

**AcroTeX Software Development Team**

# **The AcroTeX eDucation Bundle (AeB)**

**D. P. Story**

Copyright © 1999-2012  
Prepared: May 15, 2012

[dpstory@acrotex.net](mailto:dpstory@acrotex.net)  
<http://www.acrotex.net>

## Table of Contents

<b>Preface</b>	<b>8</b>
<b>1 Getting Started</b>	<b>9</b>
1.1 Installing the Distribution . . . . .	9
1.2 Installing aeb.js . . . . .	10
1.3 Language Localizations . . . . .	11
1.4 Sample Files . . . . .	11
1.5 Package Requirements . . . . .	11
1.6 L <sup>A</sup> T <sub>E</sub> Xing Your First File . . . . .	12
• For pdftex and dvi <sub>pdfm</sub> Users . . . . .	12
• For Distiller Users . . . . .	12
<b>The Web Package</b>	<b>14</b>
<b>2 Introduction</b>	<b>14</b>
2.1 Overview . . . . .	14
2.2 Package Requirements . . . . .	14
<b>3 Basic Package Options</b>	<b>15</b>
3.1 Setting the Driver Option . . . . .	15
3.2 The tight Option . . . . .	16
3.3 The usesf Option . . . . .	16
3.4 The draft Option . . . . .	16
3.5 The nobullets Option . . . . .	16
3.6 The unicode Option . . . . .	16
3.7 The useui Option . . . . .	16
<b>4 Setting Screen Size</b>	<b>17</b>
4.1 Custom Design . . . . .	17
4.2 Screen Design Options . . . . .	17
4.3 \setScreenSizeFromGraphic . . . . .	18
4.4 Using \addtoWebHeight and \addtoWebWidth . . . . .	18
4.5 Using a panel option . . . . .	19
<b>5 Hyperref Options</b>	<b>19</b>
<b>6 Running Headers and Footers</b>	<b>19</b>
<b>7 The Title Page and TOC</b>	<b>20</b>
7.1 Basic Information Commands . . . . .	21
7.2 Title Page Structure . . . . .	22
7.3 Redefining \maketitle . . . . .	24
7.4 The Title Page Directory . . . . .	24

7.5	The TOC for Web . . . . .	26
7.6	The <code>usedirectory</code> Option . . . . .	27
7.7	The <code>latex toc</code> Option . . . . .	27
7.8	The <code>centertitlepage</code> Option . . . . .	27
7.9	The <code>\makeinlinetitle</code> Command . . . . .	28
<b>8</b>	<b>Template Options</b>	<b>28</b>
8.1	The <code>usetemplates</code> Option . . . . .	28
8.2	The <code>leftpanel</code> and <code>rightpanel</code> Options . . . . .	28
8.3	The <code>leftpanelprt</code> and <code>rightpanelprt</code> Options . . . . .	29
<b>9</b>	<b>Navigation Aids</b>	<b>29</b>
9.1	A Navigation Bar . . . . .	29
9.2	<code>\newNaviIcon</code> . . . . .	29
9.3	Direction Icons . . . . .	30
9.4	<code>\panelNaviGroup</code> . . . . .	31
<b>10</b>	<b>The Language Options</b>	<b>31</b>
<b>11</b>	<b>Paper Related Options and Commands</b>	<b>31</b>
11.1	The <code>forpaper</code> Option . . . . .	31
11.2	The <code>forcolorpaper</code> Option . . . . .	32
11.3	The <code>latex layout</code> Option . . . . .	32
11.4	The <code>uselatexparts</code> and <code>uselatexchapters</code> Options . . . . .	32
11.5	The <code>\useFullWidthForPaper</code> Command . . . . .	32
<b>12</b>	<b>Formatting for screen and paper</b>	<b>32</b>
<b>13</b>	<b>Template Building and Management</b>	<b>33</b>
13.1	Template options . . . . .	34
13.2	Text Screen Template . . . . .	34
13.3	Panel Screen Template . . . . .	35
13.4	Template Management . . . . .	36
<b>14</b>	<b>The <code>pro</code> Option</b>	<b>37</b>
14.1	<code>\DeclareDocInfo</code> . . . . .	37
14.2	<code>\DeclarePageLayout</code> . . . . .	40
14.3	The Title Page: <code>\maketitle</code> . . . . .	43
	• The Title Page Structure . . . . .	43
	• Greater Control of the Top Title Page . . . . .	44
	• Greater Control over Title Page Trailer . . . . .	47
14.4	The Table of Contents . . . . .	49
14.5	Selecting Colors . . . . .	49
14.6	Section Headings . . . . .	50

<b>15 On Parts, Chapters, and Final Dots</b>	<b>53</b>
15.1 The <code>\part</code> command . . . . .	53
15.2 The <code>\chapter</code> command . . . . .	54
15.3 The <code>\noFinalDot</code> Command . . . . .	54
<b>The Exerquiz Package</b>	<b>55</b>
<b>16 Exerquiz and Acrobat JavaScript</b>	<b>55</b>
<b>17 Package Requirements</b>	<b>55</b>
<b>18 Basic Package Options</b>	<b>56</b>
18.1 The <code>pdftex</code> Option . . . . .	56
18.2 The <code>dvipdfm</code> Option . . . . .	57
18.3 The Language Option . . . . .	57
18.4 The <code>forpaper</code> and <code>forcolorpaper</code> Options . . . . .	57
18.5 The <code>preview</code> Option . . . . .	58
18.6 The <code>nodljs</code> Option . . . . .	58
18.7 The <code>exercisonly</code> Option . . . . .	58
18.8 The <code>debug</code> Option . . . . .	58
18.9 The <code>allowrandomize</code> Option . . . . .	59
18.10 The <code>unicode</code> Option . . . . .	59
18.11 The <code>useui</code> Option . . . . .	59
18.12 The <code>usesumrytbls</code> Option . . . . .	59
<b>19 The exercise Environment</b>	<b>59</b>
19.1 Basic Usage . . . . .	60
• Exercises with Parts: The <code>exercise*</code> Environment . . . . .	62
19.2 Options of the <code>exercise</code> Environment . . . . .	63
• Leaving Vertical Space instead of a Solution . . . . .	63
• Hiding some Solutions . . . . .	64
• The <code>nohiddensolutions</code> Option . . . . .	65
• The <code>noHiddensolutions</code> Option . . . . .	65
• The <code>exercise environment Counter</code> . . . . .	65
• The <code>nosolutions</code> Option . . . . .	65
• The <code>solutionsafter</code> Option . . . . .	66
• Moving the Solution Set . . . . .	67
19.3 Redesigning the <code>exercise</code> Environment . . . . .	67
<b>20 The shortquiz Environment</b>	<b>70</b>
20.1 Basic Usage . . . . .	70
• <code>shortquiz</code> with Radio Buttons . . . . .	72
• <code>shortquiz</code> with Solutions . . . . .	73
• The <code>\bChoices/\eChoices</code> Commands . . . . .	74
• The <code>questions</code> Environment . . . . .	76

20.2	Options of the <code>shortquiz</code> Environment . . . . .	77
	• The <code>forpaper</code> Option . . . . .	77
	• The <code>solutionsafter</code> Option . . . . .	77
	• The <code>proofing</code> Option . . . . .	78
	• The <code>showgrayletters</code> Option . . . . .	78
	• Moving the Solution Set . . . . .	79
20.3	Redesigning the <code>shortquiz</code> Environment . . . . .	79
	• Changing Titles . . . . .	79
	• Modifying Form Elements . . . . .	80
20.4	<code>\titleQuiz</code> and <code>\fancyQuizHeaders</code> . . . . .	80
	• <code>\titleQuiz</code> . . . . .	81
	• <code>\titleQuiz*</code> . . . . .	82
	• Cross-Referencing . . . . .	82
	• <code>\fancyQuizHeaders</code> . . . . .	83
<b>21</b>	<b>The <code>quiz</code> Environment</b> . . . . .	<b>83</b>
21.1	Options of the <code>quiz</code> Environment . . . . .	85
	• The option <code>noquizsolutions</code> . . . . .	85
21.2	The <code>answers</code> Environment . . . . .	85
21.3	Basic Usage . . . . .	85
	• Link-Style Quiz . . . . .	86
	• Form-Style Quiz . . . . .	87
	• Using Circular Radio Buttons with MC Questions . . . . .	88
	• Overriding the ‘ <code>quiztype</code> ’ Parameter . . . . .	88
	• The <code>BeginQuiz</code> and <code>EndQuiz</code> Form Buttons . . . . .	88
	• The <code>proofing</code> Option . . . . .	89
	• The <code>showgrayletters</code> Option . . . . .	89
	• Setting the Threshold . . . . .	89
21.4	The <code>manswers</code> Environment . . . . .	90
21.5	Correcting the Quizzes with JavaScript . . . . .	90
	• The <code>nocorrections</code> Option . . . . .	92
	• Showing Partial Credit Markup . . . . .	92
21.6	Quizzes with Solutions . . . . .	93
21.7	How to Modify the <code>quiz</code> Environment . . . . .	94
	• The Quiz Titles . . . . .	94
	• The check appearance . . . . .	95
	• Change color of Correction Marks . . . . .	95
	• The ‘Correction’ Button . . . . .	96
	• The Score Field . . . . .	96
21.8	<code>\titleQuiz</code> and <code>\fancyQuizHeaders</code> . . . . .	97
	• <code>\titleQuiz</code> . . . . .	97
	• Cross-Referencing . . . . .	98
	• <code>\fancyQuizHeaders</code> . . . . .	99
21.9	Adding Points to a Quiz . . . . .	100
	• <code>\negPointsAllowed</code> . . . . .	102
21.10	Floating a Quiz . . . . .	102

<b>22 Objective Style Questions</b>	<b>103</b>
22.1 Math and Text Questions . . . . .	103
• The Mathematical Question . . . . .	103
• The Text Question . . . . .	104
22.2 The <code>oQuestion</code> Environment . . . . .	104
• <code>\RespBoxMath</code> : The Math Question . . . . .	104
• <code>\RespBoxTxt</code> : The Text Question . . . . .	107
• <code>\RespBoxTxtPC</code> : The Text Question with Partial Credit . . . . .	109
22.3 Some Enhancements . . . . .	110
• Including an Answer Key with <code>\CorrAnsButton</code> . . . . .	110
• Including a Solution . . . . .	110
• Including a Tally Box . . . . .	112
• Clearing the Fields . . . . .	112
22.4 More on Math Fill-in Questions . . . . .	112
• Comments on Authoring . . . . .	113
• Comments on User Input . . . . .	113
22.5 The <code>shortquiz</code> Environment . . . . .	114
22.6 The <code>quiz</code> Environment . . . . .	116
• The Prompt Button . . . . .	117
• Grouped Math/Text Fill-in Questions . . . . .	118
22.7 Modifying Form Elements . . . . .	118
22.8 Evaluating Equivalent Expressions . . . . .	119
<b>23 Randomizing the Multiple Choices</b>	<b>123</b>
<b>24 Creating a Quiz Summary Table</b>	<b>126</b>
24.1 Placing the Quiz Summary Table Elsewhere . . . . .	128
<b>25 Bookmarking Exercises and Quizzes</b>	<b>129</b>
25.1 For Exercises . . . . .	129
25.2 For Quizzes . . . . .	130
25.3 Final Note . . . . .	130
<b>26 Extending <code>AcroTeX</code> with <code>dljslib</code> and <code>insdljs</code></b>	<b>131</b>
26.1 Using the <code>dljslib</code> Package . . . . .	131
26.2 Using the <code>insdljs</code> Package . . . . .	131
<b>27 Submitting a quiz to a Web Server</b>	<b>131</b>
27.1 Technical Info for “Do It Yourself” . . . . .	132
• Redefining “End Quiz” . . . . .	132
• Gathering ID Information with <code>\textField</code> . . . . .	132
• Gathering Quiz Specific Information with <code>\eqSubmit</code> . . . . .	133
• Some Variables to Submit . . . . .	134
27.2 Features <i>apropos</i> to Submitting . . . . .	134
• Assigning Points . . . . .	134
• <code>\NoPeeking</code> . . . . .	134

<b>28 Functions and Syntax supported by Exerquiz</b>	<b>135</b>
28.1 Functions supported by Exerquiz . . . . .	135
28.2 Syntax supported by Exerquiz . . . . .	136
<b>The eq2db Package</b>	<b>138</b>
<b>AcroTeX eForm Support</b>	<b>139</b>
<b>The AeB JavaScript Library</b>	<b>140</b>
<b>29 Usage and Options</b>	<b>140</b>
29.1 Parse Input Extensions . . . . .	142
• ImplMulti . . . . .	142
29.2 Response Functions . . . . .	142
• equations . . . . .	143
• vectors . . . . .	143
• setSupport . . . . .	144
• unordered . . . . .	145
• factors . . . . .	145
• point . . . . .	145
• intervals . . . . .	146
• complex . . . . .	147
29.3 Compare Functions . . . . .	148
• indefIntegral . . . . .	148
• satisfyEq . . . . .	148
29.4 Filter User's Responses . . . . .	150
• nodec . . . . .	150
• noBinFac . . . . .	150
• limitArith . . . . .	151
29.5 Extending Numerical Functions . . . . .	153
• combinatorics . . . . .	153
<b>30 List of Options</b>	<b>154</b>
<b>Solutions to Exercises</b>	<b>158</b>
<b>Solutions to Quizzes</b>	<b>160</b>
<b>References</b>	<b>162</b>

## Preface

The **AcroTeX eDucation Bundle** (abbreviated **AeB** and read “AcroTeX Education Bundle”) is a collection of  $\LaTeX$  macro files (briefly described below) along with various support and sample files. The theme of **AeB** is *ePublication* in the *education* sector using  $\LaTeX$  as the authoring application and Adobe’s Portable Document Format (PDF) as the file format of the output document. The **AcroTeX Bundle** should be useful to educators who want to post interactive materials for their students on the WWW.

