quiz-begins*⟩	

```
\newcommand{\thQuizHeader}{\let\Hy@EveryPageAnchor\relax
  \@ifstar{\thPageOne\thQuizHeaderLayout}
  {\newpage\thPageOne\thQuizHeaderLayout}%
}
```

`\thQuizHeaderLayout` The body of this command contains the arraignment of the above defined commands. It is this command that may be redefined.

```
\newcommand\thQuizHeaderLayout{\noindent
  \th@QHFirstName\FirstName{1.5in}{13bp}\vcgBdry[3pt]
  \th@QHLastName\LastName{1.5in}{13bp}\vcgBdry[6pt]
  \begin{minipage}[t]{1.2in}\kern0pt
    \makebox[0pt][r]{\raggedleft\markQz{}{11bp}%
    \hspace{\marginparsep}}%
  \th@QHPoints\studentReport{\widthof{000/000}}{11bp}\vcgBdry[6pt]
    \makebox[0pt][r]{\raggedleft\freezeOrSave{}{11bp}%
    \hspace{\marginparsep}}%
  \th@QHGrade\studentGrade{14bp}{14bp}\vcgBdry[6pt]
  \end{minipage}\hfill
  \begin{minipage}[t]{\linewidth-1em-1.2in}\kern0pt
  \begin{sumryTblAux}{\currQuiz}
  \displaySumryTbl[ntables=1,showmarkup]{\currQuiz}
  \end{sumryTblAux}
  \end{minipage}}
```

This assumes the English language and the `usesumrytbls` option of exerquiz.

8.2 Commands that usually follow the quiz

`\completeMsgFldV{⟨pdfstr⟩}` is the message that is displayed in this multi-line text field.

`\completeMsgFld[⟨options⟩]{⟨wd⟩}{⟨ht⟩}` When the student completes the quiz, a hidden text field appears and reminds the student to save the document.

```
\def\completeMsgFldV#1{\def\completeMsgFld@V{#1}}
\completeMsgFldV{Congratulations, you have completed the quiz,
  before doing anything else, you need to save this document.}
```

The definition of `\completeMsgFld`

```
\newcommand{\completeMsgFld}[3][]{\textField[\F\FHidden\Ff\FfMultiline
  \Ff\FfReadOnly\V{\completeMsgFld@V}
  \DV{\completeMsgFld@V}]{postQzMsg}{#2}{#3}}
```

`\ShrtPtsFld{⟨options⟩}{⟨quiz-name⟩}` This is exerquizs `\PointsField`, but with special format script.

`\ShrtPtsFldFmt{⟨js-str⟩}` is the formatting string used; ⟨*js-str*⟩ should incorporate the event property `event.value` in its definition. See default definition below.

```
\def\ShrtPtsFldFmt{\bgroup\obeyspaces\ShrtPtsFldFmt@i}
\def\ShrtPtsFldFmt@i#1{\egroup\flJSStr[noquotes]{\ShrtPtsFld@Fmt}{#1}}
\ShrtPtsFldFmt{"Short Pts: "+event.value}
```

The definition of `\ShrtPtsFld`

```
\newcommand{\ShrtPtsFld}[2][]{%
  \PointsField[\AAformat{if(event.value!="")
    event.value=\ShrtPtsFld@Fmt}\F\FHidden\protect\AA
    \protect\F#1]{#2}}
```

`\LngPtsFld{⟨options⟩}{⟨quiz-name⟩}` This field will hold the total points for the essay or extended response questions. It is modeled after `\PointsField`, having the same defaults and width and height.

`\LngPtsFldFmt{⟨js-str⟩}` is the formatting string used; ⟨*js-str*⟩ should incorporate the event property `event.value` in its definition. See default definition below.

```
\def\LngPtsFldFmt{\bgroup\obeyspaces\LngPtsFldFmt@i}
\def\LngPtsFldFmt@i#1{\egroup\flJSStr[noquotes]{\LngPtsFld@Fmt}{#1}}
\LngPtsFldFmt{"Long Pts: "+event.value}
```

The definition of `\LngPtsFld`

```
\newcommand{\LngPtsFld}[2][]{%
  \textField[\presets{\PointsFieldDefaults}\F\FHidden
  \AAformat{if(event.value!="") event.value=\LngPtsFld@Fmt}
  \AAcalculate{var f=this.getField("essayMrkUp");\r
  if(f!=null)EFSimple_Calculate("SUM",%
new Array("essayMrkUp.\currQuiz"));}
  ]{EssayField.\currQuiz}{\PtFW}{\DefaultHeightOfWidget}}
```

`\TotalsFld{⟨options⟩}{⟨quiz-name⟩}` This is modeled after exerquizs `\PointsField`, but with special format script.

`\TotalsFldFmt{⟨js-str⟩}` is the formatting string used; ⟨*js-str*⟩ should incorporate the event property `event.value` in its definition. See default definition below.

```
\def\TotalsFldFmt{\bgroup\obeyspaces\TotalsFldFmt@i}
\def\TotalsFldFmt@i#1{\egroup\flJSStr[noquotes]{\TotalsFld@Fmt}{#1}}
```

```
\TotalsFldFmt{"Total: "+event.value+"\space\eqOutOf\space"%
+NPointTotal}
```

The definition of `\TotalsFld`

```
\newcommand{\TotalsFld}[2][]{%
  \textField[\presets{\PointsFieldDefaults}\F\FHidden
  \AAformat{try{event.value=(\TotalsFld@Fmt)}catch(e){}}
  \AAcalculate{EFSimple_Calculate("SUM",%
new Array("PointsField.\currQuiz","EssayField.\currQuiz"));\r
    var\eqSP f=this.getField("studentenReport");\r
    f.value=(1*event.value)+"\eqSP/\eqSP"+\theeqpointvalue;
  }]{TotalsField.\currQuiz}{\PtFW}{\DefaultHeightOfWidget}}
```

`\thQuizTrailer` The above commands can be arranged in some way following the quiz; one such arrangement is found in the command `\thQuizTrailer`.

```
\newcommand{\thQuizTrailer}{\raisebox{\baselineskip-\fboxsep}%
  {\makebox[0pt][l]{\parbox[t]{3in}{\kern0pt
  \completeMsgFld{3in}{3\baselineskip}}}}%
  \makebox[0pt][l]{\hspace{3in}\quad
    \ifthtestmode\CorrButton{\currQuiz}\else
    \stuSaveBtn{}{11bp}\fi}\parbox[t]{3in}
    {\ShrtPtsFld{\currQuiz}\vcgBdry[6pt]
    \LngPtsFld{\currQuiz}\vcgBdry[6pt]
    \TotalsFld{\currQuiz}}}
```

8.3 Controls inside the quiz environment

`\essayQ{⟨nPts⟩}` When we have an essay type question we need to mark it prior to the `\item`. In order to test whether the instructor has put in more credit than specified by the `\PTs` command, we need to pass the number of points for this question.

`\essayQFldTU{⟨pdfstr⟩}` is the tool tip for this field.

```
\def\essayQFldTU#1{\def\essayQFld@TU{#1}}
\essayQFldTU{Assign points to extended responses}
```

`\EsW` `\EsH` We fix the width `\EsW` and the height `\EsH` of `\essayQ` using commands, these can be redefined.

```
\def\EsW{33bp}\def\EsH{14bp}
```

Now for the definition of `\essayQ`

```
\def\essayQ#1{\let\qMark@HookSave\qMark@Hook
  \def\qMark@Hook{\makebox[0pt][r]{\smash
  {\raisebox{-7bp+\fboxsep}{\stepcounter{questionno}\textField[%
    \cmd{\bParams{#1}\eParams}\F\FHidden\Q{1}
    \AAkeystroke{\essayQKey}
%     \AAonfocus{var essayPtsAssigned=(1*event.value);}
    \AAformat{if(event.value!="") event.value=event.value
      +((event.value==1)?" \eqptLabel":" \eqptsLabel")}
  \TU{\essayQFld@TU}
  ]{essayMrkUp.\currQuiz.\thequestionno}{\EsW}{\EsH}%
  \addtocounter{questionno}{-1}}}}\global
\let\qMark@Hook\qMark@HookSave}}
```

The way you pose an essay question is as follows:

```
\essayQ{5}
\item\PTs{5} The question ...\\[3pt]
             \RespBoxEssay{4in}{4\baselineskip}
```

`\essayitem{⟨num⟩}` We simply this workflow a little, define `\essayitem`:

```
\def\essayitem#1{\essayQ{#1}\item\PTs{#1}}
```

Thus, we can now type,

```
\essayitem{5} The question ...\\[3pt]
              \RespBoxEssay{4in}{4\baselineskip}
```

9 Field level JavaScript for form field commands

`\pwdInstrFld` JS Keystroke action for `\pwdInstrFld`. This script uses three parameters passed to it through `\pwdInstrFld`: `@p(1)` is the quiz name (`\currQuiz`); `@p(2)` is the number of questions; and `@p(3)` is the password.

```
\begin{defineJS}[\catcode`\@=0\relax]{\pwdKeyJS}
if (event.willCommit) {
  if (event.value==@p(3)) {
    @commonPassKey
  }
}
\end{defineJS}
\begin{defineJS}[\catcode`\@=0\relax]{\commonPassKey}
```

Added code from `\qz@IDTxtField` to avoid the dreaded ‘q1 is undefined’ JavaScript error message. This happends when the Mark It control and the Begin Quiz controls are on different pages. When Mark It is pressed, ‘q1’ has not been defined yet, not until the next page.

```
if(typeof aQuizzesInDoc=="undefined")
  var aQuizzesInDoc=new Array();
if (aQuizzesInDoc.indexOf("@oField"))
  aQuizzesInDoc.push("@oField");
if (typeof @oField=="undefined")
  var @oField=new Object;
restoreQuizData();
this.calculate=true;
var f=this.getField("postQzMsg");
if (f!=null) f.display=display.hidden;
var f=this.getField("pbStuSvCl");
if (f!=null) f.display=display.hidden;
var f=this.getField("ScoreField.@p(1)");
if (f!=null) f.display=display.visible;
var f=this.getField("PointsField.@p(1)");
if (f!=null) f.display=display.visible;
```

```
var f=this.getField("EssayField.@p(1)");
if (f!=null) f.display=display.visible;
var f=this.getField("TotalsField.@p(1)");
if (f!=null) f.display=display.visible;
var f=this.getField("essayMrkUp");
if (f!=null) f.display=display.visible;
correctQuiz("@p(1)",@p(2));
var f=this.getField("qzreset");
if (f!=null) f.display=display.visible;
var f=this.getField("freezeQz");
if (f!=null) f.display=display.visible;
var f=this.getField("studentenReport");
if (f!=null) f.display=display.visible;
var f=this.getField("studentenGrade");
if (f!=null) f.display=display.visible;
if (typeof correctSumryTbl == "function")
  correctSumryTbl("@p(1)",@p(2));
\end{defineJS}
```

`\essayQKey` Keystroke JS action for `\essayQ`. The `@p(1)` parameter is the weight of this essay question, it is passed to this script by `\essayQ`.

