The eq-pin2corr Package

D. P. Story
Email: dpstory@uakron.edu

processed February 20, 2021

Contents

1 ⟨*package⟩

1 Introduction

This package is an add-on to the quiz environment of the exerquiz package. It uses the eq-save package. To correct a quiz, the document consumer must press the Correct button of a quiz and successfully enter the correct PIN number.

Purpose. This package adds PIN security to a quiz created by the quiz environment. This package is designed for the educational sector, for instructors who use the quizzes of exerquiz to assess their students understanding of the course material.

PDF Viewers. Discussion of PDF viewers for document author and document consumer.

Instructor Any PDF viewer may be used as a PDF previewer, sumatraPDF, for instance, can be used, but it has not functionality. To test the newly created document to see if it is functioning correctly, must use Adobe Reader DC or Acrobat DC.[1]

[1] We use DC here to refer to actually any Adobe AA/AR application. Earlier versions of these applications will work.

Document consumers (students) The exerquiz and eq-pin2tocorr extensively use JavaScript to perform many background tasks. For the student to have any success in this workflow, he/she must use Adobe Reader.

Workflow. The package is designed for the following workflow:

1. The instructor creates the quiz using the exerquiz and eq-pin2corr packages.
2. The instructor delivers the "PDF quiz" to each student. (System drive or email)
3. The student takes the quiz. The student can press the Correct but, unless he/she knows the PIN, the quiz is not marked up.
4. The student saves the PDF quiz in Adobe Reader DC.
5. The student returns the PDF to the instructor. (System drive or email)
6. The instructor presses the Correct button to mark up the quiz and record the grade of the student. The instructor saves the quiz.
7. The instructor returns the PDF, at some point, to the student.
8. Both instructor and student happily go on with their lives.

2 Options and package requirements

```
\newif\ifPINshowScore \PINshowScorefalse
\DeclareOption{showscore}{\PINshowScoretrue}
\DeclareOption{!showscore}{\PINshowScorefalse}
\ProcessOptions\relax
\RequirePackage{exerquiz}[2021/02/17]
\RequirePackage{eq-save}[2021/02/17]
```

3 Package commands

`\showScoreOn` `\showScoreOff`

Implement local versions of the package options `showscore` and `!showscore`, these are `\showScoreOn` and `\showScoreOff`.

```
\def\showScoreOn{\PINshowScoretrue}
\def\showScoreOff{\PINshowScorefalse}
```

`\SaveAndSendMsg`

Define a message that appears on the console when the PIN entered is not correct.

```
\flJSStr[noquotes]{\SaveAndSendMsg}{Success! %
Now save and send to the instructor}
```

`\postSubmitQuiz`

Make changes to the End Quiz control and to the Correct control. Begin by modifying the `\postSubmitQuiz` command, which is a hook within the executing code of the End Quiz control.

```
\begin{defineJS*}[\makeesc\@\makecmt\%]{\postSubmitQuiz}
// Begin post submit quiz code%
```

```
@ifPINSecurity%
@ifPINshowScore@else
    var f = this.getField("ScoreField.@oField");
    if ( f!=null ) {
      f.textSize=0;
      f.value = "@SaveAndSendMsg";
    } else {
      var f = this.getField("PointsField.@oField");
      if (f!=null) {
        f.textSize=0;
        f.value = "Success! Now save and send to instructor";
      }
    }@fi@fi
    oRecordOfQuizData["ScoreData.@oField"]=%
[1*Score,1*NQuestions,1*ptScore,1*NPointTotal];
    oRecordOfQuizData["RightWrong.@oField"]=%
eval(RightWrong.toSource());
    oRecordOfQuizData["ProbDist.@oField"]=%
eval(ProbDist.toSource());
    cntCorrectResponses();
\end{defineJS*}
```

\eQzBtnActns The command name for the action of the End Quiz control is `\eQzBtnActns`. We save this and pre-pend a single code line, as needed.

```
\let\eQzBtnActnsSave\eQzBtnActns
\def\eQzBtnActns{\ifPINshowScore\else
  var bDisplaySilent=true;\r\fi
  \eQzBtnActnsSave
}
```

The command name for the action of the Correct control is `\CorrBtnActionsJS` we save this and later modify it.

```
\let\CorrBtnActionsJSSave\CorrBtnActionsJS
```

\usePINCorrBtn \restoreCorrBtn We can turn on and off the PIN feature by expanding `\usePINCorrBtn` and `\restoreCorrBtn`.

```
\newif\ifPINSecurity \PINSecurityfalse
\def\usePINCorrBtn{\PINSecuritytrue
  \let\CorrBtnActionsJS\CorrBtnActionsPwdJS}
\def\restoreCorrBtn{\PINshowScoretrue\PINSecurityfalse
  \let\CorrBtnActionsJS\CorrBtnActionsJSSave}
```

The instructor can tediously press the Correction button, or place an entry,

```
var _PinCode1 = "O2JRVZdRgYgCA-Rtje8VkD";
```

in the file `config.js`. If such a variable exists and its value matches the PIN hash string, the instructor clicks the Correction button to get the quiz markup. Conceivably, the instructor might want different PIN names and string hash values. The `\classPINVar` is a convenient way of declaring the PIN variable name; eg, if `\classPINVar{_PinCode1}` is declared prior to the `quiz` environment, the instructor need not manually enter the PIN.

\classPINVar

```
\def\classPINVar#1{\def\PINclassPV{#1}}
\let\PINclassPV\@empty
```

The modified action for the Correct button. we save this and later modify it.

```
\begin{defineJS*}[\makeesc\!\makecmt\%]{\CorrBtnActionsPwdJS}
!ifx!PINclassPV!@empty%
var userPIN = "";!else%
var userPIN = "!PINclassPV";!fi
if (userPIN == "" ) userPIN = undefined;
try {
  if ( typeof eval(userPIN) == "undefined") userPIN = undefined;
} catch(e) { userPIN = undefined; }
if (typeof userPIN == "undefined") {
  var resp=app.response({
    cQuestion: "Enter the PIN number",
    cTitle: "View Answers",
    bPassword: true
  });
  var _resp=Collab.hashString(resp);
  var _bQzResults = ( _resp ==_PinCode );
} else var _bQzResults = ( eval(userPIN) ==_PinCode );
if (_bQzResults) {
  RightWrong=eval("RightWrong.!currQuiz");
  ProbDist=eval("ProbDist.!currQuiz");
  correctQuiz("!currQuiz",3);
  DisplayQuizResults("!currQuiz",3,3);
  if (typeof correctSumryTbl == "function")
    correctSumryTbl("!currQuiz",3);
} else {
  console.println("Something went wrong, \\
you entered an incorrect PIN Id, \\
or the class PIN Id (\\\\classPINVar) was incorrect or undefined");
  console.show();
}
\end{defineJS*}
```

`\declPINId{`⟨*PIN-Id*⟩`}{`⟨*hash-str*⟩`}` Set the basic parameters of this PIN security scheme: the pin-id and its corresponding hash-string.

```
\def\declPINId#1#2{\def\numPINId{#1}\def\hashPINId{#2}}
\@onlypreamble\declPINId
\declPINId{5243}{02JRVZdRgYgCA-Rtje8VkD} % PIN Id, hash-str
\begin{insDLJS}{pin}{Pin Code}
var _PinCode = "\hashPINId";
\end{insDLJS}
```

⟨/package⟩

4 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.

5 Change History

v1.0 (2021/02/20) package for first time 1
General: Completed documentation, publish