Currently, there are seven components to the bundle:

1. The **web package** is used to create an attractive, easy-on-the-eye page layout suitable for the WWW (or classroom/conference presentations). There is support for background graphics; a Web document can be re-purposed for paper.
  2. The **exerquiz package** makes it very easy to create interactive exercises and quizzes. There are numerous options, solutions at the end of the document, solutions following the question, no solutions; quizzes are self marking, the student can get immediate feedback, or get a final assessment after the student finishes the quiz.
  3. The **eforms** package provides support for PDF form fields, and links with arbitrary actions.
  4. The **insdljs** package allows for the automatic insertion of document level JavaScript. Document authors can use **insdljs** to customize the processing of the **exerquiz** quizzes. See the documentation that accompanies the package (**insdljs.dtx**) and see also the sample file **jqzspec.tex** for an extensive example of how to modify the **exerquiz** macros. The **insdljs** package also has an **execJS** environment which can be used to create executable and “discardable” JavaScript; see the DTX file for details. Documentation for the **insdljs** package appears in the **eforms** reference manual (**eformman.pdf**).
  5. The **taborder** package supports the creation of the order of tabbing between Acrobat annotations (fields, links, and comments). See the part **Setting the Tab Order** in **eforms** reference manual (**eformman.pdf**) for more details.
  6. The **dljslib** package is used as a library of JavaScript functions. Some types of question require special processing. A JavaScript function written to process a particular function can be stored in the library, then retrieved as needed. See the documentation contained in the file **dljslib.dtx**.
  7. The **eqexam** Package is a stand alone  $\LaTeX$  package for creating exams, quizzes, homework assignments. There are several options to re-purpose the source document to produce a document with no solutions, solutions at end, solutions after the questions, and solutions only. It can also be used to create surveys, questionnaires, teacher evaluations, etc. The package has an **email** mode and comes with a server-side script that is used to send the data entered into the form fields to the instructor. See my **TeX/LaTeX Survey** for a demo of this feature.
- The **web**, **exerquiz**, **eforms**, and **insdljs** packages are stand alone packages, though they were designed to work together. The **dljslib** package is a companion package to **insdljs**. The **eqexam** package is stand along, but uses some code from the **exerquiz** package; **exerquiz** need not be installed on your system to use **eqexam**, however.

Here is an important point that should be emphasized early in this manual.  $\text{AcroT}_{\text{E}}\text{X}$  only supports three ways of producing a PDF file: (1) the [Adobe Acrobat Distiller](#) (version 5.0 or higher *required*); (2) `pdftex`; or (3) `dvipdfm`. In the case of (1), you probably use `dvips` to create a postscript file before distilling. Some users have tried to use [GhostScript](#) to produce a PDF from  $\text{AcroT}_{\text{E}}\text{X}$ ; this will not work! (You will get the PDF file, but not much functionality.)

Please contact me at [dpstory@acrotex.net](mailto:dpstory@acrotex.net) should you encounter any problems, or have suggestions to make.

- ▶ See '[Getting Started](#)' on page 9 for instructions on how to get up and running.

## 1. Getting Started

The distribution for the  $\text{AcroT}_{\text{E}}\text{X}$  Bundle comes in a single ZIP file

- `acrotex_pack.zip`: This ZIP file contains program files (`web`, `exerquiz`, `eforms`, `insdljs`, `dljslib`, `taborder`, and `aeb.js`), this manual, and the eForms manual.

### 1.1. Installing the Distribution

The following are the instructions for installing the [AeB](#).

- 1. Placement.** The ZIP file installs in a folder called `acrotex`, so place the ZIP file in a directory in the search path of your  $\text{E}_{\text{T}}\text{X}$  system where you want the `acrotex` folder to reside. If you already have an `acrotex` folder, place the ZIP files so that they unzip into this `acrotex` folder, *unless you are using  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  2.8*.

**$\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  2.8 Users.**  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  2.8 provides problems for  $\text{AcroT}_{\text{E}}\text{X}$  installations in that it is more particular about where you install packages by hand.  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  2.8 requires that you install the distribution in a local root TDS tree. Review the  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  help page on this topic

<http://docs.miktex.org/manual/localadditions.html>

Create a new  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  root folder, in my case, I created

`C:\Users\Public\Documents\My TeX Files`

Within the `My TeX Files` folder (you can name this folder anyway you wish), create a `tex` folder, within the `tex` folder, create a `latex` folder. Now copy the ZIP file to the `latex` folder, for example, copy them into

`C:\Users\Public\Documents\My TeX Files\tex\latex`

Finally, you need to register your new root folder with  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$ . Open the  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  Options dialog box (accessible through the WinEdt toolbar, or through

Start > All Programs >  $\text{MiK}_{\text{T}}\text{E}_{\text{X}}$  2.8 > Maintenance > Settings

Within this dialog box, select the **Roots** tab, then press **Add**, and browse for your newly create `My TeX Files`. When finished, the dialog should look like my own dialog box, as shown in [Figure 1](#), page 10.

- 2. Unzipping the Distribution.** Once you have found the location to unzip the ZIP file, unzip it!

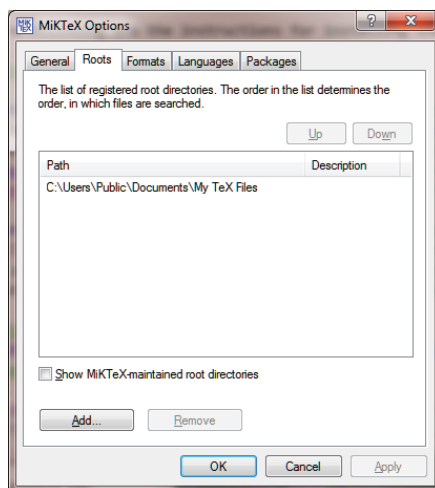


Figure 1: Roots tab of MiKTeX Options

- 3. Unpacking the distribution.** The whole distribution can be unpacked by latexing `acrotex.ins`. (The other `*.ins` files are the installation files for the individual packages, `acrotex.ins` is the combined installation file.)

MiKTeX users should not forget to refresh the file name database.

### 1.2. Installing `aeb.js`

The JavaScript file `aeb.js` is only needed if you use **Acrobat Pro 8.1** or later. Increased security in that version has made it necessary to install a folder JavaScript file. `aeb.js` raises the “trust” of certain JS methods used to import document JavaScript into the document just after installation. The installation of this file is not needed if you use `pdftex` or `dvipdfm`, or use a version of **Acrobat** prior to 8.1.

**Acrobat Pro 8.1 or later.** Start **Acrobat Pro 8.1**<sup>1</sup> or later, and open the console window `Advanced > JavaScript > Debugger (Ctrl+J)`. Copy and paste the following code into the window.

```
app.getPath("user", "javascript");
```

Now, with the mouse cursor on the line containing this script, press the `Ctrl+Enter` key. This will execute this JavaScript. This JavaScript method returns the path to where `aeb.js` should be placed. For example, on my system, the return string is

```
/C:/Documents and Settings/story/  
Application Data/Adobe/Acrobat/8.0/JavaScripts
```

Follow the path to this folder. If the JavaScripts folder does not exist, create it. Finally, copy `aeb.js` into this folder. The file is read each time Acrobat starts.

<sup>1</sup>The file will cause no harm if you install it in an earlier version of Acrobat, as early as version 7.

**Acrobat Pro 10.1.1.** For this version of **Acrobat**, things have tightened up even more. The user JavaScripts folder has moved to


```
%AppData%\Adobe\Acrobat\Privileged\10.0\JavaScripts
```


where `%AppData%` is an environment variable defined by Acrobat. For details of how to install the folder JavaScripts in the new location, see my blog article [Acrobat Security Changes in 10.1.1 and AcroTeX](#).

### 1.3. Language Localizations

Should you wish to customize **AeB** to a language other than the ones already supported, open the file `template.def` and add in your language customizations.

### 1.4. Sample Files

Test, example, and demo files have been moved to the [Example Files for AeB](#) web page. References seen in this manual reside on this page; each sample file is a PDF with the source file attached to the PDF. Files on the [Example Files for AeB](#) web page are referenced with the icon  in the margin.

There is another more recent collection of examples on [AcroTeX Blog](#), these will be referenced in the margin using the icon .

### 1.5. Package Requirements

If you use the Acrobat Distiller, as I do, to create a PDF document, the [AcroTeX Bundle](#) now requires the use of version 5.0 or later. I've given up on trying to support prior version of Acrobat.

In terms of  $\LaTeX$ , the following is a listing of package requirements:

1. The Web Package
  - `color`: `color` is distributed with  $\LaTeX$ , but Web will use `xcolor`, if available, unless the `noxcolor` global option is used.
  - `amssymb`: standard with  $\mathcal{A}\mathcal{M}\mathcal{S}$ - $\LaTeX$
  - `hyperref`: available from CTAN, get newer version
  - `eso-pic` and `everyshi`: available from CTAN
2. The Exerquiz Package
  - `color`: distributed with  $\LaTeX$
  - `verbatim`: distributed with  $\LaTeX$
  - `hyperref`: available from CTAN, get newer version
  - `insdljs`: distributed with AcroTeX
3. The `insdljs` Package
  - `hyperref`: available from CTAN, get newer version

- `verbatim`: distributed with  $\LaTeX$
- `everyshi`: available from CTAN


#### 4. The `dljslib` Package

- `insdljs`: distributed with  $\AcroTeX$

### 1.6. $\LaTeX$ ing Your First File

The functionality of the `shortquiz` and `quiz` environments depends on JavaScript code that is placed at the “document level”, to use Adobe’s terminology. The applications `pdftex` and `dvipdfm` offer direct support for writing to this document level. For those who use the [Adobe Distiller](#), things aren’t quite so easy.

In this section, we describe how to insert document level JavaScripts into a PDF file, prepared from a  $\LaTeX$  source that uses the `exerquiz` package. Even though the handling and insertion of document level JavaScript is done with the package `insdljs`, a little care must be taken, at least in the Distiller case, when building your PDF document.

 Open `webeqtst.tex` in your favorite text editor. The top lines read:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[tight,designi]{web}
\usepackage{exerquiz}
```

#### • For `pdftex` and `dvipdfm` Users

Edit the third line by inserting your driver; the choices are `pdftex` and `dvipdfm`. For example, if you use `dvipdfm`, the lines should read:

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvipdfm,tight,designi]{web}
\usepackage{exerquiz}
```

For the `pdftex` application, you simply call `pdf $\flat$ atex`, and you have your nice PDF file, ready for review. The insertion of the document level JavaScript is automatic.

For `dvipdfm`, you  $\LaTeX$  the document, then hit it with `dvipdfm`, and your ready to review your PDF file.

#### • For Distiller Users

If you use the distiller, as I do, the sophisticated features of  $\AcroTeX$  Bundle require Acrobat 5.0 or higher.

Edit the optional parameters of the `web` package by inserting your driver; the choices are `dvips` and `dvipsone`.<sup>2</sup> For example, if you use `dvips`, the lines should read:

<sup>2</sup>The choice of driver can be placed in the `web.cfg` configuration file, in which case, you need not specify the driver. See ‘[Setting the Driver Option](#)’ on page 15.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage[dvips,tight,designi]{web}
\usepackage{exerquiz}
```

When you latex the source file, you create a DVI file and one or more FDF files.<sup>3</sup> You then convert your .dvi to .ps using either dvips or dvipsone, and distill.

**Important:** When you distill, *save the .pdf back to the same folder* in which your source file (.tex) resides as this is where the .fdf files reside too. Insertion of document level JavaScripts automatically takes place when you open your newly distilled document in the Acrobat application. (It is actually Acrobat that imports the scripts, not the Distiller.)

- When your document is opened in Acrobat for the first time, the JavaScript contained in the .fdf files (for example, exerquiz.fdf) is imported into the document and is stored at the document level.

► **Important:** *Save your document.* When you save, the JavaScripts you just imported are also saved with the document. At this point you can move your PDF to another folder, or to the web. The document does not need the .fdf files any more.

There are several ways the Distiller workflow can be accomplished:

For distiller users, the **AeB Pro package** has many exciting features, functionality requires the document author use Acrobat 7.0 or higher.

---

<sup>3</sup>The FDF files (for example, exerquiz.fdf) contain the document level JavaScript that needs to be imported into your document. They are imported when the newly created document is opened in Acrobat for the first time.

## The Web Package

### 2. Introduction

The purpose of the `web` package is to create a page layout for documents meant for screen presentation, whether over the WWW or classroom/conference presentations, in PDF. Such documents are *not* (necessarily) *intended to be printed*; consequently, the page layout is, in some sense, optimized for screen viewing.

#### 2.1. Overview

The `web` package redefines `\maketitle` and `\tableofcontents` in a more web friendly way; it colors the section headings, and inserts `\bullets` (•) at the `\subsubsection` level. This, to my eyes, is very attractive. Additionally, certain navigation devices (a navigation bar and some direction icons) are included in the package.

There are options for a small collection of drivers: `dvipsone`, `dvips`, `dvipdfm` and `pdftex`. The language option redefines certain language dependent elements of the package to other languages. Currently, the following options are supported:

dutch	french	german
italian	norsk	russian
spanish	dansk	polish
finnish	czech	catalan
brazil	turkish	

There is even an option for reformatting the `web` style to a print format!

The capabilities of the `web` package and its options are discussed below. Any comments and suggested improvements (new features) would be greatly appreciated.

#### 2.2. Package Requirements

The `web` package was designed for screen presentations tutorials, such as classroom or conference lectures, short technical articles, etc.; consequently, the `article` class of  $\LaTeX$  seems to be sufficient for these purposes. Though you can use `web` with any of the standard classes that define the `\section`, `\subsection` and `\subsubsection` commands, the package is really meant to be used with the `article` class. It is **strongly** suggested!

The package heavily depends on Sebastian Rahtz' `hyperref` package (now maintained and developed by Heiko Oberdiek). The `web` package was developed using version 6.56 of `hyperref`. Using prior versions of `hyperref` *may* lead to successful compilation, no guarantees offered. It is best to work with the most recent version of `hyperref`.

The `color` and `amssymb` packages are also required. The former is for obvious reasons, the later is to provide certain navigation symbols when the `navibar` option is invoked.

Finally, to create quality PDF documents, type 1 fonts *must* be used. Fortunately, type 1 fonts in the Computer Modern font set are freely available, and come with all the major freeware, shareware and commercial  $\TeX$  systems. If you haven't done so already, learn how to use the type 1 fonts.

In this regard, I have written an article that may be of interest to you entitled “*Using L<sup>A</sup>T<sub>E</sub>X to Create Quality PDF Documents for the WWW*”, see reference [10].

### 3. Basic Package Options

To use the web package, insert into the preamble of your document the following:

```
\usepackage[<driver_option>,<other_options>]{web}
```

Replace <other\_options> with any of the options recognized by web; see [Section 30](#) for a complete list of options. The first and optional argument <driver\_option> above defines the driver to be used; for example,

```
\usepackage[dvipsone]{web}
```

Currently, the web package supports five drivers: dvipsone, the dvi-to-ps converter by Y&Y, Inc.; dvips, the freeware dvi-to-ps converter; pdftex, the tex-to-pdf application; and dvipdfm, the dvi-to-pdf application by Mark Wicks,<sup>4</sup> and the commercial T<sub>E</sub>X system for the Mac, textures and T<sub>E</sub>Xshop.

► The package has been tested using `\documentclass{article}` and it is *strongly* recommended that this class be used; however, Web does support the book class as well.

#### 3.1. Setting the Driver Option

You can set your driver option in one of three ways:

- Pass as a local option:  
`\usepackage[<driver_option>]{web}`
- Pass as a global option:  
`\documentclass[<driver_option>]{article}`
- Create the file `web.cfg` with the single command in it:  
`\ExecuteOptions{<driver_option>}`  
Place the file `web.cfg` in any folder where L<sup>A</sup>T<sub>E</sub>X looks for input files. Then, you need only type `\usepackage{web}`.

Note that <driver\_option> is any of the following options: dvipsone, dvips, pdftex, dvipdfm or textures

The macros of the web package have been extensively tested using the Y&Y T<sub>E</sub>X System for the dvipsone option and a MiK<sub>T</sub>E<sub>X</sub> System ([www.miktex.org](http://www.miktex.org)) for the dvips, pdftex and dvipdfm options.

<sup>4</sup><http://odo.kettering.edu/dvipdfm/>

### 3.2. The `tight` Option

In an effort to compact more material per page, I've introduced a `tight` option. When this option is used, many of the list parameters are redefined so that there is not so much space around these environments, and between items.