`\NoNumEnteredMsg{`⟨*jsstr*⟩`}` When you enter a non-number, and an alert box pops up with this as its message.

```
\def\NoNumEnteredMsg#1{\flJSStr*[noquotes]{\cNoNumEnteredMsg}{#1}}
\NoNumEnteredMsg{"You did not enter a number, %
enter a nonnegative number only"}
```

`\TooMuchCreditMsg{`⟨*jsstr*⟩`}` When you assign too much credit for the problem, an alert box appears containing this message.

```
\def\TooMuchCreditMsg#1{\flJSStr*[noquotes]{\cTooMuchCredit}{#1}}
\TooMuchCreditMsg{"You've assigned too much credit for this %
problem, assigning the maximum instead"}
```

Now the definition of `\essayQKey`

```
\begin{defineJS}[\makeesc\@\catcode`\%=14\relax]{\essayQKey}
if (event.willCommit) {
  var qpts=(1*event.value);
  if (isNaN(qpts)) {
    app.alert(@cNoNumEnteredMsg);
    event.rc=false;
  } else if (qpts<0) {
    event.value=-1*event.value;
    qpts=1*event.value;
  }
  if (event.rc) {
    if (qpts > @p(1) ) {
      app.alert(@cTooMuchCredit);
      qpts=@p(1);
    }
    // update ProbDist array
    ProbDist[@thequestionno]=qpts;
    // see if table is present
    if (typeof correctSumryTbl == "function") {
      f=this.getField("%
@dlcombine(@currQuiz)(SanityCheckPts).@thequestionno");
      var thesePts= qpts + (( qpts == 1 )?%
" @eqptLabel":" @eqptsLabel");
      f.value=thesePts;
      // add color
      var cb=this.getField("%
@dlcombine(@currQuiz)(SanityCheck).@thequestionno");
      if (qpts==@p(1)) cb.strokeColor=@rghtColorJS;
      else if (qpts>0) cb.strokeColor=@partialColorJS;
      else cb.strokeColor=@wrngColorJS;
    }
    event.value=qpts;
  }
}
\end{defineJS}
```

`\instrAutoSaveOn` `\instrAutoSaveOff` When the instructor presses the freeze quiz control, there is an option to automatically save the document or not. `\instrAutoSaveOn` saves the document; however, if `\instrAutoSaveOff` is expanded in the preamble, no automatic save is performed. The default is `\instrAutoSaveOn`.

```
\def\instrAutoSaveOn{\def\instrAutoSave{true}}
\def\instrAutoSaveOff{\def\instrAutoSave{false}}
\instrAutoSaveOn
```

`\instrAutoCloseOn` `\instrAutoCloseOff` When the instructor presses the freeze quiz control, there is an option to silently close the document or not. `\instrAutoCloseOn` closes the document; however, if `\instrAutoCloseOff` is expanded in the preamble, no automatic closing occurs. The default is `\instrAutoCloseOn`.

```
\def\instrAutoCloseOn{\def\instrAutoClose{true}}
\instrAutoCloseOn
\def\instrAutoCloseOff{\def\instrAutoClose{false}}
```

`freezeQuizMU()` The mouse up JavaScript for `freezeQuiz()`. It makes all form fields *in the entire document* readonly. *Use only* after all markups are finished and document is ready to be moved into the student's folder.

```
\def\MarkWarningMsg#1{\dlJSStr*[noquotes]{\MarkWarning@Msg}{#1}}
\MarkWarningMsg{"You did not award the student a final mark.\
  \\n\\nAward the mark and then save."}
```

`\flattenOn` `\flattenOff` The `\flattenOn` turns on flattening, while `\flattenOff` turns flattening off. The reason you would turn flattening off is to use Thor's way for basic methods and for the `useclass` option. The default is `\flattenOff` for basic methods and `\flattenOn` for `usebatch`. Applies only when the `Freeze Quiz` button is present.

```
\def\flattenOn{\def\bFlattenState{false}}
\def\flattenOff{\def\bFlattenState{true}}
\ifbasicmethods\flattenOff\else\flattenOn\fi
```

The definition of `freezeMU()`.

```
\begin{insDLJS}{jsforthor}{thorshammer: Freeze/Save Doc}
var isthereCvrPg=\thIsCP;
var cvrPgNum="\thCvrPg";
function freezeQuizMU() {
var f, fname;
var bOK=true;
var f=this.getField("studentenGrade");
var str=""+f.value;
str=str.replace(/\s/g,"");
if (str=="") {
  app.alert(\MarkWarning@Msg);
  bOK=false;
}
```

Determine if there are solution pages, and if so, re-insert them.

```
var SolnSet=this.info.SolnSet;
if (bOK&&SolnSet!=""){
var SolnPath=this.info.SolnPath;
// var SolnSet=this.info.SolnSet;
var qzbasename=this.info.qzBaseName;
  aebTrustedFunctions(this,aebInsertPages,{
    nPage: (this.numPages-1),
    cPath: SolnPath+"/"+qzbasename+"-"+SolnSet+".pdf"
  })
};
```

If `\thUseNameToCustomize` is true, we use the current file name; otherwise we use the original file name (`\jobname`)

```
if(\instrAutoSave&&bOK) {
//  var cSave="\jobname";
  var docFN=this.documentFileName;
  docFN=docFN.substring(0,docFN.length-4);
  var cSave=(\thUseNameToCustomize)?"\jobname":docFN;
```

If `\thUseNameToCustomize` is true, we append student and `"-g"` to signal that this file has been graded.

```
  if(\thUseNameToCustomize) {
    var f=this.getField("Name.first");
    if(f!=null)cSave+=("-"+f.value+"_");
    f=this.getField("Name.last");
    if(f!=null)cSave+=(f.value);
    cSave+=("-g");
  }
  var oRetn=aebTrustedFunctions(this,aebBrowseForDoc,{bSave:true,%
cFilenameInit: cSave });
  bOK=(typeof oRetn=="object");
  if(bOK) {
```

If the file name and path are chosen, we make all files readonly

```
    for (var i=0; i<this.numFields; i++) {
```

```
   fname=this.getNthFieldName(i);
   f=this.getField(fname);
   f.readonly=true;
  }
```

After making all fields readonly, we hide the freeze quiz button itself.

```
  var f=this.getField("MarkIt");
  if (f!=null)f.display=display.hidden;
  f=this.getField("freezeQz");
  if (f!=null)f.display=display.hidden;
```

(2019/06/30) Jürgen suggested to flatten the document to add more security.

```
  if(typeof _flattenThisDoc=="undefined")this.flattenPages();
```

Now we are ready to save the file

```
  oRecordOfQuizData=undefined;
```

If instructor uses Thor's way, we don't want to reattach the solution page as it has already been reattached in this workflow.

```
  this.info.SolnSet="";
  var retn=aebTrustedFunctions(this,aebDocSaveAs,%
{cPath:oRetn.cPath,cFS:oRetn.cFS});
 }
}
if(\instrAutoClose&&bOK) this.closeDoc(true);
}
\end{insDLJS}
\begin{defineJS}[\catcode`\@=0\relax]{\freezeQuizMU}
var f, fname;
var bOK=true;
if(@instrAutoSave) {
 var cSave="@jobname";
 var f=this.getField("Name.first");
 if(f!=null)cSave+=("-"+f.value+"_");
 f=this.getField("Name.last");
 if(f!=null)cSave+=(f.value);
 var oRetn=aebTrustedFunctions(this,aebBrowseForDoc,{bSave:true,@%
cFilenameInit: cSave });
 bOK=(typeof oRetn=="object");
 if(bOK) {
```

If the file name and path are chosen, we make all files readonly

```
  for (var i=0; i<this.numFields; i++) {
   fname=this.getNthFieldName(i);
   f=this.getField(fname);
   f.readonly=true;
  }
```

After making all fields readonly, we hide the freeze quiz button itself.

```
  var f=this.getField("MarkIt");
  if (f!=null)f.display=display.hidden;
  f=this.getField("freezeQz");
  if (f!=null)f.display=display.hidden;
```

(2019/06/30) Jürgen suggested to flatten the document to add more security.

```
    this.flattenPages();
```

Now we are ready to save the file

```
    var retn=aebTrustedFunctions(this,aebDocSaveAs,@%
{cFS:oRetn.cFS,cPath: oRetn.cPath });
  }
}
if(@instrAutoClose&&bOK) this.closeDoc(true);
\end{defineJS}
```

10 Modifications and redefinitions of AeB

We modify various commands of exerquiz to conform the goals of the mighty Thor.

10.1 Quiz components modified

We begin by modifying the `\RespBoxEssay` action.

```
\def\@@RespBoxEssayActions{%
  \AA{\if\eqQuizType\isQZ
    \AAKeystroke{%
      if(event.willCommit){\jsR\jsT
      RecordPointValue(\eqPTs,\thequestionno);\jsR\jsT
      RecordProblemType("\eqQT",\thequestionno);\jsR
```

The next three lines are inserted. After user has left the text field, we determine if he/she did anything. If `event.value`, stripped of all white space, is empty, nothing was done and we mark the response as `undefined`; otherwise we mark it as `"<essay>"`. The fact that the `Responses` array is nonempty for this question will cause a check mark to appear in the summary table.

```
      var stripResp=stripWhiteSpace(event.value);\jsR\jsT
      if(stripResp=="")Responses[\thequestionno]=undefined;\jsR\jsT
      else Responses[\thequestionno]="<essay>";\jsR\jsT
      if ( typeof fieldPopTbl == "function" ) fieldPopTbl("\currQuiz");
      }\jsR
      if (!isQuizInitialized("\curr@quiz")) {\jsR\jsT
          \eqObjAlert\space eqAppAlert(%
              InitMsg("\bqlabelISO"),3);\jsR\jsT
          event.rc = false;\jsR
      }%
    }%
  \fi
  }
}
```

We redefine `\@initQuiz` from exerquiz to first test whether name fields have been entered.

```
\def\InitQzMsg#1{\flJSStr*[noquotes]{\InitQzMsg@Msg}{#1}}
\InitQzMsg{"You cannot begin the quiz before entering
```

```
  your first and last names in the fields provided.\n\n
  Enter the name as you are known in the class; otherwise,
  you will receive no credit for your work."}
\def\IfbQzChkSnippet{%
this.calculate=false;\jsR
if(\thOrdQz) bOk=true\jsR
else {\jsR\jsT
  var f=this.getField("Name.first");\jsR\jsT
  var str1=stripWhiteSpace(f.value);\jsR\jsT
  var f=this.getField("Name.last");\jsR\jsT
  var str2=stripWhiteSpace(f.value);\jsR\jsT
  bOk=(str1!=""&&str2!="");\jsR
}
if(bOk)}
\expandafter\def\expandafter\@initQuiz\expandafter
  {\expandafter\IfbQzChkSnippet\expandafter{\@initQuiz}
  else app.alert({cMsg:\InitQzMsg@Msg,cTitle:\ThorsAlert@Title});
}
```

`\postSubmitQuiz` Modify `\postSubmitQuiz`. When the End Quiz control is pressed, we make visible the post quiz message, placed in the document by the `\completeMsgFld` command.

```
\toks@=\expandafter{\postSubmitQuiz\t\t
  oRecordOfQuizData["ProbDist.\oField"]=ProbDist;\r\t\t
  oRecordOfQuizData["RightWrong.\oField"]=RightWrong;\r\t\t
  var f=this.getField("postQzMsg");\r\t\t
  if (f!=null) f.display=display.visible;\r\t\t
  var f=this.getField("pbStuSvCl");\r\t\t
  if (\stuAutoSave&&f!=null)f.display=display.visible;} %\r\t\t
\edef\postSubmitQuiz{\the\toks@}
```

The action for the End Quiz button, we modify it to give the student a chance to reconsider his decision to end the quiz.

`\EndQzWarningMsg{⟨jsstr⟩}` The message that appears on the alert box asking to student to verify ending the quiz.

```
\def\EndQzWarningMsg#1{\flJSStr*[noquotes]{\EndQzWarning@Msg}{#1}}
\EndQzWarningMsg{"When you end the quiz, you cannot change
any of your answers without starting the quiz over from the
beginning.\n\n Press \\"Yes\\" to end the quiz."}
\def\ThorsAlertTitle#1{\flJSStr*[noquotes]{\ThorsAlert@Title}{#1}}
\ThorsAlertTitle{"Thor's Hammer"}
```

`\eq@EndQzBtnScript` The modified script for the end of the quiz button. We rework the script of `\eq@@EndQuizButtonActions`, taken from exerquiz.

```
\begin{defineJS}[\makeesc\*\catcode`\%=14\relax]{\eq@EndQzBtnScript}
if (!isQuizInitialized("*currQuiz"))
  eqAppAlert(InitMsg("*bqlabelISO"),3);
else {
  var retn=app.alert({cMsg: *EndQzWarning@Msg,%
cTitle: *ThorsAlert@Title, nIcon: 2, nType: 2});
  if (retn==4) {
    if (*minQuizResp(*thequestionno)&&_ModalNotOn){
      *currQuiz.PtValues=(new %
Array(*pointValuesArray));
      ProbType=[*ptypeArray];
*if@inclkey%
      *currQuiz.CorrAns=(new %
Array(*corrAnsArray));
*fi%
      DisplayQuizResults("*currQuiz",*theeqpointvalue,%
*thequestionno);
      var h=this.getField("ScoreData.*currQuiz");
      h.value=Score+";"+NQuestions+";"%
+ptScore+";"+NPointTotal;
%      *eq@submitURL
      *postSubmitQuiz
      resetQuiz("*currQuiz");
    }
  }
}
\end{defineJS}
```

Now, we redefine `\eq@@EndQuizButtonActions` of exerquiz.

```
\def\eq@@EndQuizButtonActions{\A{\JS{\eq@EndQzBtnScript}}}
```

Add a SaveAs menu item to end of the quiz

`\stuAutoSaveOn` `\stuAutoSaveOff` When expanded in the preamble, a save button will appear (`\stuSaveBtn`) when the End Quiz control is pressed. A dialog appears to save the file, the student can choose the file location and the file name at that time. When `\stuAutoSaveOff` is in effect, the save button does not appear, and the student must press the save button the on Adobe Reader toolbar. The default is `\stuAutoSaveOn`.

```
\let\stuASOn\ef@YES
\def\stuAutoSaveOn{\let\stuASOn\ef@YES
  \def\stuAutoSaveScript{\t app.execMenuItem("SaveAs");\r}%
  \def\stuAutoSave{true}}
\def\stuAutoSaveOff{\let\stuASOn\ef@NO
  \let\stuAutoSaveScript\@empty
  \def\stuAutoSave{false}}
\stuAutoSaveOn
```

`\stuAutoCloseOn` This command is obeyed only if `\stuAutoSaveOn` is in effect. After the student presses the save button (`\stuSaveBtn`), the document is closed after the student save the document. Note that if the student cancels saving the document and if the document still needs saving, the document is not closed. The default is `\stuAutoCloseOn`.

```
\def\stuAutoCloseOn{\def\stuAutoCloseScript{\t
  if(!this.dirty)this.closeDoc(true);\r}%
  \def\stuAutoClose{true}}
\stuAutoCloseOn
\def\stuAutoCloseOff{\let\stuAutoCloseScript\@empty
  \def\stuAutoClose{false}}
```

\stuSaveBtnCA⟨*jsstr*⟩ The caption for this button

```
\def\stuSaveBtnCA#1{\def\stuSaveBtn@CA{#1}}
\stuSaveBtnCA{Save}
```

\stuSaveBtnTU⟨*jsstr*⟩ The tool tip for this button

```
\def\stuSaveBtnTU#1{\def\stuSaveBtn@TU{#1}}
\stuSaveBtnTU{Press to save and close the document}
```

\stuSaveBtn[⟨*options*⟩]{⟨*wd*⟩}{⟨*ht*⟩} This button is initially hidden and becomes visible within the student presses the End Quiz control; provided \stuAutoSaveOn is in effect. The button saves and optionally closes the document.

```
\newcommand\stuSaveBtn[3][]{\pushButton[\F\FHidden
  \CA{\stuSaveBtn@CA}\TU{\stuSaveBtn@TU}
```

Here, we leverage the new \cmd command to test if auto save is on, if not we gobble the \AAmouseup action.

```
  \cmd{\ifx\stuASOn\ef@NO\let\@eqAAmouseup\@gobble\fi}
  \AAmouseup{if(\stuAutoSave){\r
    \stuAutoSaveScript\stuAutoCloseScript
  }}\protect\AA\protect\F#1
]{pbStuSvCl}{#2}{#3}}
```

\DeclareQuiz{⟨*qz-name*⟩} We modify the \DeclareQuiz command of exerquiz, by appending some code that defines \eq@prior@endQuiz to write the number of questions and the number of points to the AUX file. This command *must appear* at the top of the source file, just after \begin{document} or in the *preamble*, it defines \currQuiz for the rest of the document, and write to AUX file. When using multi-quizzes in one source file, this package redefines the \currQuiz; for example, if originally, you declared \DeclareQuiz{Quiz1}, the first renditions uses the quiz name Quiz1a, the second Quiz1b, and so on. (Limit of 26 renditions). To preserve the original quiz name, we define \thQuizName, this command expands to \Quiz1 within all renditions; consequently, can be used in the running head to consistently display the quiz name.

Required in preamble or at top of file

\thQuizName

```
\let\DeclareQuizSAVE\DeclareQuiz
\def\DeclareQuiz#1{\def\thQuizName{#1}\th@DeclareQuiz{#1}}
\def\th@DeclareQuiz#1{\DeclareQuizSAVE{#1}%
  \expandafter\gdef\expandafter
  \eq@prior@endQuiz\expandafter{\eq@prior@endQuiz\wrtQzInfo}}
```

\thQzName{⟨*friendly-qz-name*⟩} This is the friendly (human readable) quiz name, suitable for use in the running header and elsewhere.

```
\def\thQzName#1{\def\thqzname{#1}}
\thQzName{\thQuizName}
```

```
\def\wrtQzInfo{\eq@IWAuxOut{\string
  \csarg\string\gdef{\currQuiz-nQs}{\thequestionno}^^J\string
  \csarg\string\gdef{\currQuiz-nPts}{\theeqpointvalue}}}
```

\eqQuizPointsMsg We modify \eqQuizPointsMsg, its default definition is the string

```
"\eqptScore\space"+ptScore+" \eqOutOf\space"+nPointTotal
```

but for Thor's way, we simplify to ptScore.

```
\renewcommand\eqQuizPointsMsg{ptScore}
```

10.2 Modify margin points markup

Make the markup boxes in the margins larger.

```
\renewcommand{\aeb@creditmarkup}{\bgroup
  \edef\markupWidth{\EsW}\edef\markupHeight{\EsH}%
  \textField[\Ff\FfReadOnly\BC{}\F\FHidden
  \textColor{\pcMarkupColor}\textSize{\markupTextSize}\autoCenter{y}%
  \DV{0 \eqptsLabel}\V{0 \eqptsLabel}]%
  {qMark.\currQuiz.\thequestionno.\arabic{qMarkCnt}}%
  {\markupWidth}{\markupHeight}\egroup}
```

10.3 Modifications to the summary table

We modify the summary table to make the markup points larger and left align the second column. Thor may come down on me with his mighty hammer, but I'll take the chance.

```
\def\eq@begintab{% second column left aligned
    \begin{tabular}[t]{llc}\sumryTblQ&\sumryTblR&\sumryTblP\\\sthline
    {\Large\strut}}%
\let\st@scndclmnSAVE\st@scndclmn
```

Offset the check boxes by 2bp to better align with the heading

```
\def\st@scndclmn{\kern2bp\st@scndclmnSAVE}
```

Increase the width of the markup boxes (from 12bp to 20bp, and change to a fixed text size

```
\def\stmarkupWidth{20bp} % normally 12bp
\def\stmarkupHeight{9bp} % unchanged
\def\stmarkupTextSize{8} % normally 0pt
```

Offset the check boxes by 2bp to better align with the heading

```
\def\stmarkupbox{\mbox} % normally {\makebox[0pt][l]}
%\def\sumrytbllinkHook#1{\the\value{page}}
\def\st@thrdclmn#1{\setLink[\linktxtcolor{black}
  \A{\JS{this.pageNum=(this.pageNum+#1-1)}}]{\sumrytbllinkHook{#1}}}
```

10.4 Boom! Thor's thunders: "Thor needs solutions!"

The package was complete, then it wasn't. \RespBoxEssay never supported solutions, so that needed to be fixed, now requiring exerquiz dated 2019/08/13 or later.

The first issue addressed here is the labeling of the solutions to the quiz. We try a simple enumeration of the solutions. For that, the \fancyQuizHeaders is used from exerquiz.

```
\fancyQuizHeaders
```

```
\setsolnspace{}
\let\FncyHdrsFmtNoTitleQuiz\@empty
```

The question numbers protrude into the left margin, to disguise this, wes shift the running header over a little

```
\setlength{\eflength}{\widthof{\textbf{00.}\space}}
\edef\th@leftShiftHdr{\the\eflength}
\def\th@HeaderOffset{\hskip-\th@leftShiftHdr\relax}
\def\doNotShirtSonsHdrs{\let\th@HeaderOffset\relax}
```

`\thQzSolnMrkr` `\thQzSolnMrkr` is a small text field that is inserted under the section title. This is used to identify on what page the solutions begin. Later used by `\sadQuizzes`.

```
\def\thQzSolnMrkr{\textField[\BC{}]{thsolns4.\currQuiz}{1bp}{1bp}}
```

The quiz numbers will go in the left margin, so we'll shift the section title over a little to disguise this.

```
\def\quizSolnsHeadnToc{\section*
  {\makebox[0pt][l]{\th@HeaderOffset
    \thQzSolnMrkr\sqslsectitle}}%
    \addcontentsline{toc}{section}{%
    \@ifundefined{web@latextoc}{}{%
    \ifx\web@latextoc\eq@YES\else
    \protect\numberline{}\fi}\sqslsectitle}}
\renewcommand\eq@sqslsectitle{Solutions to the Quiz}
```

`\myFQHFmt` describes the numbering scheme for the solutions.

```
\newcommand\myFQHFmt{%
  \string\bfseries\string\color{\fncyQHdrsColor}%
  \ifx\aebTitleQuiz\@empty
    \ifnum\@eqquestiondepth>0\relax
        \FncyHdrsFmtNoTitleQuiz\fi\else
        \aebTitleQuiz\protect\
        \ifnum\@eqquestiondepth=0\else\\\relax
        \FncyHdrsFmtQuestion\fi
  \fi %\space
  \ifcase\@eqquestiondepth
    \ifx\aebTitleQuiz\@empty\FncyHdrsFmtNoTitleQuiz\fi
    \or
      \string\llap{\arabic{eqquestionnoi}.\space}%
    \or
      \string\llap{\arabic{eqquestionnoi}.\space}%
      (\alph{eqquestionnoii})\space
    \or
      \string\llap{\arabic{eqquestionnoi}.\space}%
      (\alph{eqquestionnoii})%
      (\roman{eqquestionnoiii})\space
  \fi
}
\dclrFncyQzHdrsFmt{\myFQHFmt}
```

No return symbol or link to the question.

```
\let\ReturnTo\@gobbletwo
```

10.5 Modifications to the **web** package

The TEX template file (`tex-template.tex`, generated by `thmclass.ps1`) specifies the web package and uses many command particular to that package. Here, we create a special command to input customization commands that are specified in the `web.cfg` file. Place `\inputWebCfg` in the preamble to input the `web.cfg`; any `\ExecuteOptions` commands are ignored. Customization commands are placed between the two marks `\bWebCustomize` and `\eWebCustomize`.