```
\usepackage[<driver_option>,tight,<other_options>]{web}
```

This screen version of this manual was typeset with the `tight` option, the print version was typeset without it.

### 3.3. The `usesf` Option

For presentations, some people like a sans serif font. Using this option make san serif the default font style.

### 3.4. The `draft` Option

When you take this option, graphic overlays are not allowed. This is useful when you rely heavily on graphic overlays, but during the development phase, don't need to read and re-read your overlays. The defined background colors will be used instead. Remove this option to build the final version of your document.

### 3.5. The `nobullets` Option

Traditionally, the subsections have been denoted by a bullet, by using this option, you can force the use of numbers for the subsections, This option automatically executes the `latexoc` option.

There is another option `forceSubSubNumbers`, which does the same thing as `nobullets`. This option is included for compatibility with the AcroTeX Presentation Bundle (APB).

### 3.6. The `unicode` Option

This option is passed to `hyperref`, and is placed in `Web` as a convenience. If `eforms` is subsequently loaded, `eforms` (and `exerquiz`) will accept  $\LaTeX$  markup in the optional argument of the form fields. See the [eforms manual](#) for details.

### 3.7. The `useui` Option

This option is passed to `eforms`, and is placed in `Web` as a convenience. If `eforms` is subsequently loaded, `eforms` loads the `xkeyval` package, and key-value pairs are defined for use in the optional argument of form and link commands. See the [eforms manual](#) for details.

## 4. Setting Screen Size

Beginning with version 2.0, the screen size can be set by the author. There are two ways to do this: (1) use the macros `\screensize` and `\margins` (These are the same macros, slightly redefined, for setting the screen size used by Radhakrishnan in his fine screen package `pdfscreen`.); or (2) use a screen design option. The next two sections address each of these in turn.

### 4.1. Custom Design

There are six dimensions that need to be specified. As with `pdfscreen`, the two commands `\screensize` and `\margins` are used for doing so.

The command `\screensize` takes two length parameters:

```
\screensize{<height>}{<width>}
```

The `<width>` and `<height>` parameters are the desired screen size of the page. The screen version of this manual uses

```
\screensize{3.72in}{4.67in}
```

The other command, `\margins`, which determines the desired margins, takes four length parameters:

```
\margins[<panel_width>]{<left>}{<right>}{<top>}{<bottom>}
```

The values of `\textheight` and `\textwidth` are computed based on the screen size and the margins. The margin settings for this document are given below:

```
\margins{.25in}{.25in}{30pt}{.25in}
```

The optional first parameter `<panel_width>` is used to set the width of the panel, if there is one. The default is 1in.

► An important comment about the third parameter `<top>` is the following: as with `pdfscreen`, we put `\@Topmargin=<top>`. The running header fits within the top margin (this varies from standard L<sup>A</sup>T<sub>E</sub>X practice). The web package dimension `\web@Topmargin` is the distance from the top of the screen down to the top of the running header. Thus,

$$\@Topmargin = \web@Topmargin + \headheight + \headsep$$

Also, `\web@Topmargin` can be used to adjust the positioning of a running header, which is specified in the `\margins` command. The default value of `\headheight` is 8pt, so the value of `\headsep` is determined by the above equation. See the `web.dtx` file for more details.

### 4.2. Screen Design Options

For your convenience, I've included six options, `designi`, `designii`, ... `designviii`. The first one roughly corresponds to the original screen dimensions of `web`. The dimensions of these designs options are

Design Option	width × height	AR	Design Option	width × height	AR
designi	4.67in × 3.736in	5:4	designv	6in × 4.5in	4:3
designii	5in × 4.5in	10:9	designvi	4.67in × 4.17in	28:25
designiii	6in × 5in	6:5	designvii	10in × 7.5in	4:3
designiv	5in × 4in	5:4	designviii	6.67in × 3.75in	16:9

Most of these design values were chosen for their visual appeal, or for some particular need. The designs `designv` and `designvii` have an aspect ratio of 4:3, this is a standard for video displays, while `designviii` is another one with a standard aspect ratio for wide screen computer monitors.

To select a particular design for your document, simply include its name amongst the option list for the web package; for example,

```
\usepackage[designv, pdftex]{web}
```

creates a document with dimensions of `designv`, 6 in × 4.5 in.

► When you specify a screen design, the macros `\screensize` and `\margins` are re-defined to gobble up their parameters. To define a custom screen size, therefore, do not specify a screen design option for `web`.

### 4.3. `\setScreensizeFromGraphic`

If a design is *not specified* in the option list of `Web`, you can use a graphic to set the screen size with `\setScreensizeFromGraphic`. The command takes two parameters that correspond to the parameters of `\includegraphics`, and are just passed to `\includegraphics`. This is useful if you have a graphic to be used for a template but cannot be deformed to fit one of the standard designs. The solution is to create a screen size matching the graphic, then using that graphic as a template. Example usage,

```
\margins{1in}{1in}{24pt}{.25in}
\setScreensizeFromGraphic{acro_30}
\template{acro_30}
```

It may become necessary with some graphics to use the `hiresbb` option for the command `\includegraphics`, both in `\setScreensizeFromGraphic` and `\template` like so,

```
\margins{1in}{1in}{24pt}{.25in}
\setScreensizeFromGraphic[hiresbb]{acro_30}
\template[hiresbb]{acro_30}
```

This command should not be used with a panel option for the text screen size will be adjusted for the panel, and the graphic will be rescaled appropriately.

### 4.4. Using `\addtoWebHeight` and `\addtoWebWidth`

There may be an occasion when you've chosen your design dimensions (`designi`, `designii`, etc.), but you have determined the dimensions provided by the selected design are not quite *wide* or *high* enough. You can make adjustments to the width and height without explicitly specifying the commands `\margins` and `\screensize`, by using `\addtoWebHeight` and/or `\addtoWebWidth`. The syntax for each is

```
\addtoWebHeight{<length>}
\addtoWebWidth{<length>}
```

These two can only be executed in the preamble, and they add (or subtract) the specified `<length>` to the height and width of the page. For example,

```
\addtoWebHeight{1in}
```

adds one inch to the height of the page. This command is roughly equivalent to executing,

```
\screensize{<current_height>+1in}{<current_width>}
```

#### 4.5. Using a panel option

When the `rightpanel` or `leftpanel` option is used the page is divided into the text screen and a panel (either on the left or right of the page). The default width of the panel is `1in` and the minimum width of the panel is `1in`.

When designing the screen using `\screensize` and `\margins`, the panel width of the panel can be set by the first optional parameter of `\screensize`, as described in section 4.1, page 17. When using one of the standard design options, the panel width is set to the default value of `1in`. To set the panel width to another value, use `\panelwidth` in the preamble. `\panelwidth{1.25in}` sets the panel width to 1.25 inches.

If you set the panel width to a length less than the minimal panel width, the minimal panel width (`1in`) will be used. You can reset the minimal panel width using `\minPanelWidth`. See section 13.3, page 35, for more details on this command.

## 5. Hyperref Options

The web package loads `hyperref` into the document and sets some selected options of that package; therefore, including the `hyperref` package is not needed in the preamble of your own document.

Any additional `hyperref` options that are needed can be introduced into the package using `hyperref`'s `\hypersetup` macro, for example,

```
\documentclass{article}
\usepackage[dvipson]{web} % or dvips or pdftex

% Declare additional hyperref options using \hypersetup
\hypersetup{pdfpagemode=None,bookmarksopen=false}
```

Documentation of the options that `hyperref` recognizes can be had by either  $\TeX$ ing the file `hyperref.dtx`, or by getting a copy of the *The  $\TeX$  Web Companion* [5] by Michel Goossens, *et al.*

## 6. Running Headers and Footers

There are convenience commands setting the running headers and footers; they are `\lheader`, `\cheader`, `\rheader`, `\lfooter`, `\cfooter`, and `\rfooter`. Each of these

takes a single argument, which is the text for the header or footer in the position desired. In the standard setup for `Web`, `\lheader`, `\rheader` and `\cfooter` are used for, respectively, the current section title, the current page and the navigation bar (if activated).

You can set some basic formatting styles for the header and footer using `\headerformat` and `\footerformat`. For example, executing

```
\headerformat{\bfseries\color{red}}
```

causes the running header to appear in bold red color.

There are a couple of hooks for the header and footers, these are `\webheadwrapper` and `\webfootwrapper`. These commands can be used to stylize the entire header or footer. They can be defined to have one argument. The argument is the content of the header or footer. For example,

```
1 \headerformat{\bfseries\color{red}}
2 \footerformat{\bfseries\color{red}}
3 \renewcommand{\webheadwrapper}[1]{\fcolorbox{red}{yellow}{%
4   \makebox[\linewidth-2\fboxsep-2\fboxrule][s]{#1}}}
5 \renewcommand{\webfootwrapper}[1]{\fcolorbox{red}{yellow}{%
6   \makebox[\linewidth-2\fboxsep-2\fboxrule][s]{#1}}}
```

Lines (1) and (2) makes declares bold font for the header and footer. Lines (3) and (4) defines a `\fcolorbox` with hideous red and yellow colors; lines (5) and (6). Note that adjustment in the width to account for the `\fcolorbox` rule width and the space surrounding the text in the `\fcolorbox`.

On occasion you may want to have no running headers at all, this can be accomplished by saying in the preamble, or anywhere, `\lheader{}`, `\cheader{}`, and/or `\rheader{}`. Another solution is to use the commands `\clearHeaders` and `\restoreHeaders`. The command `\clearHeaders` first saves the current definitions of the headers then sets the headers to the empty header. The command `\restoreHeaders` restores the definitions that were in effect when the last `\clearHeaders` was executed.

**No running headers on section pages.** One user did not like a running header that includes a section title on the same page where the section begins. The default behavior is to create a running header where the section title does not appear in the header on the same page as the section is inserted, on subsequent pages, the section head appears, I hope.

The two commands control this behavior, the default is *to not show* the section heading (`\lheader`) on the page where the section begins. Use `\headersOnSectionPage` to place headers on the same page as the beginning of a section. You can also change back to the default behavior using `\noHeadersOnSectionPage` (the default).

## 7. The Title Page and TOC

The title page is constructed from the values of the macros: `\title`, `\author`, `\university`, `\email`, and `\version`. The values of some of the macros `\title` and `\author`

are also transferred to the PDFDocInfo section of the Acrobat/Adobe Reader. (Press `Ctrl+D` while viewing a PDF document to see this information.)

Additionally, the values of `\subject` and `\keywords`—which do not appear on the title page—are inserted into the PDFDocInfo section.

The colors of the text corresponding to `\title`, `\author` and `\university` on the title page can be set by using `\titleColor`, `\authorColor` and `\universityColor`. Each takes a single argument, a named color. The defaults are

```
\universityColor{blue}
\titleColor{black}
\authorColor{black}
```

For more information on defining named colors, see the documentation of the `color` or `xcolor` packages. A simple example would be

```
\definecolor{MyOwnColor}{rgb}{0.945,0.87,0.518}
```

► When the `pro` option is used, all colors can be specified through the `\selectColors` command, see ‘[Selecting Colors](#)’ on page 49.

### 7.1. Basic Information Commands

The basic information commands allows you a convenient way to enter the title and author of the document, as well as other information. To enter this basic information, just fill in the values of all the basic macros briefly described below.

For example, the following is a copy of the title information for this document:

```
% \title,\author,\subject,\keywords are sent to DocInfo
\title{The Web and Exerquiz Packages Manual of Usage}
\author{D. P. Story}
\subject{How to create on-line exercises and quizzes}
\keywords{LaTeX,hyperref,PDF,exercises,quizzes}

% \university,\email,\version are used only on title page
\university{Northwest Florida State College\
  Department of Mathematics}
\email{dpstory@acrotex.net}
\version{1.0}
\copyrightyears{1999-2010}
```

You can optionally specify the `\date`. The web packages uses the value of `\date` at the bottom of the title page. There is says, “Last Revision Date: `<date>`”. If the `\date` command is not used, the current date is used.

► The `\title`, `\author`, `\subject`, `\keywords` are a convenient way of entering information in the Document Information fields—see

```
File > Document Info > General... (Ctrl+D)
```

in the Acrobat/Adobe Reader.

If `\title` contains control sequences that do not expand to the Standard PDFDocEncoding character set, Distiller will be thrown into a tailspin; `hyperref` defines the `\texorpdfstring` macro<sup>5</sup> to avoid these kinds of problems. For example,

```
\title{The \texorpdfstring{$e^x$}{EXP} Function}
```

The first argument is the one that is typeset (on the title page, the title of the document will be ‘The  $e^x$  Function’); the second argument is the one that is sent to the title field of DocInfo in the Adobe Reader (and will read ‘The EXP Function’).

Of all the **Basic Information Commands**, as listed on page 21, use `\texorpdfstring` only with the `\title` and `\author`; these are the only two that are both typeset and placed in the DocInfo field of the Adobe Reader.

► `\texorpdfstring` works for `\section`, `\subsection`, etc. as well.

Having entered the values of the basic information commands, you can now type the standard sort of L<sup>A</sup>T<sub>E</sub>X commands of `\maketitle` and `\tableofcontents`:

```
\begin{document}
\maketitle
\tableofcontents
...
...
\end{document}
```

✎ Use the file `webeqstst.tex`, which comes with the distribution, as a prototype or template for your own document.

► When the `pro` option is in effect, the basic document information just described can be entered through the `\DeclareDocInfo` command, as described on page 37.

## 7.2. Title Page Structure

The title page is divided into three parts: top, middle and bottom.

- **Title Top:** The content of the top is determined by the `\topTitlePage` command. (This command can be redefined, but it is not recommended.) The `\topTitlePage` command expands to three elements: the university (affiliation), the title, and the author(s), in that order vertically. These are the values of the commands `\university`, `\title` and `\author` discussed in the previous section.
- **Title Middle:** The `\optionalPageMatter` command is used to enter content into this part of the title page. This middle part is optional; if `\optionalPageMatter` does not appear in the preamble, this part of the title page is empty. Here is an example of usage:

```
\optionalPageMatter{\vfill
\begin{center}
```

<sup>5</sup>The code for handling PDFDocEncoding for `hyperref` is due to Heiko Oberdiek

```

\fcolorbox{blue}{webyellow}{
\begin{minipage}{.67\linewidth}
\noindent\textcolor{red}{\textbf{Abstract:}} This
file attempts to teach you how to create a simple
\LaTeX\ document.
\end{minipage}}
\end{center}}

```

The above definition will create the framed box seen below.

**Abstract:** This file attempts to teach you how to create a simple  $\LaTeX$  document.

The `\optionalPageMatter` appears just below the `\webauthor` and above the directory listing if there is any.

- **Title Bottom:** Bottom of the title page is controlled by the contents of the command `\titlepageTrailer` and consists of the values of the commands `\email`, `\version`, `copyrightyears` Below is a rough figure depicting the location of the elements found in the title page trailer.

```

<copyright notice>                                \thewebemail
<revision date>                                   \webversion

```

The definition of these elements are as follows:

```

\def\maketitle@trailer@u1{\web@copyright\ \web@copyright@symbol\
\webcopyrightyears}
\def\maketitle@trailer@l1{\web@revision\ \@date}
\def\maketitle@trailer@ur{\thewebemail}
\def\maketitle@trailer@lr{\webversion}

```

The `u1` corner: The macro `\web@copyright` expands to ‘Copyright’, it can be redefined, or removed by executing `\nocopyright`. `\webcopyrightyears` expands to the argument of `\copyrightyears`.

The `l1` corner: The `<revision>` expands to `\web@revision` followed by `\@date`. The `\web@revision` expands to “Last Revision Date:”, and can be changed by using `\revisionLabel` (for example `\revisionLabel{Published:}`). The `\@date` expands to the argument of the `\date` command, or if this was not used, expands to the current date.

The `ur` corner: The upper-right corner goes the email address, as enter through the `\email` command.

The `lr` corner: The `\webversion` command goes into the lower-right corner. The command expands to `\web@versionlabel\ <version_number>`. The command `\web@versionlabel` expands to “Version”. This label can be changed by using `\versionLabel` (for example, `versionLabel{Attempt}`). Finally, the variable `<version_number>` is the number entered through the `\version` command.

The font size of the title page trailer is set by the command `\trailerFontSize`, the default is `\footnotesize`. This command must be re-defined in the usual way: for example, `\renewcommand{\trailerFontSize}{\small}`.

► When using the pro option, the elements of the title page trailer can be entered by using `\DesignTitlePageTrailer`, see ‘[Greater Control over Title Page Trailer](#)’ on page 47.

### 7.3. Redefining `\maketitle`

The arguments of the ‘[Basic Information Commands](#)’ on page 21 macros, as just discussed, are used to define text macros with no parameters; for example, when you type `\title{Web Package}`, the macro `\title` takes its argument and defines a macro `\webtitle` that expands to ‘Web Package’.

You can redesign the title page to suit your needs simply by redefining `\maketitle`: rearrange the macros listed in the second column of [Table 1](#) on the page, or include a graphic, or change the background color. To redefine `\maketitle`, use the commands:

```
\renewcommand\maketitle{...your design...}
```

See the definition of `\maketitle` in the `web.sty` file for an example.

This macro	defines this macro
<code>\title</code>	<code>\webtitle</code>
<code>\author</code>	<code>\webauthor</code>
<code>\subject</code>	<code>\websubject</code>
<code>\keywords</code>	<code>\webkeywords</code>
<code>\university</code>	<code>\webuniversity</code>
<code>\email</code>	<code>\webemail</code>
<code>\version</code>	<code>\webversion</code>
<code>\copyrightyears</code>	<code>\webcopyrightyears</code>

Table 1: The Basic Information Macros

When making the design, it is useful to know that the Web package uses the command `\hypertarget` to create a named destination, ‘webtoc’, in the table of contents. Use this `webtoc` to jump to the table of contents using the macro `\hyperlink`.

► When the pro option is in effect, a slight redefinition of `\maketitle` occurs. See the details in section 14.3, ‘[The Title Page: `\maketitle`](#)’ on page 43.

### 7.4. The Title Page Directory

When the `usedirectory` option is taken, a directory listing appears on the title page. The default structure of the directory is

#### Directory

- [Table of Contents](#)
- [Begin Article](#)

There are several commands that can be used to modify the directory.

```
\directoryName{Directory}
\tocName{Table of Contents}
\dirContentLink{Begin \hyperlink{\web@Start.1}{Article}}
```

These three define the default text strings that appear. In the case of `\dirContentLink`, there is a built-in `\hyperlink` to the start of the article.

```
\formatWordDirectory{\bfseries\large}
\formatDirectoryItems{\bfseries}
```

The formatting of the `\directoryName` can be set by `\formatWordDirectory`, the default definition is given above. The (global) formatting for the listing of the items can be set using `\formatDirectoryItems`, the default is shown above.

```
\removeDirTOC
\removeDirArticle
```

These two commands remove the “Table of Contents” and the “Begin Article” items from the list, respectively.

```
\addtoDirList{<new item>}
\dirTOCItem
\dirArticleItem
```

To add a new item to the end of the directory list, use `\addtoDirList`, the argument is the new item. The other two commands `\dirTOCItem` and `\dirArticleItem` are commands representing the first two default items. For example, typing

```
\addtoDirList{My home page is \url{www.example.com}}
```

in the preamble, adds this item to the end of the list. Should you want your home page listed first, you could do the following

```
\removeDirTOC
\removeDirArticle
\addtoDirList{My home page is \url{www.example.com}}
\addtoDirList{\dirTOCItem}
\addtoDirList{\dirArticleItem}
```

The following are for creating some special effects around the directory, such as putting the directory into a multicolumn format.

```
\priorDirMatter{<text/commands>}
\afterDirMatter{<text/commands>}
\priorDirList{<text/commands>}
\afterDirList{<text/commands>}
```

Below is a partial listing of the definition of `\webdirectory`, the internal name for the directory. We present the code so you can see where the prior/after commands write to.

```
\web@priorDirMatter
\begin{flushleft}{\web@formatWordDirectory\web@directory}%
\web@priorDirList
\vspace{\vspaceAfterDirName}%
\begin{itemize}
<directory listing>
\end{itemize}
\web@afterDirList
\end{flushleft}
\web@afterDirMatter
```


Contained within the code is another command, `\vspaceAfterDirName`, not mentioned before. It is used to adjust vertical spacing after the directory name and the beginning of the itemize list.

A final command that may be useful is `\directoryhook`. It is used for manipulating the directory as a whole.

```
\directoryhook{
  <code that manipulates #1>
}
```

The argument of `\directoryhook` is some  $\text{\LaTeX}$  code that manipulates `#1`, within the argument `#1` refers to `\webdirectory`, the internal name of the directory. The following example places a colored box around the directory, and centers it on the title page.

```
\directoryhook{%
  \begin{center}
  \setlength{\topsep}{0pt}%
  \fcolorbox{red}{webyellow}{%
  \parbox{.4\linewidth}{#1}}%
  \end{center}
}
```

 See the file [multicol\\_dir.pdf](#) for examples of many of these commands. This file is posted on the [AcroTeX Blog](#).

## 7.5. The TOC for Web

The Web style comes with its own table of contents format, as seen in the table of contents for the screen version of this document. The amount of indentation can be adjusted using `\tocindent`. The default is

```
\tocindent{20pt}
```

There is another relevant parameter, `\widestNumber`. The value of the argument of this command sets the amount of indentation for the subsection numbers. The default is

```
\widestNumber{0.0.}
```

This is a template for the subsection numbers, the default is a one digit section number and a one digit subsection number. In the preamble of this document, I've set `\widestNumber{0.00.}`, since some subsection numbers have two digits.

The color of the heading of the table of contents is set through

```
\tocColor{<named_color>}
```

Specifying a color through this command at any time before the creation of the table of contents will change the color of the heading. The default is blue.

- ▶ If you prefer the standard  $\LaTeX$ , the `latexdoc` option can be used.

## 7.6. The `usedirectory` Option

When the `usedirectory` option is specified, a short directory appears on the first page, see ‘[The Title Page Directory](#)’ on page 24 for more details.

- ▶ **Important Note:** In previous versions of `Web`, the directory automatically appeared, and there was a `nodirectory` option to make it go away. Now, the directory does not appear by default, and the option `usedirectory` makes it appear. The `nodirectory` option is still defined, but does nothing.

## 7.7. The `latexdoc` Option

If you don't like the default design for the table of contents, you can always recover the standard  $\LaTeX$  table of contents by using the `latexdoc` option with the `web` package:

```
\usepackage[latexdoc]{web}
```

Should you want to go with this option, you might consider including

```
\hypersetup{linktocpage}
```

Look at the table of contents with and without this `hyperref` option to decide which you prefer.

## 7.8. The `centertitlepage` Option

Beginning with version 5.0, there is a `centertitlepage` option that attempts to center the title page better when the `forpaper` option is taken. Seems to work best with `designv`. Will try to improve.

## 7.9. The `\makeinlinetitle` Command

For some short documents,<sup>6</sup> a formal title page may not be needed or desirable. In this case, use `\makeinlinetitle`, the “in-line” title.

```
\makeinlinetitle
```

The command, which can be redefined to your needs, has the following design:

	<code>\webtitle</code>	
<code>\webuniversity</code>	<code>\webauthor</code>	<code>\webemail</code>
<code>date</code>		<code>\webversion</code>

The table above extends the entire `\linewidth`. The values of `\webtitle`, `\webuniversity`, `\webauthor`, `\webmail` and `\webversion` are populated by their command counterparts, see [Table 1](#), page 24. The `date`, as appears in the lower left row, will expand to the current date, or the value specified by the `\date` command.

Below are notes in the case the `pro` option is in effect:

- The basic information above can be entered through `\DeclareDocInfo` command as described in section 14.1, entitled ‘[\DeclareDocInfo](#)’ on page 37.
- If the value of the `prepared` key is not specified, then `date` is the current date, otherwise, `date` is the one specified as the value of the `prepared` key.
- If the value of the `talksite` key is not specified, then `\webversion` is used; but if `talksite` is specified, the value of this key is used instead of `\webversion`.

## 8. Template Options

The Web Package has three options (and supporting commands) for creating colored backgrounds, graphics backgrounds, and various overlays.

### 8.1. The `usetemplates` Option

The `usetemplates` option activates the mechanism for creating colored backgrounds and graphic overlays. A complete discussion of the commands related to this option can be found in the section entitled ‘[Template Building and Management](#)’ on page 33.

- ▶ See the demo file `bgtest.tex` for examples.

### 8.2. The `leftpanel` and `rightpanel` Options

When either of these two options is specified, a vertical panel is created. See the section entitled ‘[Template Building and Management](#)’ on page 33 for a complete discussion of the commands related to these options.

- ▶ See the demo file `bgtest.tex` for examples.

<sup>6</sup>Short documents such as white papers, homework assignments, for example.

### 8.3. The `leftpanelprt` and `rightpanelprt` Options

These two options are the same as `leftpanel` and `rightpanel` *when there is no paper option* (for `paper`, for `colorpaper`). When a paper option is taken, these options do nothing. The options are useful for creating a paneled PDF for the screen, but the paper version that has no panel.

## 9. Navigation Aids

The `web` package offers a couple of navigation aids to help you move around: the `navibar` Option, and some [direction icons](#).

### 9.1. A Navigation Bar

Use the `navibar` option of `web` to add a navigation toolbar, as seen at the bottom of this page. Usage:

```
\usepackage[<driver_option>,navibar]{web}
```

The result is the navigation bar you see at the bottom of the page.

► The toolbar can be turned on or off by the following commands: `\NaviBarOn` and `\NaviBarOff`. The navigation toolbar at the bottom of the page was generated by the `\NaviBarOn`. `\NaviBarOff` was placed on the next page to turn off the bar. The exact code used on this page is

```
\NaviBarOn\AddToShipoutPicture*{\NaviBarOff}
```

The colors for the navigation bar can be set with commands `\navibarTextColor` and `\navibarBgColor`. Each take a single argument, a named color. The default colors are `webblue` and `webgray`, respectively.

### 9.2. `\newNaviIcon`

The `\newNaviIcon` can be used to define a navigation icon. The action of the icon can be to execute a menu item, perform a hyper-jump, or execute the JavaScript code. It takes six parameters:

Parameters

```
#1 = m, j, or l
#2 = command name of the new navigation icon
#3 = width of icon
#4 = height of icon
#5 = text to appear in the center of the icon.
#6 = if m: named menu action, e.g., NextPage, PrevPage, etc.
      if j: execute JavaScript
      if l: \hyperlink{arg} or \href{arg}
```

Once the `\newNaviIcon` command is executed, a new icon is defined. The name of this new icon is the value of parameter #2.

► **Example:**



```
\newNaviIcon[m]{\myNext}{34pt}{10pt}{Next}{NextPage}
\newNaviIcon[j]{\jsWarning}{34pt}{10pt}{Hi}{app.alert("Hi there")}
\newNaviIcon[l]{\linkJump}{34pt}{10pt}{Go}{\hyperlink{page.1}}
```

By typing `\myNext` `\jsWarning` `\linkJump`, we get

`Next` `Hi` `Go`

The colors for the navigation bar can be set with commands `\navibarTextColor` and `\navibarBgColor`.

### 9.3. Direction Icons

The up arrow you see in the upper right-hand corner was constructed using colored rules and the AMS symbol font, `amssymb`. The uparrow icon was produced by the command:

```
\insertnaviiconhere{\ArrowUp{\hyperlink{webtoc}}}
```

The definition of `\ArrowUp`, which be found in the `web.dtx` file is

```
\newcommand\ArrowUp[1]
{%
  \setlength{\fboxsep}{6pt}\normalsize
  \raisebox{-\depth}[0pt][0pt]{%
    #1{\web@colorbox@w@transparency{\web@directionIconBgColor}%
      {\textcolor{\web@directionIconTextColor}{\bigl\Uparrow}}}}%
  }
```

Or, more generally,

```
\insertnaviiconhere{\ArrowUp{link_command}}
\insertnaviiconhere{\ArrowDown{link_command}}
```

This will insert direction icons on the current page (I hope).

If you want a running direction icon you can use

```
\insertnaviiconhereafter{\ArrowUp{link_command}}
```

or

```
\insertnaviiconhereafter{\ArrowDown{link_command}}
```

► To discontinue a running arrow icon type