`\inputWebCfg`

```
\let\bWebCustomize\endinput
\let\eWebCustomize\relax
\providecommand{\inputWebCfg}{%
  \let\bWebCustomize\relax
  \let\eWebCustomize\endinput
  \let\ExecuteOptions@SAVE\ExecuteOptions
  \let\ExecuteOptions\@gobble
  \makeatletter
  \InputIfFileExists{web.cfg}{}{}\makeatother
  \let\ExecuteOptions\ExecuteOptions@SAVE
  \let\bWebCustomize\endinput
  \let\eWebCustomize\relax
}
```

11 The useclass option and above

These options (`useclass`, `usebatch`, and `batchdistr`) are designed for the mass production of the quizzes, one for each student in the class. The quiz is built and saved (for each student), saved to the instructor's designated folder, as declared by `\instrPath`, and to the student's personal folder as declared within the `\classMember` entry and `\classEntries` array.

11.1 Declaring class members

In conjunction with `\instrPath` and `\classPath`, use `\classMember` to declare the identity of each member of the class.

`\classMember*{⟨first-name⟩}{⟨last-name⟩}*{⟨folder|path⟩}` Enter the first name, last name, and folder name of each student in the class. When the star form is used, ⟨*first-name*⟩ and ⟨*last-name*⟩ are first passed through `\pdfstringdef`. If the second star-open is specified between the second and third arguments, the third argument should be the absolute path to the (exceptional) student.

There are several ways of producing characters in the Latin-1 character set:

`\u`

- unicode method: `\classMember{J\u00FCrgen}{Loki}{B}`

`\classMember*`

- using `\pdfstringdef`, in this case use the star version of `\classMember` `\classMember*{J\"{u}rgen}{Loki}{B}`

`\oct` • octal method: `\classMember{J\oct374rgen}{Loki}{B}` or
`\classMember{J\string\374rgen}{Loki}{B}`

```
\def\classEntriesDef{["","","",false]}
\newif\ifClassEntries\ClassEntriesfalse
\let\classEntries\@gobble
\def\classMember{\ClassEntriestrue\@ifstar
  {\let\th@star\ef@YES\classMember@i}
  {\let\th@star\ef@NO\classMember@i}}
\newcommand\classMember@i[2]{%
  \@ifstar{\let\th@exstar\ef@YES\classMember@ii{#1}{#2}}
    {\let\th@exstar\ef@NO\classMember@ii{#1}{#2}}}
\newcommand\classMember@ii[3]{%
  \ifx\th@exstar\ef@YES\def\AbsPth{true}\else
    \def\AbsPth{false}\fi
  \ifx\th@star\ef@NO
    \ifx\th@exstar\ef@YES
      \g@addto@macro\classEntries{,["#1","#2","#3",true]}\else
      \g@addto@macro\classEntries{,["#1","#2","#3",false]}\fi
  \else
    \g@addto@macro\classEntries{,["}%
    \pdfstringdef\x{#1}\expandafter
    \g@addto@macro\expandafter\classEntries\expandafter{\x}%
    \g@addto@macro\classEntries{","}%
    \pdfstringdef\x{#2}\expandafter
    \g@addto@macro\expandafter\classEntries\expandafter{\x}%
    \g@addto@macro\classEntries{","}%
    \pdfstringdef\x{#3}\expandafter
    \g@addto@macro\expandafter\classEntries\expandafter{\x}%
    \ifx\th@exstar\ef@YES
      \g@addto@macro\classEntries{",true]}\else
      \g@addto@macro\classEntries{",false]}\fi
  \fi}
```

11.2 Some process controls

`\autoCopyOff` `\autoCopyOn` During document development, you don't want to copy the files each time you build and review the document. Set `\autoCopyOff` during quiz development, and declare `\autoCopyOn`. The default is `\autoCopyOn`.

```
\def\autoCopyOn{\def\autoCopy{true}}
\def\autoCopyOff{\def\autoCopy{false}}
\autoCopyOn
```

`\cFS{empty|CHTTP}` (This command is obsolete, and should be removed. Its functionality is accessed through the optional arguments of `\instrPath` and `\classPath`.) The `Doc.saveAs()` method has a `cFS` key for determining the file system, the value of the key is either empty or the string `CHTTP`. We offer this option. This key is recognized for the `\classPath`, we assume the `\instrPath` is in his local file system; however, it is easy to incorporate the `cFS` key here as well.

```
\newcommand{\cFS}[1]{\def\@rgi{#1}\ifx\@rgi\@empty
  \let\cFSth\@empty\else\def\cFSth{CHTTP}\fi}
\let\cFSth\@empty
```

\distrToStudentsOff
\distrToStudentsOn

Allow the instructor to turn off the distribution of the quizzes to the students' folders using `\distrToStudentsOff`, the default is `\distrToStudentsOn`.

```
\def\distrToStudentsOn{\def\distrToStudents{true}}\distrToStudentsOn
\def\distrToStudentsOff{\def\distrToStudents{false}}
```

\distrToInstrOff
\distrToStudentsOn

Allow the instructor to turn off the distribution of the quizzes to himself by using `\distrToInstrOff`, the default is `\distrToInstrOn`.

```
\def\distrToInstrOn{\def\distrToInstr{true}}\distrToInstrOn
\def\distrToInstrOff{\def\distrToInstr{false}}
```

(2019/08/26) Combined `\sadMultQuizzes` with `\sadQuizzes`

11.3 Working with multiple quizzes in one source

This section we develop some ideas of creating a single source file with multiple quizzes, these quizzes should be roughly equivalent. One such approach is to have a single quiz, and randomly permute the questions as well as randomly permute any MC or MS choice fields.

`\declareQuizBody{⟨name⟩}` This macro defines a verbatim environment with ⟨*name*⟩. Such an environment cuts and saves its contents under the name of ⟨*name*⟩`.cut`. It typically is designed to enclose the 'body of a quiz,' the 'quiz body' can then be input later using `\InputQuizBody`, define below. Associated with the declaration is a version number, `\QzVer`, which may be used in the titles, refer to `thexrt.tex` in the `examples/misc` folder.

\QzVer

```
\def\th@QzVer{0}
\def\QzVer{1}
\newcommand{\declareQuizBody}[1]{%
  \bgroup\@tempcnta\th@QzVer\relax
  \advance\@tempcnta\@ne
  \edef\th@qbCnt{\the\@tempcnta}%
  \csarg\xdef{#1-QzVer}{\th@qbCnt}\egroup
  \csarg\def{#1}{\immediate\openout\CommentStream #1.cut
    \let\verbatim@out\CommentStream
    \immediate\write\verbatim@out{\string
      \def\string\QzVer{\@nameuse{#1-QzVer}}}%
    \verbatimwrite}%
  \csarg\def{end#1}{\endverbatimwrite
    \immediate\closeout\CommentStream}}
```

`\InputQuizBody{⟨name⟩}` We input a 'quiz body' that has been earlier CUT and saved under the name of ⟨*name*⟩`.cut`.

```
\newcounter{th@qzCnt}
\def\theth@qzCnt{\alph{th@qzCnt}}
\newcommand{\InputQuizBody}[1]{\newpage %\thPageOne
  \@ifundefined{thisQuizOrig}{\edef\thisQuizOrig{\thisQuiz}
  \let\Hy@EveryPageAnchor\relax}{}\stepcounter{th@qzCnt}%
  \edef\x{\thisQuizOrig\theth@qzCnt}%
  \expandafter\th@DeclareQuiz\expandafter{\x}%
  \renewcommand\sqslsecrunhead{}%
  \InputIfFileExists{#1.cut}{}{}
```

Before we include quiz solutions, we close the `\quiz@solns` stream.

```
\immediate\closeout\quiz@solns %\th@QzHeaderLS
\let\eq@normallheader\relax
\newpage
\@ifundefined{ps@webheadings}{%
  \def\th@QzHeaderL{\th@QzHeaderLS}%
  \def\th@QzHeaderC{\th@QzHeaderCS}%
}{%
  \lheader{\th@QzHeaderLS}%
  \cheader{\th@QzHeaderCS}%
}
\includequizsolutions*\relax
\global\therearequizsolutionsfalse
\renewcommand\sqslsecrunhead{\eq@sqslsecrunhead}%
\eq@noformstrue
```

Putting `\eq@noformtrue` assures us that the solution file will not be input a `\end{document}`. Next, we open a new quiz solution file stream so the another rendition to write solutions to a fresh file.

```
\immediate\openout \quiz@solns \jobname.qsl
\@ifundefined{ps@webheadings}{%
  \def\th@QzHeaderL{\th@QzHeaderLQ}%
  \def\th@QzHeaderC{\th@QzHeaderCQ}%
}{%
  \lheader{\th@QzHeaderLQ}%
  \cheader{\th@QzHeaderCQ}%
}%
}
```

11.4 Building quizzes with `makeClassFiles` & `\sadQuizzes`

Central to this whole process is building customized quizzes. This done by the `\sadQuizzes` expanded within the `makeClassFiles` environment.

`makeClassFiles` is an `execJS` environment, the base name has been preset to be `mcfthor`. The contents of this environment is `\sadQuizzes`. Beginning the 2019/07/15 of insdljs, `execJS` has an option argument that is used to pass a command to the `.djs` file. `\oct` `\u` We use this to make special definitions to support `\oct` and `\u`.

```
\def\mkClFlsSpcls{\let\oct\eqbs\let\u\relax}
\newenvironment{makeClassFiles}{%
\execJS[\mkClFlsSpcls]{mcfthor}}{\endexecJS}
```

`\sadQuizzes` (save and distribute quizzes) is a script for the `makeClassFiles` environment.

```
\begin{makeClassFiles}
```

```
\sadQuizzes
\end{makeClassFiles}
\begin{document}
...
```

The above is placed just above `\begin{document}`. The script populates, for each entry in the `\classEntries` array, the `Name.first` and `Name.last` fields with the student's first and last name. It then saves a copy of the document to the instructor's folder under the name `\jobname-`⟨*first-name*⟩_⟨*last-name*⟩. It does the same thing for the student's private folder. Finally, it clears the `Name` fields, and saves itself to the source folder. The script may be redefined in the preamble of the document using the `defineJS*` environment. The script also deals with solution and cover pages.

```
\def\setClassArray{\ifClassEntries
  \classEntries\else\classEntriesDef\fi}
\def\setArrayLength{\ifbasicmethods0\else lst.length\fi}
\def\setfilesuffix{\ifuseclassOpt"-"+fN+"_"+lN\else
  \ifbasicmethods""\else"-"+(i+1)\fi\fi+".pdf"}
\begin{defineJS}[\def\defineJSjsR{^^J}\let\u\relax
  \catcode`\@=0\relax]{\sadQuizzes}
```

If `\autoCopyOff`, then this script does nothing

```
if(@bFlattenState)
  this.addScript({
    cName: "thorshammer: Do not flatten",
    cScript:"var _flattenThisDoc=false;"
  });
if (@autoCopy) {
```

JavaScript variables common to building quizzes

```
  var bUseClass=@bUseClass;
  var reRmFn=new RegExp(this.documentFileName,"i");
  var instrPath=@InstrPath;
  var cLast=instrPath[instrPath.length-1];
  if (cLast=="/")
    instrPath=instrPath.substring(0,instrPath.length-1);
  var classPath=@ClassPath;
  cLast=classPath[classPath.length-1];
  if (cLast=="/")
    classPath=classPath.substring(0,classPath.length-1);
  var thInstrFS="@thInstrFS";
  var thClassFS="@thClassFS";
  var isthereCvrPg=@thIsCP;
  var cvrPgNum="@thCvrPg";
  console.println("autocopy "+((@autoCopy)?"on":"off"));
  var retn;
  var solnSuffix="";
  var oSolnSuffix=new Object;
  var parentoDoc=this;
  var workingFolder=this.path;
  var pos=workingFolder.lastIndexOf("/");
  workingFolder=workingFolder.substring(0,pos+1);
  var _workingFolder="";
  // console.println("working folder: " + workingFolder);
  var willSaveScript='isAQuizUnfinishedAtSave();\r'
      +'if (oRecordOfQuizData !=undefined) collectQuizData();';
  var oHSD=this.getField("holdScoreData");
  var Rect=oHSD.rect;
  this.removeField("holdScoreData");
  var hsdFmt='if(typeof oRecordOfQuizData=="undefined")\r\t\
  oRecordOfQuizData=new Object;';
  var restQD='@restoreQD';
```

Cover page or pages. Determine if there are cover pages, if yes, extract it and save it to the instructor's folder. Cover pages are declared in the preamble with the command `\DeclareCoverPage`, the argument of which is either a single number (usually 0) of a range of zero-based page numbers.

```
  if(isthereCvrPg) {// -------- extract cover pages ------------
    var aCvrPgRng=cvrPgNum.split("-");
    if (aCvrPgRng[0]=="") {
      console.println("Start Range not specified, using 0 instead");
      var bPg=0;
    } else var bPg=aCvrPgRng[0];
    var ePg=(aCvrPgRng.length>1)?aCvrPgRng[1]:aCvrPgRng[0];
    var oDoc=aebTrustedFunctions(this,aebExtractPages,
      {nStart: bPg, nEnd: ePg});
    // save cover page(s) to the instructor folder
    _workingFolder=(@InstrPathFull)?"":workingFolder;
    aebDocSaveAs.msg="Cannot access the local folder "
      + _workingFolder+instrPath+"/@jobname-cvrpg.pdf";
    var retn=aebTrustedFunctions(oDoc,aebDocSaveAs,
      {cFS:thInstrFS,cPath: _workingFolder+instrPath+"/@jobname-"
      +"cvrpg.pdf",bCopy:false});
    oDoc.dirty=false;
    oDoc.closeDoc(true);
    // delete cover page before continuing, will reinsert it later
    this.deletePages({nStart: bPg, nEnd: ePg});
    this.dirty=false;
  }
// ------- end cover page code --------------
  var nQz=aQuizzesInDoc.length; // number of quizzes this doc
```

Solution pages. Before getting into generating the customized quizzes for the class, we must first see if there are any solutions in this document, if so, we separate the solution page(e) from the quiz.

```
  var bOkBasicSolns=false;
  var f=this.getField("thsolns4");
  if (f != null ) {
    console.println("There are solutions");
    bOkBasicSolns=true;
```

Save the solution suffixes for later use

```
    var g=f.getArray();
    for (var i=0; i<g.length; i++) {
      var solnSuffix=g[i].name; // eg solns4.q1a
      var pos=solnSuffix.indexOf(".");
      var qzName=solnSuffix.substring(pos+1);   // eg q1a
      solnSuffix=solnSuffix.replace(/\./g,"-"); // eg solns4-q1a
      // oSolnSuffix records whether a quiz has solution pages
      oSolnSuffix[qzName]=solnSuffix;
    }
```

Extract the solution page save it, close it, delete the page from master document

```
    // ------------- extraction--------------
// console.println("aQuizzesInDoc: " + aQuizzesInDoc.toSource());
// console.println("nQz="+nQz);
    for (var i=0; i< nQz; i++) {
      var f=this.getField("thsolns4");
      var g=f.getArray();
      var qzName=aQuizzesInDoc[i];
      var bOk2Extract=(typeof oSolnSuffix[qzName]!="undefined");
      if (bOk2Extract) {
// console.println(g[0].name + ", begins on page "+g[0].page);
        var bPg=g[0].page;
// console.println("bPg= " + bPg);
//      var f=this.getField("Name.first."+(i+1));
        var f=this.getField("bMrkQz."+(i+1));
        var ePg=(f==null)?(this.numPages-1):(f.page-1);
// console.println("ePg= " + ePg);
        var solnSuffix=g[0].name; // solns4.<qzName>
        solnSuffix=solnSuffix.replace(/\./g,"-"); // solns4-<qzName>
        var oDoc=aebTrustedFunctions(this,aebExtractPages,
          {nStart: bPg, nEnd: ePg});
        // save
        _workingFolder=(@InstrPathFull)?"":workingFolder;
        aebDocSaveAs.msg="Cannot access the local folder "
          + _workingFolder+instrPath+"/@jobname-"+solnSuffix+".pdf";
        var retn=aebTrustedFunctions(oDoc,aebDocSaveAs,
          {cFS:thInstrFS,cPath: _workingFolder+instrPath+"/@jobname-"
          +solnSuffix+".pdf",bCopy:false});
        // close
        oDoc.dirty=false;
        oDoc.closeDoc(true);
        // delete solution page before continuing
        this.deletePages({
          nStart: bPg,
          nEnd: ePg});
        this.dirty=false;
      }
    }
  }
```

```
// ------ begin creating custom quizzes for class members ------
  var cnt=0;    // determines which quiz to generate
  var lst=new Array(@setClassArray);
  var l=(bUseClass)?@setArrayLength:1; // dps
  for (var i=0; i < l; i++) {
    var qzName=aQuizzesInDoc[cnt];
// console.println("Working on " + qzName);
    var fN=lst[i][0];
    var lN=lst[i][1];
    var folder=lst[i][2];
    var isAbsPth=lst[i][3]; // dps
    if (folder!="")folder+="/";
    //  pre-populate with the student's name
    this.getField("Name.first").value=fN;
    this.getField("Name.last").value=lN;
    // extract quiz
//    var f=this.getField("Name.first."+cnt);
    var f=this.getField("bMrkQz."+cnt);
    var bPg=f.page;
//    var f=this.getField("Name.first."+(cnt+1));
    var f=this.getField("bMrkQz."+(cnt+1));
    var ePg=(f==null)?(this.numPages-1):(f.page-1);
```

Extraction: We extract a quiz from the master document. Extracting causes some problems: the restore quiz data is lost, the WillSave event is lost, and the `holdScoreData` field is lost. We try to overcome this problems.

```
    // ------------- extraction of quiz pages --------------
    if(bUseClass)var oDoc=aebTrustedFunctions(this,aebExtractPages,
      {nStart: bPg, nEnd: ePg});
    else oDoc=this; // dps
```

Restore the WillSave event.

```
    // extracting preserves doc JS but not doc actions
    oDoc.setAction({cTrigger: "WillSave", cScript: willSaveScript});
```

We have had problems with `oRecordOfQuizData` is `undefined`, to (finally) overcome this, we place some code at the document level.

```
    oDoc.addScript({
      cName: "oRecordOfQuizData Obj Declaration", cScript: hsdFmt});
```

The `holdScoreData` is on the first page of the document, this first page may not survive on the extracted pages, so we place this text field and the top of each of the first pages of the extracted quizzes.

```
    var oDocHSD=oDoc.addField({
      cName: "holdScoreData",
      cFieldType: "text",
      nPageNum: 0,
      oCoords: Rect
    });
```

Create a page open action to execute the restore quiz data

```
    oDoc.setPageAction({
      nPage: 0,
      cTrigger: "Open",
      cScript: restQD
    });
```

We save custom info to each of the files being saved: `StudentPath` has the path to the students' folders and `cFS` is the file system. These two are used by 'protect and distribute'.

```
    oDoc.info.qzBaseName="@jobname";
    if(isAbsPth) // dps
      oDoc.info.StudentPath=folder;
    else
      oDoc.info.StudentPath=classPath+"/"+folder;
    var bOkSolns=(typeof oSolnSuffix[qzName]!="undefined");
    if(bOkSolns) oDoc.info.SolnSet=oSolnSuffix[qzName];
    oDoc.info.SolnPath=_workingFolder+instrPath+"/";
    if(isthereCvrPg) oDoc.info.CvrPg="cvrpg";
    oDoc.info.cFS=thInstrFS;
```

Insert cover page, if any

```
    // Insert cover page, if any.
    if (isthereCvrPg) {
      if (typeof bCVMsg == "undefined") {
        console.println("Inserting cover page from "
          + _workingFolder+instrPath+"/@jobname-cvrpg.pdf");
        var bCVMsg=true;
      }
      if (cnt==0) var _cvrPath=_workingFolder+instrPath;
      aebTrustedFunctions(oDoc,aebInsertPages,
        {nPage: -1,
        cPath: _cvrPath+"/@jobname-cvrpg.pdf"});
    }
    var filesuffix=@setfilesuffix;
    // Now save this as a copy
```

Instructor's folder: Save a copy to the instructor's folder

```
    if(bUseClass) {
      _workingFolder=(@InstrPathFull)?"":workingFolder;
      aebDocSaveAs.msg="Cannot access the local folder "
        + _workingFolder+instrPath+"/@jobname-"+fN+"_"+lN+".pdf";
      if(@distrToInstr) retn=aebTrustedFunctions(oDoc,aebDocSaveAs,
        {cFS:thInstrFS,
         cPath: _workingFolder+instrPath+"/@jobname"+filesuffix,
         bCopy:true});
```

Student's folder: Save a copy to the student's folder

```
      var _workingFolderC=(@ClassPathFull)?"":workingFolder;
      if(isAbsPth)
        var cPath=folder+"@jobname"+filesuffix;
      else
```

```
        var cPath=_workingFolderC+classPath+"/"+folder
          +"@jobname"+filesuffix
      aebDocSaveAs.msg="Cannot access the path "+ cPath;
      if(@distrToStudents) retn=aebTrustedFunctions(oDoc,aebDocSaveAs,
        {cFS:thClassFS,
         cPath: cPath,
         bCopy:true});
      oDoc.dirty=false;
      oDoc.closeDoc(true);
    }
    cnt=++cnt % nQz
  }
}
this.resetForm(["Name"]);
console.println("automatically saving this file...");
```

Remove the following line
`// this.oRecordOfQuizData=undefined;`

Save the source document: Finally, save any changes that have occurred source document. Before saving, we make some adjustments in the case this is the basic method case.

```
var toSa=app.setTimeOut("aebTrustedFunctions(this,aebSaveAs);\
app.clearTimeOut(toSa);",50);
\end{defineJS}
\let\sadMultQuizzes\sadQuizzes
\let\rasSolns\sadQuizzes
⟨/package⟩
⟨*container⟩
```

12 Batch support files

These files are designed for a workflow wherein the `usebatch` option is taken. This option, though largely symbolic, declares the instructor is going to use a batch sequence to process the students' quizzes, after they have taken the quiz and returned to the instructor. A custom batch file `Thor's way` was written for this purpose.