```
\defaultpageheader
```

on the page you want the arrow(s) to disappear.

The colors of the direction icons by using the commands `\directionIconTextColor` and `\directionIconBgColor`. Each command takes a named color as its argument; the default values are `webblue` and `webgray`, respectively.

#### 9.4. `\panelNaviGroup`

When the `leftpanel` or `rightpanel` options are chosen, a (navigation) panel is created. The command `\panelNaviGroup` can be used to create the standard navigation panel.

 See the sample file `bgtest.tex` for an example of usage.

### 10. The Language Options

The language options redefine all of the language dependent text macros that appear on the title page, in the table of contents, and in the running headers. Invoke these options in the usual way:

```
\usepackage[<driver_opt>,<lang_opt>]{web}
```

Here, `<lang_opt>` is one of the following values: `dutch`, `french`, `german`, `italian`, `norsk`, `russian`, `spanish`, `polish`, `finnish`, `czech`, `catalan`, `brazil`, and `turkish`.

The `web` and `exerquiz` packages seem to be compatible with the `babel` package; you can use

```
\documentclass{article}
\usepackage[french]{babel}
\usepackage[dvips,french]{web}
\usepackage{exerquiz}
```

subject to the usual restrictions on these language packages. (Don't use characters declared active by these languages within a `\label`, or as a field name for a quiz.

The translations for the `french` option is due to the tremendous efforts of Jean-Michel Sarlat, and Michael Wiedmann did the translations for the `german` option.

### 11. Paper Related Options and Commands

In this section, several paper options are discussed, and one command.

#### 11.1. The `forpaper` Option

Some people may want to create exercises using the `exercise` environment for a paper document. The `forpaper` option can be used to remove the color from the document, and to restore the standard `\textheight` of a standard `article` class  $\text{\LaTeX}$  document.

The `\textwidth` is determined from the `\screensize` and `\margins` parameters or by the design option (see [Screen Design Options](#)), *if any are given*; consequently, the line breaks are the same for the “web” version and the “print” version. If the screen dimensions are not set (by a design option or by the `\screensize` and `\margins` commands) when using the `forpaper` option, the standard  $\text{\LaTeX}$  page layout dimensions are used for the class.

Using the `forpaper` option with the `latexlayout` option will give you the standard  $\text{\LaTeX}$  `\textwidth`.

The `forpaper` option also changes the `\newpage` command to `\par\medskip` at the end of each solution—we don't want to waste paper now, do we?

Finally, there is a Boolean switch `\ifeqforpaper`, which you are free to use to refine the look your `forpaper` version.

### 11.2. The `forcolorpaper` Option

Same as the `forpaper` option, but the color operators are not turned off.

### 11.3. The `latexlayout` Option

For those who want to go “totally native,” use the `latexlayout` and `forpaper` options together. When the `latexlayout` option is used, the page layout redefinitions of `web` are bypassed, leaving the original layout values of the `article` class of  $\LaTeX$ .

► If the `latexlayout` option is taken, all templates are turned off, and the `forcolor-` option is executed. To remove color, you need to explicitly take the `forpaper` option.

### 11.4. The `uselatexparts` and `uselatexchapters` Options

As described in [section 15](#), page 53, the `Web` package redefines the `\part` and `\chapter` commands for the purpose of allowing easier access to redefining how these section elements are to appear in the document and in the table of contents. You can bypass these definitions by using `uselatexparts` and the `uselatexchapters` options. This may be useful if you want to use `AeB` to create a paper document and you want to use that standard definitions, or perhaps want to use another package to define the look of the table of contents or of the `\part` and `\chapter` commands. (The other sectioning comments can be overridden as well, but any controls provided by the `pro` option will be lost.)

### 11.5. The `\useFullWidthForPaper` Command

We introduce a command of resetting the page layout paper *when the `forpaper` option is in effect*. The primary use is to create standard documents for academics such as homeworks, syllabuses or any handout to the student.

This command is used to set the page layout to its maximum width, given a 1 inch margin. The height is maximized after taking into account other page parameters that effect it. The parameters `\marginparwidth` and `\marginparsep` are set to zero; for this layout, there are no marginal comments. The command may be redefined as desired.

The command should be used in the preamble, otherwise, it has no effect.

## 12. Formatting for screen and paper

As we learned in [Section 11](#), `Web` can format a document in a for screen or for paper. The `Web` package provides several commands and environments for changing content as needed.

When the `forpaper` option is used, the switch `\ifeqforpaper` is set to true. You can use this switch, to get different content for the printed page or the screen page, or you can use the convenience command `\prtscr`:

```
\prtscr{<print_text>}{<screen_text>}
```

**Command Description:** This command expands to `<print_text>` if the `forpaper` option is taken and to `<screen_text>` if not.

The `\prtscr` command is useful for small changes in content, not involving verbatim text or paragraph breaks. `Web` defines two additional environments, `forpaper` and `forscreen`

```
\begin{forscreen}
....
....
....
\end{forscreen}
```

**Environment Location:** Anywhere.

Use the `forscreen` environment to insert commands or content meant only for the screen.

```
\begin{forpaper}
....
....
....
\end{forpaper}
```

**Environment Location:** Anywhere.

Use the `forpaper` environment to insert commands or content meant only for the paper document.

Another command, `\NewPage`, is useful for formatting the same document for print as well as screen


```
\NewPage
```

**Command Location:** Anywhere.

**Command Description:** This command expands to `\newpage` if the `forpaper` option is *not taken*, and does nothing, otherwise. Useful for inserting page breaks in the screen version that are not needed in the paper version.

### 13. Template Building and Management

The `Web` package has a template building capability. You can conveniently create backgrounds for your page, insert an arbitrary number of graphic overlays, create a left or right (navigation) side panel, define your own navigation icons that appear in the panel, and write material that will appear in a panel.

 The demo file for the template feature is `bgtest.tex`.

### 13.1. Template options

As with pdfscreen by Radhakrishnan C. V., we shall have the two options, `leftpanel` and `rightpanel`. In addition to these two, there is the `usetemplates` option. Use the option `usetemplates` if you want to use colored backgrounds or overlays without a left or right panel.

The template, or overlay, capability of the Web Package requires the use of two  $\LaTeX$  Packages: `everyshi.dtx`, by Martin Schröder, and `eso-pic.dtx`, by Rolf Niepraschk. If any of the three template options (`usetemplates`, `leftpanel` or `rightpanel`) are used, the `eso-pic` package is automatically included by web. The `eso-pic` package, in turn, inputs the `everyshi` package. These two packages need to be present on your system, unpacked, and in the search path of  $\LaTeX$ .

Templates, or overlays, are available for the `dvipsone`, `dvips`, `pdftex`, and `dvipdfm` options.

### 13.2. Text Screen Template

You can specify a graphic that will be overlaid onto the text screen, that portion of the screen to which  $\LaTeX$  content is written. If a panel option has not been specified, this is the whole screen; otherwise, it is that portion of the screen outside the panel.

If one of the options `usetemplates`, `leftpanel` or `rightpanel` is specified, the commands

```
\template{<graphics_file_name>}
\textBgColor{<named_color>}
```

insert a background graphic and a background color, respectively, onto the text screen region. The `\template` command will rescale the graphic to cover the entire text screen region.

Additional graphics can be overlaid with the `\AddToTemplate` command.

```
\AddToTemplate{<template_name>}
```

The command takes one argument, the *template\_name*. Define an overlay,

```
\newcommand\myTemplate
{%
  < commands to insert an overlay >
}
```

the *template\_name* for this template is `myTemplate`. (Note that there is no backslash.) To add this template to the list graphics to be overlaid onto the page, we would type

```
\AddToTemplate{myTemplate}
```

► **Example:** Insert the “Acro $\TeX$ ” logo in lower-left corner, offset by 36pt in the  $x$  and  $y$  directions.

```
\newcommand\AEBLogo
{%
```

```

\put(36,36){\includegraphics{acrotexlogo}}%
}
\AddToTemplate{AEBLogo}

```

Because the Web Package uses `eso-pic`, the commands will be executed within a `picture` environment. Within the `picture` environment, the reference point of the text screen is the lower-left corner. The above code puts the “AcroT<sub>E</sub>X” logo at coordinates of (36, 36) relative to the lower-left corner. The units are measured in (T<sub>E</sub>X) points.

► **Example:** Center the logo within the text screen region.

```

\newcommand\AEBLogoCenter
{%
\ifnum\arabic{page}>1\relax
\parbox[b][\paperheight][c]{\textscreenwidth}
{\centering\includegraphics{acrotexlogo}}%
\fi
}
\AddToTemplate{AEBLogoCenter}

```

See the section titled ‘[Template Management](#)’ on page 36 for details of how to manage your templates.

### 13.3. Panel Screen Template

When the `leftpanel` or `rightpanel` option is specified, a (navigation/logo) panel is created. The commands

```

\paneltemplate{<graphics_file_name>}
\panelBgColor{<named_color>}

```

set the overlay graphic and the background color, respectively. The graphic is rescaled to fit the panel region.

Once the panel and its background have been defined, contents and form elements can be placed on top of the panel. The command `\buildpanel` can be used for this purpose. For example, from the sample file `bgtest.tex`,

```

\buildpanel
{%
\href{http://www.math.uakron.edu/}
{\includegraphics[scale=.4]{uakron}}
\par\vspace{\stretch{1}}
\href{http://www.math.uakron.edu/~dpstory/acrotex.html}
{\rotatebox{-90}{\aebLogo}}
\par\vspace{\stretch{1}}
\panelNaviGroup % defined in web
}

```

The content of the panel is stacked from top to bottom.<sup>7</sup>

► Additional overlays can be added with `\AddToPanelTemplate`. The command may not be as useful as the panel overlay can always be rebuilt using `\buildpanel`.

The minimal width for the panel is set through the `\minPanelWidth` command, the default size is `1in`; that is, `\minPanelWidth{1in}` is executed by the package at start up.

```
\minPanelWidth{<length>}
```

**Command Location:** You can reset the minimal width by executing the command in the preamble before the `\screenwidth` and `\margins` commands. Or, place it in the `web.cfg`.

### 13.4. Template Management

In order to change backgrounds or templates, on any page, re-issue any one of the commands `\template` or `\textBgColor` (for the screen text region), or `\paneltemplate` or `\textBgColor` (for the panel region).

The panel overlay can be redesigned with `\buildpanel`, or some of the command components that make up the panel overlay can be redefined.

Templates that are inserted into the output stream using either `\AddToTemplate` or `\AddToPanelTemplate` can also be redefined on any page.

Templates, created by either `\AddToTemplate` or `\AddToPanelTemplate`, can also be *disabled* or *enabled* individually. For example, if the `AEBLogoCenter` template has been overlaid using the command

```
\AddToTemplate{AEBLogoCenter}
```

the template can be disabled (turned off) by typing

```
\disableTemplate{AEBLogoCenter}
```

on any page. (**Note:** The effects of this command may be not be seen until the following page.) Turn the template on by typing

```
\enableTemplate{AEBLogoCenter}
```

on any page.

For the panel region, there are commands for *disabling* (`\disablePanelTemplate`) and *enabling* (`\enablePanelTemplate`) as well. Each of these takes a *template\_name* as an argument.

There are a number of commands for *clearing* backgrounds and templates.

```
\ClearTextTemplate
\ClearPanelTemplate
```

These two clear background colors and background graphics.

<sup>7</sup>The command `\aebLogo` is defined in the file, `bgtest.tex`.

`\ClearBuildPanel`

This command will clear the build panel as well as the graphics and field elements that lay on top of the panel created by the `\buildpanel` command.

`\ClearAllTemplates`

The command `\ClearAllTemplates` is equivalent to executing `\ClearTextTemplate` and `\ClearPanelTemplate`.

`\ClearTextTemplateBuffer`

`\ClearPanelTemplateBuffer`

The commands will clear all overlays, including overlays created by `\AddToTemplate` and `\AddToPanelTemplate`.

► See the documentation file, `web.dtx`, for exact definitions of the commands in this section.

## 14. The pro Option

Version 5.0 of `Web` introduces the `pro`, which encompasses many new features. The `pro` option uses the very fine package `xkeyval` by Hendri Adriaens, which enables developers to create commands with key-value pairs.

The following new features are provided by the `pro` option:

1. `\DeclareDocInfo`: A data structure for setting various document properties.
2. A slightly re-designed `\maketitle`. Also, increased control over the placement of the elements of the title page.
3. Extensive control over the display of `\section`, `\subsection`, `\subsubsection` headings. Now color, fonts, size and so on can easily be set.
4. A choice of using section numbers (the default) or no sections, or dings.
5. A single data structure to set virtually all colors.

Details provided in subsequent sections.

### 14.1. `\DeclareDocInfo`

The preamble of your document should contain a number of keys that identify the document, including the title and author of the document. Some of this information is used to construct the title page, some is placed in the PDF, to be displayed in the 'Description' tab of the **Document Properties** dialog box, which is accessed through the `Ctrl+D` accelerator key, or through the menu system `File > Document Properties...` (for version 8, this is `File > Properties...`).

Information is passed through the `\DeclareDocInfo` command which takes a number of key-value pairs. This is a simple `xkeyval` interface to many of the text macros that are defined in `Web`.

```
\DeclareDocInfo{<key-value pairs>}
```

**Command Location:** Place in the preamble.

**Key-Value Pairs:** The following is a description of the key value pairs.

1. **title:** The title of the document.
2. **author:** The author or authors of the document.
3. **subject:** The subject of the presentation. Optional, this appears only in the ‘Description’ tab of the **Document Properties**.
4. **keywords:** A list of keywords that describe your document. Optional, this appears only in the ‘Description’ tab of the **Document Properties**. Some search engines use this field.
5. **university:** The university or company the author represents.
6. **email:** The email address of the author. This appears on the title page, and becomes an email link.
7. **version:** The version number of the document.
8. **versionLabel:** Text that precedes the version number. The default is `Version`.
9. **talkdate:** Date of the presentation.  
The fields `version` and `talkdate` occupy the same position on the title page. If the `talkdate` is specified, then the talk date will appear on the title page, of `talkdate` if not specified, the `version` will appear. Both `version` and `talkdate` can be specified, but in this case, it is `talkdate` that will appear. The `version` can be used for version management.
10. **talkdateLabel:** Text that precedes the date of the document. The default text for the talkdate label is `Presented:.`
11. **talksite:** Site of the presentation. This field can be used for generally anything, for example, you could specify your web site.
12. **copyrightyears:** Year(s) of the copyright of this publication, defaults to this year.
13. **prepared:** The date of preparation of the document, defaults to the day the file was compiled ( $\LaTeX$ ed). This was formerly the last revision date.
14. **preparedLabel:** The label that precedes the date `prepared`. The default string value is `Prepared:.` The old default was `Last Revision Date:.`, and will remain so if the pro option is not taken.
15. **copyrightStatus:** If the `aebxmp` package is loaded for advanced metadata, this key allows you to set the copyright status. Possible values `True`, `False`, or blank (no value, or the key not listed at all) corresponding to `Copyrighted`, `Public Domain` and `Unknown`, respectively. The values of this key are case sensitive, so you must enter `True` and `False`, with the first letter capitalized. If `aebxmp` is not loaded, specifying this key does nothing.

16. `copyrightNotice`: If the `aebxmp` package is loaded for advanced metadata, this key allows you to set copyright notice, short text describing the copyright, perhaps,

```
copyrightNotice={Copyright D. P. Story, 2009--2012}
```

If `aebxmp` is not loaded, specifying this key does nothing.

17. `copyrightInfoURL`: If the `aebxmp` package is loaded for advanced metadata, this key allows you to set the copyright info url, a url to a page on the web containing a more detailed description of the copyright. If `aebxmp` is not loaded, specifying this key does nothing.
18. `authors`: A list of authors, *each enclosed in braces*. Names of individual authors can be accessed using the JavaScript `info.Authors` property: the first author is `info.Authors[0]`; the second author is `info.Authors[1]`, etc.