Thor's way. This batch sequence is run after the instructor has has a final look at each quiz (additional markup needed in the case of essay questions and assigning a final mark. the batch sequence performs a number of actions, for each quiz file selected, it acquires minimal quiz data (first name, last name, number of points awarded, total points, and final mark.

12.1 The container file for Thor's way

This file should be brought into Acrobat, its fields filled, and the button pushed. It sets global JavaScript variables `global.qzName` and `global.gradedPath`.

```
\documentclass{article}
\usepackage[designi]{web}
\usepackage{eforms}
\hypersetup{pdfpagemode=UseAttachments}
%% \previewOn\pmpvOn
\parindent0pt \parskip6pt
\begin{defineJS}{\pbContainer}
var f=this.getField("qzName");
if(typeof global.RcrdData=="undefined")
  global.RcrdData=1;
```

If the ‘Record class data’ box is checked we create a new attachment to receive the data; otherwise, no new attachment file is created.

```
if(global.RcrdData) {
  global.qzName=f.value;
  if (global.qzName=="") {
    f.value="qzData";
    global.qzName="qzData.txt";
  } else global.qzName=f.value+".txt";
  var d=this.dataObjects;
  if (d!=null) {
    for(var i=0; i< d.length; i++)
      this.removeDataObject(d[i].name);
  }
  this.createDataObject({
    cName: global.qzName,
    cValue: "First\\tSecond\\tPoints\\tTotal\\tGrade"
  });
}
var _path=this.path;
var pos=_path.lastIndexOf("/");
global.containerPath=_path.substring(0,pos+1);
var f=this.getField("gradedPath");
var v=f.value;
var pos=v.indexOf(":");
if(pos!=-1||v[0]=="/") global.gradedPath=v;
else global.gradedPath=global.containerPath+v;
aebTrustedFunctions(this,aebSaveAs);
this.closeDoc(true);
\end{defineJS}
```

(2019/11/21) Put JS for Clear Btn in `defineJS` environment

```
\begin{defineJS}{\clrContainer}
this.resetForm(["qzName","gradedPath"]);
try{
```

(2019/11/21) Fixed a bug with exception thrown

```
  if (typeof global.qzName!="undefined")
    delete global.qzName; global.qzName="";
  if (typeof global.gradedPath!="undefined")
    delete global.gradedPath; global.gradedPath="";
  if (typeof global.appndSolns!="undefined")
    delete global.appndSolns; global.appndSolns=true;
  if (typeof global.RcrdDat!="undefined")
    delete global.RcrdDat; global.RcrdDat=true;
} catch(e){}
\end{defineJS}
\begin{document}
This file contains the quiz data as an attachment. Before you
start the batch action \textsf{Thor's way}, build and
\emph{place this file in the class folder of the instructor}.
\begin{center}
\begin{tabular}{rl}
\pushButton[\TU{Fill in the two fields then push this button
before starting the batch sequence}\CA{Push}\AAmouseup{\pbContainer}
]{pbContainer}{}{13bp}&%
\parbox[c]{1.5in}{\textField[\TU{Enter base name of the file that
stores quiz results}]{qzName}{1.5in}{13bp}\vcgBdry[3bp]
\textField[\TU{The path to the folder that will hold the graded
quizzes, it may be a relative or an absolute path}
]{gradedPath}{1.5in}{13bp}}\cgBdry[\columnsep]%\makebox[0pt][l]
{\pushButton[\CA{Clear}
  \AAmouseup{\clrContainer}]{clear}{}{13bp}}\\[12pt]
  \checkBox[\V{Yes}\DV{Yes}\AAmouseup{%
    global.appndSolns=(event.target.isBoxChecked(0));
}]{AppdSolns}{11bp}{11bp}{Yes}&%
\rlap{Append solutions, if they exist}\\[6pt]
\checkBox[\V{Yes}\DV{Yes}\AAmouseup{%
  global.RcrdData=(event.target.isBoxChecked(0));
}]{RecordData}{11bp}{11bp}{Yes}&%
\rlap{Record class data}
\end{tabular}
\end{center}
Fill in the base name of the file in the text field above. After
you push the button, the file is saved, then start
\textsf{Thor's way} action. After the batch sequence finishes,
this file is opened again. Open the attachments panel and save
the attached file. The file just saved is a tab delimited text
file that can be opened in Microsoft Excel.
\end{document}
⟨/container⟩
⟨*bterminate⟩
```

12.2 The batch termination file

When running Thor's way, it is important that `terminate-batch.pdf` is the last file processed by the batch. The batch test each file initially, if the current file being process, the batch exits 'gracefully', otherwise, the batch processes the current file. The role `terminate-batch.pdf` plays it that it is detected by the batch, it performs no actions.

```
\documentclass{article}
\usepackage[designi]{web}
\parindent0pt\parskip6pt
\begin{document}
\null\vfil
\begin{center}
\fbox{\begin{minipage}{.67\linewidth}
This file is the last to be processed by \textsf{Thor’s way},
the batch action identifies it and gracefully terminates.
\end{minipage}}
\end{center}
\vfil
\end{document}
⟨/bterminate⟩
⟨*package⟩
```

13 Configuration files

13.1 Class configureation

It was suggested by Loki that we should be able to have a configuration file for the class. In the case an instructor has several classes, this is a great convenience.

`\InputClassData{⟨base-name⟩}` Input the class file with name `⟨base-name⟩.cfg`. The class configuration file (`⟨base-name⟩.cfg`) consists of a series of declarations of the form,

```
\instrPath{⟨path⟩}
\classPath{⟨path⟩}
\classMember*{⟨first-name⟩}{⟨last-name⟩}{⟨folder⟩}
...
```

where the `*` is optionally included.

```
\def\InputClassData#1{\def\InputCl@ssData{#1}\mkClFlsSpcls
  \InputIfFileExists{#1.cfg}
   {\PackageInfo{thorshammer}{Inputting class file #1.cfg}}
   {\PackageWarning{thorshammer}
     {Cannot find the class file #1.cfg}}}
```

`\InputFormattedClass[⟨\cmd⟩]{⟨base-name⟩}` We define a more general input method. The format for the class configuration file is as follows:

```
\instrPath{⟨path⟩}
\classPath{⟨path⟩}
\bClassData
⟨entry1⟩
⟨entry2⟩
...
```

Prior to `\bClassData`, the entries are passed through, so you can, for example, set the `\instrPath` and `\classPath` here. After `\bClassData` comes a series of lines, one each student. Each line (⟨*entry*⟩) should be marked up so it can be parsed and information extracted; at the minimum, each line should contain ⟨*first-name*⟩, ⟨*last-name*⟩, and ⟨*folder*⟩, with possibly more. The default for ⟨`\`*cmd*⟩ is `\classMember`. For each line following `\bClassData`, ⟨`\`*cmd*⟩ is placed in front of each entry like so, ⟨`\`*cmd*⟩⟨*entry*⟩.

The first optional argument is the command (⟨`\`*cmd*⟩) that parses each line. The end purpose of the command is to construct a data structure,

`\classMember*{`⟨*first-name*⟩`}{`⟨*last-name*⟩`}{`⟨*folder*⟩`}`

A simple example is the following (filename is `myclassroll.cfg`):

```
 \instrPath{myClass}
 \classPath{/z/Users/thor/myClass}
 \bClassData
 {Peter}{Pan}{A}
*{J\"{u}rgen}{Loki}{B}
 {Thors}{Hammer}{C}
```

We can then input this CFG file with `\InputFormattedClass{myclassroll}`. For each line after `\bClassData`, the default ⟨`\`*cmd*⟩ is placed in front; for example, the first two lines of class data become `\classMember{Peter}{Pan}{A}` and `\classMember*{J\"{u}rgen}{Loki}{B}`, and so on.

```
\newcommand\InputFormattedClass[2][\classMember]{\ClassEntriestrue
  \begingroup
  \mkClFlsSpcls
  \endlinechar=-1
  \let\procThisLine\relax
  \let\bClassData\relax
  \let\re@dOK\dl@YES
  \newread\fmtclass
  \immediate\openin\fmtclass=#2.cfg
  \loop
    \read\fmtclass to \classmember
    \ifeof\fmtclass\let\re@dOK\dl@NO
    \else
      \expandafter
      \ifx\classmember\endinput\let\re@dOK\dl@NO
      \else
        \ifx\classmember\@empty %\let\procThisLine\relax
        \else
          \expandafter\procThisLine\classmember
          \expandafter\ifx\classmember\bClassData
```

After we process the current line, if this line is `\bClassData`, we switch over to `#1` as the definition of `\procThisLine`.

```
          \def\procThisLine{#1}\fi
        \fi
```

```
      \fi
    \fi
    \ifx\re@dOK\dl@YES\repeat
  \immediate\closein\fmtclass
  \endgroup
}
```

13.2 Load the configuration file

```
\def\th@InputCFG{\InputIfFileExists{thorshammer.cfg}
  {\PackageInfo{thorshammer}
    {Inputting the configuration file}}
  {\PackageInfo{thorshammer}
    {No configuration file found}}}
\ifx\th@loadCFG\dl@YES\expandafter\th@InputCFG\fi

\catcode`\"=\th@dquoteCat
⟨/package⟩
```

14 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols

`\%` . . . 207, 375, 563
`\@@RespBoxEssayActions` . . . 509
`\@eqAAmouseup` . . . 611
`\@eqquestiondepth` . . . 665, 668, 671
`\@evenhead` . . . 89
`\@folder` . . . 135, 140–142
`\@initQuiz` . . . 545, 546
`\@makeother` . . . 134
`\@oddhead` . . . 88, 89
`\@onlypreamble` . . . 111
`\@rgi` . . . 137, 732
`\_` . . . 134
`\␣` . . . 417, 667, 830, 1001

A

`\A` . . . 590, 645
`\AA` . . . 181, 201, 210, 228, 278, 510, 614
`\AAcalculate` . . . 170, 286, 297
`\AAformat` . . . 190, 277, 285, 296, 320
`\AAKeystroke` . . . 511
`\AAkeystroke` . . . 180, 237, 318
`\AAmouseup` 200, 209, 218, 612, 1065, 1073, 1074, 1078
`\AAonfocus` . . . 319
`\AbsPth` . . . 709, 710
`\addcontentsline` . . . 657
`\addtocounter` . . . 324
`\addToDocOpen` . . . 55, 57
`\aeb@creditmarkup` . . . 627
`\aebTitleQuiz` . . . 664, 667, 672
`\AtBeginDocument` . . . 106
`\AtEndOfPackage` . . . 22, 230
`\autoCenter` . . . 630
`\autoCopy` . . . 729, 730
`\autoCopyOff` . . . *28*, 730
`\autoCopyOn` . . . *28*, 729, 731

B

`\baselineskip` . . . 302, 304
`\basicmethodsfalse` . . . 16, 114, 133
`\basicmethodstrue` . . . 12
`batchdistr` (option) . . . *3*
`\BC` . . . 158, 164, 166, 190, 233, 236, 629, 653
`\bClassData` . . . 1118, 1132
`\bDistrQuizzes` . . . 144, 146
`\bEnumQuizzes` . . . 121, 122
`\bFlattenState` . . . 419, 420
`\BG` . . . 164, 166, 190, 233, 236
`\bParams` . . . 179, 199, 317
`\bqlabelISO` . . . 522
`\bUseClass` . . . 14, 16, 114, 133
`\bWebCustomize` . . . 686, 689, 696

C

`\CA` . . . 200, 208, 217, 610, 1065, 1072
`\cFS` . . . *28*, 732
`\cFSth` . . . 733, 734
`\cgBdry` . . . 1071
`\cheader` . . . 93, 102, 770, 782
`\checkBox` . . . 1074, 1078
`\classEntries` . . . 701, 713, 714, 716, 718, 719, 721, 722, 724, 726, 727, 789
`\classEntriesDef` . . . 699, 789
`\ClassEntriesfalse` . . . 700
`\ClassEntriestrue` . . . 115, 134, 702, 1113
`\classMember` . . . *27*, 119, 142, 702, 1113
`\classmember` . . . 1123, 1127, 1129, 1131, 1132
`\classMember*` . . . *27*
`\classMember@i` . . . 703–705
`\classMember@ii` . . . 706–708
`\ClassPath` . . . 72, 73, 116
`\classPath` . . . *5*, 69
`\classPath@i` . . . 70–72
`\ClassPathFull` . . . 70, 71, 74, 115
`\classPathIsCHTTP` . . . *5*, 67
`\clrContainer` . . . 1045, 1073
`\cmd` . . . 179, 199, 207, 317, 611
`\cNoNumEnteredMsg` . . . 369
`\color` . . . 663
`\columnsep` . . . 1071
`\CommentStream` . . . 40, 746, 747, 752
`\commonPassKey` . . . 200, 334
`\completeMsgFld` . . . *13*, 270, 304
`\completeMsgFld@V` . . . 267, 271, 272
`\completeMsgFldV` . . . *13*, 267, 268
`\CorrButton` . . . 306
`\csarg` . . . 41, 624, 625, 745, 746, 751
`\cTooMuchCredit` . . . 372

\curr@quiz 520
\currQuiz 163, 165, 177–179, 188, 189, 199, 263, 264, 288, 289, 298, 301, 306, 308–310, 323, 518, 624, 625, 632, 653

D

\dclrFncyQzHdrsFmt 684
\DeclareCoverPage *6*, 108, 111
\DeclareOptionX 5, 8, 9, 11, 15, 19, 21
\DeclareQuiz *24*, 616, 617
\declareQuizBody *29*, 741
\DeclareQuizSAVE 616, 618
\DefaultHeightOfWidget 289, 301
\define@choicekey 45
\defineJSjsR 793
\defjsLB 207
\displaySumryTbl 264
\distrQuizzes *7*, 123, 127
\distrToInstr 737, 738
\distrToInstrOff *29*, 738
\distrToInstrOn 737
\distrToStudents 735, 736
\distrToStudentsOff 22, 23, *29*, 736
\distrToStudentsOn *29*, 735
\dl@fitpage 54
\dl@NO 5, 27, 1124, 1127
\dl@YES 6, 32, 39, 1119, 1137, 1146
\dlJSStr 416
docassembly (environment) *8*
\documentclass 1008, 1093
\doNotShirtSonsHdrs 652
\DV 272, 631, 1074, 1078

E

\ef@NO 595, 611, 704, 707, 711
\ef@YES 591, 592, 703, 706, 709, 712, 725
\eflength 649, 650
\egroup 120, 144, 274, 281, 291, 633, 745
\emph 1061
\endexecJS 154, 156, 787
\endinput 686, 690, 696, 1127
\endlinechar 1116
\EndQzWarning@Msg 557
\EndQzWarningMsg *22*, 557, 558
\endverbatimwrite 751
\enumQuizzes *7*, 114, 143
environments:
  docassembly *8*
  makeClassFiles *30*
\eParams 179, 199, 317
\eq@@EndQuizButtonActions 590
\eq@begintab 634
\eq@EndQzBtnScript *22*, 563, 590
\eq@IWAuxOut 623
\eq@noformstrue 775
\eq@normallheader 96, 763
\eq@prior@endQuiz 620
\eq@sqslsecrunhead 774
\eq@sqslsectitle 661
\eq@YES 659
\eqbs 785
\eqObjAlert 521
\eqOutOf 292
\eqptLabel 321
\eqPTs 513
\eqptsLabel 321, 631
\eqQT 514
\eqQuizPointsMsg *24*, 626
\eqQuizType 510
\eqSP 42, 299, 300
\EsH *15*, 313, 323, 628
\essayitem *16*, 326
\essayQ *15*, 314, 326
\essayQFld@TU 311, 322
\essayQFldTU *15*, 311, 312
\essayQKey *17*, 318, 375
\EsW *15*, 313, 323, 628
\eWebCustomize 687, 690, 697
\execJS 154, 156, 787
\execjs 39
\ExecuteOptions 691, 692, 695
\ExecuteOptions@SAVE 691, 695
\ExecuteOptionsX 20, 21
\executeSave 148

F

\F 199, 201, 208, 210, 217, 228, 233, 235, 270, 278, 279, 284, 295, 317, 609, 614, 629
\fancyQuizHeaders 646
\fbox 1099
\fboxsep 302, 316
\Ff 158, 164, 166, 180, 181, 233, 270, 271, 629
\FfMultiline 270
\FfPassword 180
\FfReadOnly 158, 164, 166, 233, 271, 629
\FHidden 199, 208, 217, 233, 235, 270, 278, 284, 295, 317, 609, 629
\FirstName *9*, 159, 252
\flattenOff *18*, 420, 421
\flattenOn *18*, 419, 421

\flJSStr 274, 281, 291, 369, 372, 529, 557, 561
\fmtclass 1120, 1121, 1123, 1124, 1138
\FncyHdrsFmtNoTitleQuiz 648, 666, 672
\FncyHdrsFmtQuestion 669
\fncyQHdrsColor 663
\freezeOrSave *12*, 230, 231, 258
\freezeQuiz *11*, 207, 230
\freezeQuizFld@CA 205, 208
\freezeQuizFld@TU 203, 209
\freezeQuizFldCA *11*, 205, 206
\freezeQuizFldTU *11*, 203, 204
\freezeQuizMU 480
freezeQuizMU() (JS function) *18*
\FullName *9*, 169

G

\g@addto@macro 713, 714, 716, 718, 719, 721, 722, 724, 726, 727
\GoToD 57

H

\H 190
\Hy@EveryPageAnchor 247, 757
\hypersetup 1011

I

\ifbasicmethods 12, 157, 421, 790, 792
\IfbQzChkSnippet 534, 546
\ifClassEntries 700, 788
\ifeof 1124
\ifth@allowfreeze 18, 230
\ifthCoverPage 107
\ifthordinary 10, 25
\ifthtestmode 7, 306
\ifuseclassOpt 13, 124, 791
\includequizsolutions 772
\InitQzMsg 529, 530
\InitQzMsg@Msg 529, 547
\inputAltAdbFncs 28, 34, 36
\InputCl@ssData 1108
\InputClassData *39*, 1108
\InputFormattedClass *39*, 1113
\InputIfFileExists 28, 694, 761, 1109, 1141
\InputQuizBody *29*, 755
\inputWebCfg *27*, 688
\instrAutoClose 413, 415, 477
\instrAutoCloseOff *18*, 415
\instrAutoCloseOn *18*, 413, 414
\instrAutoSave 410, 411, 445
\instrAutoSaveOff *18*, 411
\instrAutoSaveOn *18*, 410, 412
\InstrPath 63, 65, 66, 117
\instrPath *5*, 60
\instrPath@i 61–63
\InstrPathFull 61, 62, 64, 115
\instrPathIsCHTTP *5*, 58
\instrSave *11*, 216, 231
\instrSaveFld@CA 214, 217
\instrSaveFld@TU 212, 218
\instrSaveFldCA *11*, 214, 215
\instrSaveFldTU *11*, 212, 213
\isQZ 510
\ISSTAR 138, 142
\item 326

J

\JS 55, 590, 645
JS functions:
  freezeQuizMU() *18*

K

\kern 254, 262, 303, 638

L

\Large 636
\LastName *9*, 161, 253
\lheader 92, 101, 769, 781
\linktxtcolor 644
\llap 674, 676, 679
\LngPtsFld *14*, 283, 309
\LngPtsFld@Fmt 281, 285
\LngPtsFldFmt *14*, 280, 282
\LngPtsFldFmt@i 280, 281
\loop 1122

M

\m@ne 120
\makeatletter 693
\makeatother 694
makeClassFiles (environment) *30*
\makeesc 375, 563
\marginparsep 256, 259
\markQz *10*, 187, 255
\markQzFld@CA 183, 200
\markQzFld@TU 185, 201
\markQzFldCA *10*, 183, 184
\markQzFldTU *10*, 185, 186
\markupHeight 628, 633
\markupTextSize 630
\markupWidth 628, 633

\MarkWarning@Msg . . . 416, 432
\MarkWarningMsg . . . 416, 417
\mkClFlsSpcls . . . 154, 156, 785, 787, 1108, 1115
\myFQHFmt . . . *26*, 662, 684

N

\n . . . 531, 560
\newwrite . . . 40
nocfg (option) . . . *3*
\NoNumEnteredMsg . . . *17*, 369, 370
\nQs . . . 177–179, 188, 189, 199
\null . . . 1097
\numberline . . . 660

O

\obeyspaces . . . 273, 280, 290
\oct . . . *28*, *30*, 785
\oField . . . 550, 551
\openin . . . 1121
\openout . . . 746, 776
options:
  batchdistr . . . *3*
  nocfg . . . *3*
  ordinary . . . *3*
  testmode . . . *3*
  usebatch . . . *3*
  useclass . . . *3*
ordinary (option) . . . *3*

P

\PackageInfo . . . 29, 1110, 1142, 1144
\PackageWarning . . . 30, 49, 125, 1111
\Page . . . 57
\parindent . . . 1013, 1095
\parskip . . . 1013, 1095
\pbContainer . . . 1014, 1065
\pcMarkupColor . . . 630
\pdfstringdef . . . 717, 720, 723
\pmpvOn . . . 1012
\PointsField . . . 277
\PointsFieldDefaults . . . 284, 295
\postSubmitQuiz . . . *22*, 549, 556
\presets . . . 160, 162, 170, 284, 295
\previewOn . . . 1012
\ProcessOptionsX . . . 24
\procThisLine . . . 1117, 1129, 1131, 1133
\providecommand . . . 42, 688
\PtFW . . . 289, 301
\PTs . . . 326
\pushButton . . . 199, 207, 216, 609, 1064, 1072
\pwdInstrFld . . . *10*, *16*, 176
\pwdInstrFld@TU . . . 174, 181
\pwdInstrFldTU . . . *10*, 174, 175
\pwdKeyJS . . . 180, 327

Q

\Q . . . 236, 317
\qMark@Hook . . . 314, 315, 325
\qMark@HookSave . . . 314, 325
\quiz@solns . . . 762, 776
\quizSolnsHeadnToc . . . 654
\QzVer . . . *29*, 740, 749

R

\r . . . 171, 172, 191–197, 219–222, 224–227, 286, 298, 299, 550–555, 593, 600, 612, 825, 830
\raggedleft . . . 255, 258
\raisebox . . . 302, 316
\rasSolns . . . *8*, 1005
\re@dOK . . . 1119, 1124, 1127, 1137
\read . . . 1123
\repeat . . . 1137
\RequirePackage . . . 2, 26, 37, 38
\ReturnTo . . . 685
\rheader . . . 94
\rhPgNumsOnly . . . *6*, 105
\rlap . . . 164, 1077, 1081
\rmSTAR . . . 135, 138
\roman . . . 681