```
authors={D. P. Story}{J\u00FCrgen Gilg}
```

Notice the use of unicode to create a u-umlaut (ü).

19. `Keywords`: Similar to `keywords`, but not the capital 'K'. Then `Keywords` is used to list the keywords, they are insert into the metadata as an array. The individual keywords can be accessed through a special JS function, `aKeywords()`, defined by the `aebxmp` package. The syntax is the same as `keywords`:

```
Keywords={AcroTeX.Net,XMP,E4X,Adobe Acrobat,JavaScript}
```

If you execute `aKeywords(0)`, the string "AcroTeX.Net" is returned. Executing `aKeywords(4)` returns "JavaScript", while executing `aKeywords(5)` returns undefined.

20. `authortitle`: The `authortitle` is a field that appears on the Additional Metadata dialog box. It can be used for whatever purpose you wish.

```
authortitle={Programming and Development, AcroTeX.Net}
```

21. `descriptionwriter`: This key fills a field on the

```
descriptionwriter={A good, well-liked guy}
```

Additional Metadata dialog box. For example

22. `customProperties`: Acrobat allows for the creation of custom properties which are accessible through the `info` object. The value of this key consists of one or more property definition(s) enclosed in braces. Each property definition has two required key-value pairs, `name` and `value`. The `name` needs to be a simple name consisting of letters and numbers (not starting with a number). The `value` string can be any string. See the example following:

```
customProperties={name=Developer,value={D. P. Story, Esq.}}
                 {name=Motivator,value=J\u00FCrgen Gilg}
```

Executing `this.info.Developer` returns the value of "D. P. Story, Esq.", while, `this.info.Motivator` returns "Jürgen Gilg".

**Example 1:** Example of usage of `\DeclareDocInfo` can be found in [Figure 2](#), page 41.

**Discussion of keywords versus Keywords.** `Keywords` will overwrite `keywords`; the value of `keywords` is passed to `hyperref` which enters them as a single string, while `Keywords` enters the keywords. as an array. There is no need to have both `keywords` and `Keywords`.

**Discussion of author versus authors.** Similarly, `authors` overwrites `author`, individual authors names can be accessed; however, unlike `keywords/Keywords`, it may make sense to use both `author` and `authors`. The `web` package takes a copy of the value of `author` and uses it in the title page. It is possible to have two versions of the author's names, one for presentation and one for the PDF info.

```
author={Dr. D. P. Story, Herr J. Gilg},
authors={D. Story,J. Gilg},
```

You can do the same thing, with the `author` key only by using the `\texorpdfstring` command of `hyperref`:


```
author={\texorpdfstring{Dr. D. P. Story, Herr J. Gilg}
{D. Story,J. Gilg}},
```

The difference is that with `authors`, individual authors are accessible through the JavaScript command `this.info.Authors`.

- **Two tricks of importance:** When the value contains a comma, then the whole value should be delimited by matching braces, as in the `talkdate` key-value in the **Example 1** above. The `hyperref` command `\texorpdfstring` is handy for giving alternate wording, when some of the  $\LaTeX$  commands do not transfer to the PDF's **Document Properties**. For example, the title might have been "A Discussion of  $e^x$ "; this title should appear in `\DeclareDocInfo` as follows:

```
title=A Discussion of \texorpdfstring{$e^x$}{exp(x)},
```

Now the phrase, "A Discussion of  $\exp(x)$ " will appear in the 'Title' field of the **Document Properties**.

 The demo file `web_pro` illustrates this command.

## 14.2. `\DeclarePageLayout`

The `\DeclarePageLayout` command gathers together many page layout parameters, both for  $\LaTeX$  and for Web, into one convenient package of key-value pairs.

**Command Location:** Place in the preamble.

```

\DeclareDocInfo
{
  title=My First Presentation,
  author=D. P. Speaker,
  university=My University,
  email=dpspeaker@myu.edu,
  talkdate={Dec.\ 17, \the\year},--
  talksite=The Talking University,
  subject=On the Theory of AcroTeX,
  keywords={AcroTeX.Net,XMP,E4X,Adobe Acrobat,JavaScript},
  copyrightStatus=False,
  copyrightNotice={Copyright D. P. Story, 2009--\the\year},
  copyrightInfoURL=http://www.acrotex.net,
  authors={D. P. Story}{J\u00FCrgen Gilg},
  Keywords={AcroTeX.Net,XMP,E4X,Adobe Acrobat,JavaScript},
  authortitle={Programming and Development, AcroTeX.Net},
  descriptionwriter={A good, well-liked guy},
  customProperties={name=Developer,value={D. P. Story, Esq.}}
                    {name=Motivator,value=J\u00FCrgen Gilg}
}

```

Figure 2: The \DeclareDocInfo Command

**Key-Value Pairs:** The following is a description of the key value pairs.

1. **design:** This key sets the screensize according to preset dimensions, permissible values are `designi`, `designii`, `designiii`, `designiv`, `designv`, `designvi`, `designvii`, and `designviii`. These correspond to the package options of the same names. This key is ignored if one of the design options was taken as a Web option. This key sets both the `\screensize` and `\margins` commands.
2. **screensizeOf:** This key sets the screensize only (does not change the margins), possible values are `designi`, `designii`, `designiii`, `designiv`, `designv`, `designvi`, `designvii`, and `designviii`. Use this key, in combination with the `margins` key, to use a standard design screensize, but with different margins.
3. **screensize:** Sets the screensize by executing the `\screensize` command. The key takes two dimensions as parameters:

```
screensize={<height>}{<width>}
```

The `screensize` key sets the screen to a height of `<height>` and a width of `<width>`. This key is ignored if a design option has been taken in the Web option list, or if the `\screensize` command has already been executed.

4. **margins:** Sets the dimensions of the margins of the screen page. The key takes four dimensions as parameters:

```
margins={<left>}{<right>}{<top>}{<bottom>}
```

There `<left>`, `<right>`, `<top>`, and `<bottom>` are the dimensions of the margins of the screen page. These are the same arguments as the `\margins` command. This key is ignored if a design option has been taken in the Web option list, or if the `\margins` command has already been executed.

5. `headheight`: Sets the standard page layout parameter `\headheight`. The default is 8pt.
6. `topmargin`: Sets the position of the running header. The default is 8pt for screen sizes, and the  $\LaTeX$  default is the forpaper option is taken. You can say

```
topmargin=\prtscr{<paper_dim>}{<screen_dim>}
```

To get a `topmargin` value that works for paper and screen. (Perhaps you have a border running around the page, and you need to reposition the header below the border. This repositioning differs depending on whether you compile with the forpaper option, or not.

7. `additionalheadsep`: Additional separation between the running header and the beginning of the text. This parameter modifies `\headsep`, which is computed automatically by the Web package. Use `additionalheadsep` to add in a little bit more, if the calculation does not yield good results. The default value is 0pt.
8. `marginparsep`: The horizontal distance between body and marginal notes. The default is 11pt. When the page size is designed for viewing on a computer, the margins are often times quite small, and marginal notes are typically not used. You can use a conditional to give two values one for screen and one for paper. See the `topmargin` key.
9. `marginparwidth`: The width of marginal notes. The default is .25in. When the page size is designed for viewing on a computer, the margins are often times quite small, and marginal notes are typically not used. You can use a conditional to give two values one for screen and one for paper. See the `topmargin` key.
10. `marginparpush`: The minimal vertical space between successive marginal notes. The default is 5pt.
11. `footskip`: Vertical distance separating the baseline of the last line and the baseline of the footer. The default is 30pt.
12. `webfootskip`: Same as `footskip`, but the `webfootskip` is measured up from the bottom edge of the document. The default is 4pt.
13. `panelwidth`: Sets the panel width. If not specified, the minimal panel width of 1in is used. Panels appear when either `leftpanel` or `rightpanel` is taken in the option list of Web. This key executes the `\panelwidth` command.
14. `panelsep`: Sets the separation between the text screen and the panel. The default is 10pt.

### 14.3. The Title Page: `\maketitle`

In this section, we described the enhanced features for controlling the title page.

- **The Title Page Structure**

The title page is divided into three parts: top, middle and bottom.

- **Title Top:** The content of the top is determined by the `\topTitlePage` command. (This command can be redefined, but it is not recommended.) The `\topTitlePage` command expands to three elements: the university (affiliation), the title, and the author(s), in that order vertically. These are the values of the keys `university`, `title` and `author` that appear in the `\DeclareDocInfo` command.
- **Title Middle:** The `\optionalPageMatter` command is used to enter content into this part of the title page. This middle part is optional; if `\optionalPageMatter` does not appear in the preamble, then this part of the title page is empty.
- **Title Bottom:** Bottom of the title page is controlled by the contents of the command `\titlepageTrailer` and consists of some of the document information entered in the `\DeclareDocInfo` command. By default, `\titlepageTrailer` lists the values of the `\DeclareDocInfo` keys `email`, `talkdate`, `talksite`, and `copyrightyears`, as described above. The font size of this bottom part is set by the command `\trailerFontSize`, the default is `\footnotesize`. This command can be re-defined in the usual way, for example,

```
\renewcommand{\trailerFontSize}{\scriptsize}
```

Figure 3, page 44, shows the basic composition of the title page of an AeB document. The title page elements are described as they relate to the key-values of `\DeclareDocInfo`, described on page 37.

- At the very top is the value of the `university` key. The color of this element can be set using the `universityColor` key of `\selectColors`, page 49.
- Next comes the value of `title`, its color is controlled using the `titleColor` key of `\selectColors`.
- The author follows, which is the value of `author`. The color is set by `authorColor` of `\selectColors`.
- The AeB logo is inserted using `\optionalPageMatter`. Normally, this macro does nothing unless it is defined. In this example, we have

```
\optionalPageMatter
{%
  \begin{center}
    \begin{minipage}{.67\linewidth}
      \centering\includegraphics[scale=.5]{AeB_Logo}
    \end{minipage}
  \end{center}
}
```

- Finally comes the `\titlepageTrailer`, a macro that can be redefined (see the package file `web.dtx` for its definition). This macro places the other elements at the bottom of the page:
  - The copyright year and email address, as given by `\email`. The color of the email address is set by `urlColor` through the command `\selectColors`. This color actually sets the color of all external URLs.
  - To the right, on the first line of the title page trailer is the value of `\talksite`. The color is the default color for text.
  - In the lower left is the date of the last revision of the document. The color is the default color for text.
  - In the lower right is the date the talk was given. The color is the default color for text.

Not shown in Figure 3 is the title page directory, which is turned off by default.

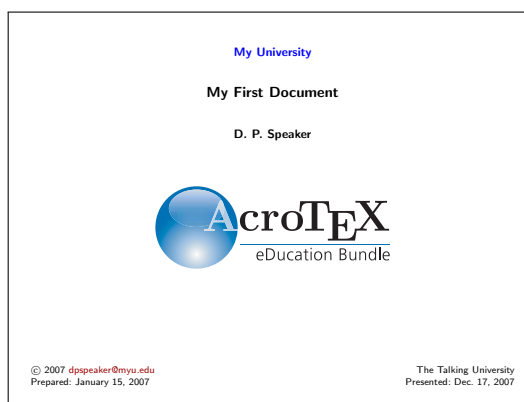


Figure 3: The Title Page

The title page layout is, of course, defined by the standard `\maketitle` command, which has been redefined by the pro option. In Web, the `\maketitle` has different behaviors, one for screen and one for paper.

#### • Greater Control of the Top Title Page

The three elements of the top title page are the values of the keys `university`, `title` and `author`. These keys are defined in the command `\DeclareDocInfo`. The pro option defines three commands `\universityLayout`, `\titleLayout`, and `\authorLayout`, to format these three keys in a variety of ways.

- A working example of the commands that follows can be found in `aebpro_ex2.tex`.

```
\universityLayout{<key-value_pairs>}
\titleLayout{<key-value_pairs>}
\authorLayout{<key-value_pairs>}
```

**Command Location:** Place these optional commands in the preamble.

**Key-Value Pairs:** Each of these commands have a number of key-value pairs. The first thirteen are the same that appear in the description of `\sectionLayout`, ‘[Section Headings](#)’ on page 50. The rest are unique to these three commands. In the descriptions below, the word ‘element’ refers to the values of the keys `university`, `title` and `author`.

1. `fontfamily`: Font family to use with this element, permitted values are `rmfamily`, `sffamily`, `ttfamily`.
2. `fontseries`: Font series to use, possible values are `bfseries` and `mdseries`.
3. `fontshape`: Font shape to use: `upshape`, `itshape`, `scshape`, `slshape`.
4. `fontsize`: Font size to use with this element, recognized values are `tiny`, `script-size`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge`, `Huge`.
5. `halign`: Alignment of this element within its enclosing `\parbox`, permissible values are `l` (left aligned), `c` (centered), `r` (right aligned). See [Example 2](#) for a visualization of the effects of the `halign` key.
6. `color`: The color of the section title, this can be any named color. The default is blue for `title`, and black otherwise.
7. `special`: Through this key, you can specify predefined layout for the title elements. Permissible values are `shadow`, `framebox`, `colorbox`, `fcolorbox`, `frameboxfit`, `colorboxfit`, `fcolorboxfit`, `custom`, `default`.

Custom titles can be created by specifying a value of `custom`. In this case, `Web` uses `\customUniversity`, `\customTitle` and `\customAuthor`. These are macros that take one argument, the code for designing the title. The title is referred to as `#1`. Depending on how these custom titles are defined, the other keys may not be obeyed.


8. `framecolor`: The color of the frame surrounding the subject when the `special` key has a value of `framebox`, `fcolorbox`, `frameboxfit` or `fcolorboxfit`.
9. `bgcolor`: The background (fill color) of the box enclosing this element, when `special` has a value of `colorbox`, `fcolorbox`, `colorboxfit` or `fcolorboxfit`.
10. `shadowcolor`: The color of the shadow, when `special` has a value of `shadow`.
11. `beforeskip`: The amount of skip before the title element.
12. `afterskip`: The amount of skip after the title element.
13. `usefont`: Through this key it is possible to specify an arbitrary font and font size. The key takes five parameters, for example,

```
usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}
```

The first four are the arguments of the  $\text{\LaTeX}$ 's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the  $\text{\LaTeX}$ 's `\fontsize`, size and baselineskip.

If the fifth parameter is empty, no font size is specified, the current default sizes are used.

14. `hproportion`: Each of the three elements (university, title, author) lie in their own `\parbox`, the width of this box is determined by the value of this key, as a proportion of the total `\linewidth`. The default for all three is `.7`. This value can be set to get more or less “natural” line breaks, without having to insert a new line with a `'\'`. See [Example 2](#) for a visualization of the effects of the `hproportion` key.
15. `xhalign`: The `\parbox` of each of the three elements are also placed in a `\makebox`, additional control over positioning can be had by setting this key, which sets the positioning parameter of `\makebox`. The default value for `xhalign` is `c`, the element is centered. See [Example 2](#) for a visualization of the effects of the `xhalign` key.

 The demo file `web_pro` illustrates this command.

 The file `webpro_titlepg` shows the structure of the title page.

Just above the **Title Top** is a skip that can be used to push the top down. The default value of this skip is `0pt`, but this can be changed through `\aboveTopTitleSkip`.