S

\s . . . 221, 430
\sadMultQuizzes . . . 1004
\sadQuizzes . . . *30*, 794, 1004, 1005
\section . . . 654
\setArrayLength . . . 790
\setClassArray . . . 788
\setcounter . . . 43
\setfilesuffix . . . 791
\setInitMag . . . *4*, 44
\setLink . . . 644
\setsolnspace . . . 647
\ShrtPtsFld . . . *14*, 276, 308
\ShrtPtsFld@Fmt . . . 274, 278
\ShrtPtsFldFmt . . . *14*, 273, 275
\ShrtPtsFldFmt@i . . . 273, 274
\smash . . . 315
\sqslsecrunhead . . . 760, 774
\sqslsectitle . . . 656, 660
\st@scndclmn . . . 637, 638

\st@scndclmnSAVE . . . 637, 638
\st@thrdclmn . . . 644
\stepcounter . . . 316, 757
\sthline . . . 635
\stmarkupbox . . . 642
\stmarkupHeight . . . 640
\stmarkupTextSize . . . 641
\stmarkupWidth . . . 639
\strut . . . 636
\stuASOn . . . 591, 592, 595, 611
\stuAutoClose . . . 601, 604
\stuAutoCloseOff . . . 603
\stuAutoCloseOn . . . *23*, 599, 602
\stuAutoCloseScript . . . 599, 603, 613
\stuAutoSave . . . 555, 594, 597, 612
\stuAutoSaveOff . . . *23*, 595
\stuAutoSaveOn . . . *23*, 592, 598
\stuAutoSaveScript . . . 593, 596, 613
\studentGrade . . . *12*, 235, 260
\studentReport . . . *12*, 232, 257
\stuSaveBtn . . . *24*, 307, 609
\stuSaveBtn@CA . . . 605, 610
\stuSaveBtn@TU . . . 607, 610
\stuSaveBtnCA . . . *24*, 605, 606
\stuSaveBtnTU . . . *24*, 607, 608
\sumrytbllinkHook . . . 643, 645
\sumryTblP . . . 635
\sumryTblQ . . . 635
\sumryTblR . . . 635

T

\t . . . 193–196, 222, 225, 226, 549–555, 593, 599, 830
\t@hQzHeaderR . . . 84, 89, 94
testmode (option) . . . *3*
\textbf . . . 239, 241, 243, 245, 649
\textColor . . . 236, 630
\textField . . . 159, 161, 164, 169, 179, 190, 233, 235, 270, 284, 295, 316, 629, 653, 1067, 1069
\textsf . . . 1060, 1086, 1100
\textSize . . . 236, 630
\th@allowfreezefalse . . . 19
\th@allowfreezetrue . . . 18
\th@bMrkQz . . . 159, 163
\th@DeclareQuiz . . . 617, 618, 759
\th@distrQuizzes . . . 130, 133
\th@distrQuizzes@i . . . 134, 139
\th@dquoteCat . . . 3, 1147
\th@exstar . . . 706, 707, 709, 712, 725
\th@fullnameFmt . . . 167, 173
\th@fullnamePresets . . . 166, 170
\th@HeaderOffset . . . 78, 651, 652, 655
\th@initmag . . . 48, 54, 57
\th@InputCFG . . . 1141, 1146
\th@leftShiftHdr . . . 650, 651
\th@loadCFG . . . 5, 6, 1146
\th@namePresets . . . 157, 158, 160, 162
\th@next . . . 125, 130, 131
\th@qbCnt . . . 744, 745
\th@QHFirstName . . . 239, 252
\th@QHGrade . . . 245, 260
\th@QHLastName . . . 241, 253
\th@QHPoints . . . 243, 257
\th@QzHeaderC . . . 81, 88, 93, 99, 767, 779
\th@QzHeaderCQ . . . 79, 81, 779, 782
\th@QzHeaderCS . . . 82, 99, 102, 767, 770
\th@QzHeaderL . . . 76, 88, 92, 98, 766, 778
\th@QzHeaderLQ . . . 75, 76, 78, 778, 781
\th@QzHeaderLS . . . 78, 98, 101, 762, 766, 769
\th@QzVer . . . 739, 742
\th@setHeaders . . . 87, 91, 106
\th@star . . . 703, 704, 711
\thClassFS . . . 67, 68
\thCoverPagefalse . . . 107
\thCoverPagetrue . . . 108
\thCvrPg . . . 109, 110, 424
\theeqpointvalue . . . 300, 625
\thEnumQuizzes . . . 121, 122
\thepage . . . 85
\thequestionno . . . 323, 513, 514, 516, 517, 624, 632
\therearequizsolutionsfalse . . . 773
\theth@qzCnt . . . 754, 758
\thfullnameFmt . . . *9*, 167, 168
\thInstrFS . . . 58, 59
\thIsCP . . . 109, 110, 423
\thisQuiz . . . 756
\thisQuizOrig . . . 756, 758
\thordinaryfalse . . . 10
\thordinarytrue . . . 11
\thOrdQz . . . 25, 536
\ThorsAlert@Title . . . 547, 561
\ThorsAlertTitle . . . 561, 562
\thPageOne . . . 43, 248, 249, 755
\thQHFirstName . . . *12*, 239, 240
\thQHGrade . . . *12*, 245, 246
\thQHLastName . . . *12*, 241, 242
\thQHPoints . . . *12*, 243, 244
\thQuizHeader . . . *13*, 247
\thQuizHeaderLayout . . . *13*, 248, 249, 251
\thQuizName . . . *24*, 80, 83, 617, 622
\thQuizTrailer . . . *15*, 302

\thQzHeaderCQ . . . *6*, 79, 80, 105
\thQzHeaderCS . . . *6*, 82, 83, 105
\thQzHeaderL . . . *5*, 75, 77, 105
\thQzHeaderR . . . *6*, 84, 85
\thQzName . . . *24*, 621, 622
\thqzname . . . 621
\thQzSolnMrkr . . . *26*, 653, 656
\thtestmodefalse . . . 7, 9
\thtestmodetrue . . . 8, 11
\thUseNameToCustomize . . . 112, 113, 449, 450
\TooMuchCreditMsg . . . *17*, 372, 373
\TotalsFld . . . *14*, 294, 310
\TotalsFld@Fmt . . . 291, 296
\TotalsFldFmt . . . *14*, 290, 292
\TotalsFldFmt@i . . . 290, 291
\tstForSTAR . . . 136, 137, 141
\tstForSTAR@i . . . 136
\TU . . . 181, 201, 209, 218, 322, 610, 1064, 1067, 1069

U

\u . . . *27*, *30*, 785, 793
usebatch (option) . . . *3*
useclass (option) . . . *3*
\useclassOptfalse . . . 13
\useclassOpttrue . . . 15
\usedAdbFuncs . . . 27, 32
\useNameToCustomize . . . *7*, 112
\usepackage . . . 1009, 1010, 1094

V

\V . . . 271, 631, 1074, 1078
\vcgBdry . . . 252, 253, 257, 260, 308, 309, 1068
\verbatim@out . . . 747, 748
\verbatimwrite . . . 750
\vfil . . . 1097, 1104

W

\web@latextoc . . . 659
\widthof . . . 257, 649
\write . . . 748
\wrtQzInfo . . . 620, 623

X

\x . . . 142, 717, 718, 720, 721, 723, 724, 758, 759

15 Change History

v1.1 (2019/06/30)
General: Added flattening . . . 20, 21

v1.1.1 (2019/06/30)
General: Corrected a bug in `\essayQKey` that caused a miscalculation. . . . 17

v1.1.2 (2019/07/02)
General: Added try/catch in `\AAformat` to avoid exceptions thrown in the case of distiller; added pdf space in the `\TotalsFld` JS to avoid unexpected wraps in the case of dvips/distiller worflow. . . . 15

v1.1.3 (2019/07/03)
General: Make `\LngPtsFld` a calculation field . . 14
Remove lines not needed since `\LngPtsFld` became a calculation field . . . 17

v1.1.4 (2019/07/04)
General: Add a SaveAs menu item to end of the quiz . . . 23

v1.1.5 (2019/07/06)
General: If document is dirty, do not save . . . 23

v1.1.6 (2019/07/06)
General: Added end quiz warning . . . 22
Added save as requiring acrobat . . . 20, 21
Check on name fields . . . 21

v1.1.8 (2019/07/08)
General: make studentenReport initially hidden 12

v1.1.9 (2019/07/15)
General: Added optional argument `\mkClFlsSpcls` . . . 30

v1.2 (2019/07/16)
General: Added `\freezeOrSave` . . . 12
Added batch support files . . . 36

v1.2.1 (2019/07/19)
General: Inserted test for `oRecordOfQuizData` before saving, this avoids the message that appears in the console that 'f is null' . . . 20

v1.3 (2019/07/20)
General: Added multiple quizzes in one source . 29

v1.3.1 (2019/07/20)
General: container: Push now closes the container . . . 36

v1.3.5 (2019/08/06)
General: Use `\usedAdbFuncs` to detect `usealtadobe` option . . . 4

v1.3.6 (2019/08/07)
General: freezeQuizMU: Cannot freeze unless

grade given and added suffix "-g" to document name on saving. 7
v1.3.7 (2019/08/11)
General: fixed typo in loading cfg 41
v1.4 (2019/08/11)
General: Begin major change to this package, leaving v1.3.8 as our best working version prior to this update. 3
Changes to \InputBodyQuiz to support solution sets 29
v1.4.1 (2019/08/16)
General: Added cFS to aebDocSaveAs 21
Implement the concept of a cover page. 6
v1.4.10 (2019/09/11)
General: Fixed a bug in calculation when there are no essay questions 14
v1.4.12 (2019/09/11)
General: Correction a problem with \MarkWarningMsg in the dvips/distiller workflow 18
v1.4.13 (2019/09/12)
General: Added \flattenOn and \flattenOff . . 18
v1.4.14 (2019/09/13)
General: Inserted \mkClFlsSpcls as optional argument of docassembly 8
v1.4.17 (2019/10/03)
General: Defined _workingFolderC to avoid redefinition of _workingFolder 35
Rewrote \LngPtsFldFmt to produce pdf spaces 14
Rewrote \ShrtPtsFldFmt to produce pdf spaces 14
Rewrote \TotalsFldFmt to produce pdf spaces 14
Saved path to cover page 35
v1.4.2 (2019/08/22)
General: Insert cover page in \sadQuizzes 35
v1.4.3 (2019/08/22)
General: let \Hy@EveryPageAnchor to \relax . . 13
v1.4.5 (2019/08/25)
General: Added command to remove shift for solns headers 26
v1.4.6 (2019/08/26)
General: Combined \sadMultQuizzes with \sadQuizzes 29
Renamed \sadMultQuizzes to \sadQuizzes, removed old \sadQuizzes. 30
v1.4.7 (2019/08/27)
General: Added format code from \qz@IDTxtField 16
Added marker field attached to name field 9
Added version support for quiz bodies 29
v1.4.8 (2019/09/05)
General: Added \distrQuizzes 7
Added \enumQuizzes 7
Added \useNameToCustomize 7
Append solution pages if there is one 19
v1.5.1 (2019/10/23)
General: Allow web.cfg to be imported in the preamble. 27
v1.5.2 (2019/11/21)
General: Fixed a bug with exception thrown . . . 37
Put JS for Clear Btn in defineJS 37
v1.5.4 (2019/11/30)
General: Reworked \distrQuizzes to account for second star option of \classMember 7