```
\aboveTopTitleSkip{<skip>}
```

There is one other title page parameter that effects the layout.

```
\topTitlePageProportion{<0..1>}
```

**Command Location:** Place this (optional) command in the preamble.

**Parameter Description:** The top part of the title page is enclosed in a big `\parbox` with depth set to a proportion of `\textheight`. `\topTitlePageProportion` is used to set this proportion. The argument of this command should be a number between 0 and 1, obviously a value of 0 makes no sense. The default value is set by the Web package to `.33`, i.e., the default is `\topTitlePageProportion{.33}`.

**Example 2:** [Figure 4](#) gives a representation of the page layout of the title page. The big `\parbox` of depth equal to the proportion of `\textheight` set by `\topTitlePageProportion` is shown as a blue box. Each of the three top title elements are enclosed in a `\makebox`, shown in yellow. Inside this `\makebox`, the top title elements are placed in a `\parbox`, shown in gray. The image shown in [Figure 4](#) came about as a result of the following commands in the preamble:

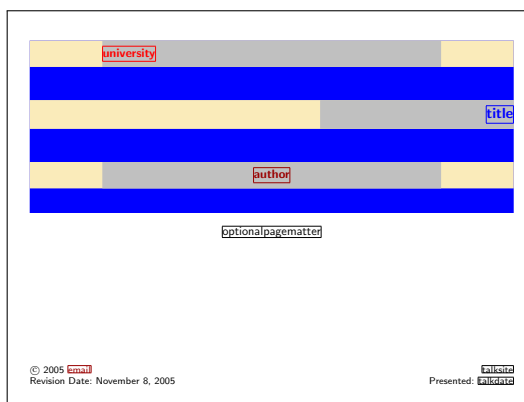


Figure 4: The Title Page Layout Structure

```
\topTitlePageProportion{.5}
\universityLayout{halign=l,color=red}
\titleLayout{halign=r,xhalign=r,hproportion=.4}
\authorLayout{color=webbrown}
```

In [Figure 4](#), notice that the `university` is left aligned within its `\parbox`. For `title`, the proportion is changed from the default of `.7` to `.4`, this is manifested by the shorter gray box (which represents the underlying `\parbox`); `halign` and `xhalign` are both set to `r`, so the title appears right aligned within its `\parbox`, and the `\parbox` is right aligned within its `\makebox`, understand? Finally, for the `author` key, we change only the color. Swave!

#### • Greater Control over Title Page Trailer

The `pro` option defines the command `\DesignTitlePageTrailer` that takes a number of key-value pairs to describe the title page trailer. The use of this command is not needed unless you want something in the trailer other than the default information.

```
\DesignTitlePageTrailer{<key_values>}
```

**Key-Value Pairs:** The key-value pairs are described below:

1. `ul`: The text in the upper left corner of the trailer, the default is

```
\web@copyright\ \web@copyright@symbol\
\webcopyrightyears\ \thewebemail
```

2. `ur`: The text in the upper right corner of the trailer, the default is `\aeb@talksite`, as defined by the `\talksite` command or the `talksite` key of the `\DeclareDocInfo` command.
3. `ll`: The text in the lower left corner of the trailer, the default is `\aeb@Prepared`, as defined by the `\prepared` command or the `prepared` key of the `\DeclareDocInfo` command.


4. `lr`: The text in the lower right corner of the trailer, the default is `\webversion`, as defined by the `\version` command, by the `version` key, or by the `talkdate` key. If `version` and `talkdate` both have values, `talkdate` is used.
5. `textColor`: The default color of the text. The default is black.
6. `bgColor`: The background of the trailer. This is a named color. The default is there is no color.
7. `borderColor`: The border color of the trailer. This is a named color. If specified, then `bgColor` must have a named color too, otherwise, this key does nothing. The default is that there is no border color.
8. `fboxsep`: The previous two keys use `\colorbox` and `\fcolorbox`. You can adjust the parameters `\fboxsep` through this key. The default is 3pt.
9. `fboxrule`: The width of the rule created when there are values for `bgColor` and for `borderColor`. This key sets the value of `\fboxrule`. The default is .4pt.
10. `graphic`: The value of this key is the base name of a graphic to be used as the background of the trailer. The graphic is rescaled to the width and the height of the trailer. It also reduces `\linewidth` so that the text is `\fboxsep` from the borders. The default is no graphic. If the key `graphic` is specified, then `bgColor` and `borderColor` are ignored.

If the `graphicxbox` package is loaded, AeB uses this package to place the graphic. In this case, the `borderColor` key is obeyed; whereas, it is not obeyed without the `graphicxbox` package.

11. `namedgraphic`: This key requires the use of `graphicxsp`, and Acrobat Distiller. Use `graphicxsp` to embed a graphic and give it a symbolic name `<mygraphic>`. If you then say `namedgraphic=<mygraphic>`, the graphic is used without re-embedding the graphic. The advantage of this method is that the graphic can be used and re-used in the document, without significantly increasing file size.

If the `graphicxbox` package is loaded, AeB uses this package to place the named graphic. In this case, the `borderColor` key is obeyed; whereas, it is not obeyed without the `graphicxbox` package.

12. `raise`: A key that takes a dimension as its value. The trailer will be raised or lowered by the amount specified. For example, `raise=-10pt` lowers the trailer 10pt from its natural position.
13. `fontSize`: Set the font size of the text, the default is `\footnotesize`. This key simply redefines `\trailerFontSize`.
14. `formatting`: Any additional formatting you wish that does not cause trouble; for example, `formatting=\bfseries`.

 The demo file `web_pro` illustrates this command.

#### 14.4. The Table of Contents

The pro option also defines `\tocLayout` for controlling the title of the table of contents.

```
\tocLayout{<key-value_pairs>}
```

The key-value pairs are the same as layout commands for the section titles, see ‘[Section Headings](#)’ on page 50.

**Command Location:** Place this command in the preamble.


```
\selectTocDings{<key-value_pairs>}
```

When the `\noSectionNumbers` command is used (‘[Section Headings](#)’ on page 50), the entries in the table of contents have no section number. You can define a ding for each of the three levels of the table of contents with this command. The key-value entries in the `\selectTocDings` command are ignored if `\noSectionNumbers` is not in effect.

**Command Location:** Place this (optional) command in the preamble.

**Key-Value Pairs:** The following are the recognized key-value pairs:

1. `dDing`: The ding for the first (section) level entry.  
Example, `dDing=\ding{072}`.
2. `ddDing`: The ding for the second (subsection) level entry.
3. `dddDing`: The ding for the third (subsubsection) level entry.
4. `dDingColor`: The (named) color for the ding of the first level entry. The default is red.
5. `ddDingColor`: The (named) color for the ding of the first level entry. The default is blue.
6. `dddDingColor`: The (named) color for the ding of the first level entry. The default is webgreen.

 The demo file `web_pro` illustrates this command.

#### 14.5. Selecting Colors

The `\selectColors` command can be used to set the color of most all elements of the document that have a color attribute.


```
\selectColors{<key-values>}
```

**Command Location:** There are no restrictions on this command. In the preamble, it will set the defaults for the entire document.

**Key-Value Pairs:** Each of the following keys take a named color. Many of these keys have a command interface as well, these are also noted.

1. `universityColor`: The color of the value of the `\university` declaration (refer to page 37). The value of `\university` appears on the title page. The default is blue.
2. `titleColor`: The color of the value of the `\title` declaration (see page 37). The value of `\title` appears on the title page. The default is black.
3. `authorColor`: The color of the value of the `\author` declaration (see page 37). The value of `\author` appears on the title page. The default is black.
4. `textBgColor`: The background color of the text screen. The default is white.
5. `panelBgColor`: The background color of the navigation panel. The default is white.
6. `urlColor`: The color of an URL link. The default is webbrown.
7. `linkColor`: The color of a link. The default is webgreen.
8. `fileColor`: The color of a link to a local file. The default is webbrown.

There are several other colors as well, these can be set through their interface, see [Section 14.6](#), page 50, for a discussion of the commands `\sectionLayout`, `\subsectionLayout` and `\subsubsectionLayout`.

 The demo file `web_pro` illustrates this command.

#### 14.6. Section Headings

The Web package uses the section heading commands `\section`, `\subsection`, and `\subsubsection`, but the `pro` option modifies their definitions so that the document author can easily design how the section titles look.

```
\noSectionNumbers
```

The command `\noSectionNumbers` causes Web to remove the usual numbering system for a  $\text{\LaTeX}$  document. This may be useful when preparing a document for presentation, and a numbering system is not needed. The default is to use section numbers.

**Command Location:** The use of this command is restricted to the preamble. The decision must be made for the whole document at the beginning of the document.

```
\sectionLayout{<key-values>}
\subsectionLayout{<key-values>}
\subsubsectionLayout{<key-values>}
```

**Command Location:** No restriction on the use of this command. In the preamble, you set the layout for sections for the whole document. In the body of the document, changes occur immediately starting with the next relevant section.

**Key-Value Pairs:** Each of these takes the same key-value pairs.

1. `fontfamily`: Font family to use for section titles, permitted values are `rmfamily`, `sffamily`, `ttfamily`.
2. `fontseries`: Font series to use for the section title, values are `bfseries` and `mdseries`.
3. `fontshape`: Font shapes to use for the section title, values are `upshape`, `itshape`, `scshape`, `slshape`.
4. `fontsize`: Font size of the section title, permissible values are `tiny`, `scriptsize`, `footnotesize`, `small`, `normalsize`, `large`, `Large`, `LARGE`, `huge`, `Huge`.
5. `halign`: Alignment of the section title, permissible values are `l` (left aligned), `c` (centered), `r` (right aligned).
6. `ding`: You can specify a ding to display where the section number is usually placed; e.g., `ding=\ding{066}`.
7. `color`: The color of the section title, this can be any named color. The default is blue.
8. `special`: Through this key, you can specify predefined layout for the section titles. Permissible values are `shadow`, `framebox`, `colorbox`, `fcolorbox`, `frameboxfit`, `colorboxfit`, `fcolorboxfit`, `custom`, `default`.

Custom section heads can be used by specifying a value of `custom`. In this case, Web uses the values of `\customSecHead`, `\customSubsecHead` and `\customSubsubsecHead`. Each of these macros take one argument, the code for designing the title. The title is referred to as #1. Depending on how these custom section titles are defined, the other keys may not be obeyed.

9. `framecolor`: The color of the frame surrounding the subject when the `special` key has a value of `framebox`, `fcolorbox`, `frameboxfit` or `fcolorboxfit`.
10. `bgcolor`: The background (fill color) of the box enclosing the section title, when `special` has a value of `colorbox`, `fcolorbox`, `colorboxfit` or `fcolorboxfit`.
11. `shadowcolor`: The color of the shadow, when `special` has a value of `shadow`.
12. `beforeskip`: The amount of skip before the section title.
13. `afterskip`: The amount of skip after the section title.
14. `usefont`: Through this key it is possible to specify an arbitrary font and font size. The key takes five parameters, for example,

```
usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}
```

The first four are the arguments of the  $\text{\TeX}$ 's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the  $\text{\TeX}$ 's `\fontsize`, size and baselineskip.

If the fifth parameter is empty, no font size is specified, the current default sizes are used.

15. `numdingcolor`: The color of the section number or ding, if defined. The default is blue.
16. `reset`: This key attempts to reset changes to their defaults. Permissible values are `font` (to reset the font changes only), or `all` (to reset all the keys).

**Example 3:** The following example sets the section title with shadows: text is red and shadow is blue. We center the title horizontally, and specify an after skip of 12pt.

```
\sectionLayout{%
  afterskip=12pt,
  align=c,
  color=red,
  shadowcolor=blue,
  special=shadow
}
```

Similarly for `\subsectionLayout` and `\subsubsectionLayout`.

```
\shadowoffset{<length>}
\shadowvoffset{<length>}
```

When `special = shadow`, shadowing is obtained by typesetting the text twice with different colors. The amount of horizontal and vertical offset are controlled by these two commands. The defaults are `\shadowoffset{.2ex}` and `\shadowvoffset{-.2ex}`.

The interface for writing custom section layouts are the following three commands.

```
\customSecHead{<tex_code>}
\customSubsecHead{<tex_code>}
\customSubsubsecHead{<tex_code>}
```

**Command Location:** Use in the preamble or before a section head.

**Parameter Description:** These three commands are used to write custom section, subsection and subsubsection layouts. Here, the parameter `<tex_code>` is  $\TeX$  code for laying out the section titles, and should use `#1` to represent the section title.


**Example 4:** The following definitions can appear anywhere, they are global unless appearing in a group. This puts a colored box around the section title, which is assumed to appear in one line. We use the internal color names for the `color` (`\aeb@sectioncolor`) and `bgcolor` (`\aeb@sectionbgcolor`) keys, in this way, the title will obey the values of these keys. This particular custom head obeys the `align` key as well.

```

\makeatletter
\customSecHead{\vbox{\colorbox{\aeb@sectionbgcolor}
  {\color{\aeb@sectioncolor}#1}}}
\sectionLayout{%
  special=custom,
  halign=l,
  bgcolor=red,
  color=white
}
\makeatother

```

The above code is a simplified version of the `colorboxfit`, a value of the `special` key.

 The demo file `web_pro` illustrates these commands.

## 15. On Parts, Chapters, and Final Dots

The `Web` package has several commands for customizing the `\part` and `\chapter` commands.

### 15.1. The `\part` command

The appearance of the part title can be controlled by `\formatPartTitle`.

```

\formatPartTitle{<tex_code>}
\noPartNumbers

```

The argument of `<tex_code>` consists of formatting commands; within the argument `#1` represents the title. The command `\noPartNumbers` is a convenience macro to turn off all part number, if you want to create a document that has no part numbers, such as this document.

The default argument for `\formatPartTitle`

```

\large\bfseries
\ifnum \c@secnumdepth >\m@ne
  \partname~\thepart:\space
\fi#1

```

This typesets as **Part I: The Web Package**, for example. You can modify the basic definition to introduce colors, fonts, sizes of your choice, and even special effects; for example, the declaration

```

\formatPartTitle{\fbox{\large\bfseries
\ifnum \c@secnumdepth >\m@ne
  \partname~\thepart:\space
\fi#1}}

```

gives a part title like this, **Part I: The Web Package**

A companion comment for changing the format of a part title, is `\tocPartTitle`, which modifies how the part title appears in the table of contents.

```
\tocPartTitle{<tex_code>}
\tocPartTitle{\thepart\hspace{1em}#1}
```

Again, within the argument of `\tocPartTitle`, `#1` refers to the text of the part title. The default definition is given above also.

► Finer control over the formatting of the `\part` command can be obtained by redefining `\web@makeparthead` or `\web@makesparthead`, the latter one for `\part*`.

## 15.2. The `\chapter` command

The `\chapter` command is available in the `book` and `report` classes. There are several defined in the `Web` package for getting a little control over these.

```
\formatChapterNumber{<tex_code>}
\formatChapterTitle{<tex_code>}
```

The argument of `<tex_code>` consists of formatting commands; within the argument `#1` represents the title.

The default definitions for these are

```
\formatChapterNumber{\large\bfseries
  \@chapapp\space\thechapter\par\nobreak}
\formatChapterTitle{%
  \interlinepenalty\@M
  \noindent\hspace{1em}\large\bfseries#1\par\nobreak
}
```

These are slightly modified versions of what is found in `book.cls`.

► A finer control can be obtained over the `\chapter` command formatting by redefining `\web@makechapterhead` or `\web@makeschapterhead`, the latter one is used for `\chapter*`.

## 15.3. The `\noFinalDot` Command

The default behavior for the `Web` package is to place a dot (.) at the end of the section number in both the body of the document and in the table of contents. Thus, this section is numbered as 15.3. (note the final dot following the section number). This section will appear in the table of contents the same way. The default behavior of all (that I know of) standard  $\text{\LaTeX}$  classes is not to place this final dot (.) at the end of the section number. To revert to this behavior, use the `\noFinalDot` command in the preamble.

## The Exerquiz Package

The `exerquiz` package provides environments for creating the following interactive elements in a PDF file.

- The `exercise` Environment: Macros for creating on-line exercises.
- The `shortquiz` Environment: Macros for creating interactive quizzes with immediate feedback.
- `shortquiz` with Solutions: Macros for creating quizzes with immediate feedback and a link to the solutions to the quizzes.
- The `quiz` Environment: Macros for creating quizzes graded by JavaScript, with an option to have the quizzes corrected using JavaScript.

In each of the quiz environments, you can pose multiple choice, math fill-in, or text fill-in questions.

The `exerquiz` provides the above listed environments for the `dvipsone`, `dvips`, `textures`, `pdftex` and `dvipdfm` options.

There are options for reformatting the exercises to a print format, for excluding the solutions to the exercises, for writing the solutions to the exercises so they follow the question, and for different languages, and much more.

The `exerquiz` also allows you to rearrange the order and location of the solutions to the exercises and quizzes, to redefine many running headers, to customize the exercises and quizzes, and to use the `exercise` environment to create a new environment with its own counter, or with no counter at all.

All the above mentioned macros and the options of the package are discussed in this section.

### 16. Exerquiz and Acrobat JavaScript

`Exerquiz` now uses the `insdljs` Package to insert Document level JavaScripts into the PDF file. The quizzes created using the `shortquiz` or `quiz` environment are graded, marked and scored using these inserted JavaScript functions.

Because the package `insdljs` is already loaded, it is easy for the document author to develop JavaScripts that can be called from the standard `Exerquiz` commands. The ability to write JavaScript, therefore, right in the  $\LaTeX$  document gives a unique programming flair to `Exerquiz`.

### 17. Package Requirements

The `exerquiz` package is independent of the `web` package; however, `exerquiz` utilizes `hyperref` just as `web` does. Use the latest version of `hyperref`. In addition to the `color` package, also used by `web`, `exerquiz` also uses the `verbatim` package. This is used to write verbatim solutions to exercises and quizzes to certain auxiliary files.

Results from the quizzes created by the `shortquiz` and `quiz` environments are evaluated using document level JavaScripts. These JavaScripts are inserted into the final PDF file using the `insdljs` package. This package makes it easy for the package writer or document author to write JavaScripts.

The `exerquiz` package uses *form features* of PDF that web does not use. For the interactive features to properly work, use [Adobe Reader 5.0](#) or higher.

## 18. Basic Package Options

Place in the preamble of your document

```
\usepackage{exerquiz}
```

- ▶ Use `exerquiz` with the `web` package:

```
\usepackage[<driver_option>,<more_options>]{web}
\usepackage[<options>]{exerquiz}
```

A complete list of the options recognized by `exerquiz` can be found in [Section 30](#), page 154; they are also discussed below.

The driver options for the `web` package are `dvipsone`, `dvips`, `pdftex`, `dvipdfm` and `textures`. No driver option with `exerquiz` is needed if you are using the `web` package, `exerquiz` inherits the driver option from the `web` package.

For the `dvipdfm` option to work properly you will need `dvipdfm`, version 0.12.7b or later, and `hyperref`, version 6.68a or later.

- ▶ Use `hyperref` and `exerquiz` with either `dvipsone` or `dvips`:

```
\usepackage[<driver_options>,<more_options>]{hyperref}
\usepackage{exerquiz}
```

Permissible driver options are `dvipsone` and `dvips`.

- ▶ Use `hyperref` and `exerquiz` with `pdftex` or `dvipdfm`

```
\usepackage[<driver_options>,<more_options>]{hyperref}
\usepackage[<driver_option>]{exerquiz}
```

See the next few paragraphs for more details.

### 18.1. The `pdftex` Option

The `exerquiz` package is independent of the `web` package. Therefore, you can create your own page layout package and use `exerquiz` to help you create exercises and quizzes. Of course, `hyperref` must be used.

Should you want to use the `exerquiz` package using `pdftex` without the `web` package, use the `pdftex` option:

```
\usepackage[pdftex,<more options>]{hyperref}
\usepackage[pdftex]{exerquiz}
```

In particular, `pdfscreen`<sup>8</sup>, a screen design package written for `pdftex` by C. V. Radhakrishnan, has been tested and works correctly with `exerquiz`. For example,

<sup>8</sup>CTAN:macros/latex/contrib/supported/pdfscreen

```
\usepackage[screen,article,sidebar]{pdfscreen}
\usepackage[pdftex]{exerquiz}
```

See the sample file `eq_pdfs.tex` already set up for use with `pdfscreen`, obtained by downloading the zipped file [eq\\_pdfs.zip](#).

### 18.2. The `dvipdfm` Option

Should you want to use the `exerquiz` package without the `web` package, in this case, the usage is

```
\usepackage[dvipdfm,<more_options>]{hyperref}
\usepackage[dvipdfm]{exerquiz}
```

► **Important Note:** *Only* the exercise environment (this is the material described in [Section 19](#)) is supported by these two options. None of the quiz environments can be used with these two options at this time.  $\TeX$  users need to use the `dvipsone` option if the quiz environment is needed.

### 18.3. The Language Option

The [language option](#), available in the `web` package, can be invoked even when the `web` package is not used.<sup>9</sup> Currently, `dutch`, `french`, `german`, `italian`, `norsk`, `russian`, `spanish`, `polish`, `finnish`, `czech`, `catalan`, `brazil`, and `turkish` are the supported options. For example, with `hyperref`, you could use:

```
\usepackage[<driver_option>,<more_options>]{hyperref}
\usepackage[<driver_option>,french]{exerquiz}
```

`<driver_option>` is any of the drivers: `dvipsone`, `dvips`, `pdftex`, or `dvipdfm`. *Note:* the `<driver_option>` is not needed with the `exerquiz` package with `dvipsone` or `dvips`.

► **Unicode required for turkish and russian options:** When using the `turkish` or the `russian` option, without using the `Web` package, use the `unicode` option for `hyperref`, like so,<sup>10</sup>

```
\usepackage[<driver_option>,<more_options>,unicode]{hyperref}
\usepackage[<driver_option>,turkish]{exerquiz}
or
\usepackage[<driver_option>,russian]{exerquiz}
```

### 18.4. The `forpaper` and `forcolorpaper` Options

The `forpaper` or `forcolorpaper` option, also available in the `web` package, is needed in the `exerquiz` package if you are using `exerquiz` without `web`. The option is invoked in the usual way.

<sup>9</sup>Otherwise, the language option is introduced as an option of the `web` package.

<sup>10</sup>The `Web` package passes the `unicode` option to `hyperref` when using the `turkish` or `russian` option through the `Web` option list so there is no need to explicitly use the `unicode` option.

```
\usepackage[<options>]{hyperref} % or pdfscreen
\usepackage[forpaper]{exerquiz}
```

See the discussion of the `forpaper` on page 31 given earlier.

### 18.5. The preview Option

The `exerquiz` package can generate a large number of form fields: buttons, check boxes, radio buttons, and text fields. These are PDF objects and cannot be seen in a dvi previewer. By using the `preview` option, the bounding rectangles of the form objects are surrounded with rules, which outline the form fields and make their positions visible.

This option may help you to fine tune the positions of the form fields. The option is for developmental use only. When you are satisfied with the positioning and are ready to publish, remove this option.

► This option is not useful with the `pdftex` option, as `pdftex` does not (normally) produce a dvi file.

### 18.6. The nodljs Option

If you are creating a document that is meant to be printed or your document only has exercises and solutions in it (which do not require JavaScript), the size of the document can be reduced significantly by using the `nodljs` option. This option is just passed on to the `insdljs` package.

### 18.7. The exercisonly Option

If the document author only uses the `exercise` environment, then all the document level JavaScripts of `exerquiz` are not needed. Use either one of these two equivalent options to exclude the insertion of the JavaScripts.

This is a convenience option that simply calls the `nodljs` option described above.

### 18.8. The debug Option

Developing JavaScript functions can be tricky. Quite often, it is useful to insert some code lines that will help you in debugging a particular function or a set of functions. For example, you might want to verify that the parameters being passed to a function are the correct ones, or that the return value is correct. You can have Acrobat write the values to its console like so:

```
console.println("Function myFunc");
console.println("Parameters: x = " x + ", y = " + y );
console.println("Return Value: retnValue = " + retnValue);
```

In the above code, I have used the `console.println()` method, which is only available in the Acrobat application, not the Reader. For the Reader, one could use `app.alert()`, but this method is not well-suited for monitoring values of a large number variables as the script executes. If you don't have the full Acrobat, the `debug` option will not be useful.

Exerquiz just passes this option on to the `insdljs` package. Additional details on the debug option can be found there. Within the `insDLJS` environment, you can place debugging code lines as follows:

```
function myFunc(x,y)
{
  retnValue = x + y;
  \db console.println("Function myFunc");\db%
  \db console.println("Parameters: x = " x + ", y = " + y );\db%
  \db console.println("Return Value: retnValue = " + retnValue);\db%
  return retnValue;
}
```

Any line that begins with `\db` and ends with `\db` is a debugging line. These lines will be included if the debug option is taken; otherwise they are removed. The `'%'`, is the comment character within the `insDLJS` environment, and prevents, in this case, the introduction of a carriage return.

### 18.9. The `allowrandomize` Option

Use this option to load in the macros to randomize the choices of a multiple choice question. See [Section 23, 'Randomizing the Multiple Choices'](#) on page 123 for details.

### 18.10. The `unicode` Option

This option is passed to `hyperref`, and is placed in `Web` as a convenience. If `eforms` is subsequently loaded, `eforms` (and `exerquiz`) will accept  $\LaTeX$  markup in the optional argument of the form fields. See the [eforms manual](#) for details.

### 18.11. The `useui` Option

This option is passed to `eforms`, and is placed in `Web` as a convenience. If `eforms` is subsequently loaded, `eforms` loads the `xkeyval` package, and key-value pairs are defined for use in the optional argument of `form` and `link` commands. See the [eforms manual](#) for details.

### 18.12. The `usesumrytbls` Option

When this option is taken, the code for creating quiz summary tables is input. See [Section 24](#), page 126 for details.

## 19. The exercise Environment

The `exerquiz` package defines `exercise` and `solution` environments, the latter being nested inside the former. With these environments, you can create questions (exercises) with solutions. Solutions are written `verbatim` to the auxiliary file `\jobname.sol`, then input back in near the end of the document. A hypertext link is created to connect the exercise with the solution.



```
\end{solution}
\end{exercise}
```

See the demo file `webeqstst.tex` for a complete listing of this exercise.

► Questions and solutions are kept together *à la Knuth*. The solutions are written to the file `\jobname.sol` verbatim then input back using `\includeexersolutions`.

► You can redefine the counter to include the section number. For example, the code

```
\renewcommand{\theeqxno}{\thesection.\arabic{eqxno}}
```

can be placed in the preamble of your document. In this case, the above exercise would appear as Exercise 19.1.

► The usual cross-referencing mechanisms for  $\text{\LaTeX}$ , i.e., using `\ref` and `\pageref`, work as expected.

For example, the label `'\label{ex:int}'` was placed just after the `\begin{exercise}` on the previous page. Let us now reference Exercise 1, on page 60.

```
Let us now reference Exercise~\ref{ex:int},
on~\pageref{ex:int}.
```

Of course, the nicer looking variations can be done as well. For example, see [Exercise 1](#).

```
\hyperref[ex:int]{\textsf{Exercise~\ref*{ex:int}}}
```

The `*`-form of `\ref` was used to turn off the redundant link creation. (`hyperref` would normally make the `\ref` macro into a link.)

► An ‘Exercise’ that is also a hypertext link appears in the default color **green**; if an ‘Exercise’ is not a link, it appears in **blue**. (The word ‘Exercise’ is not a link if it is an **exercise with parts**, or if the **nosolutions** option is used. Finally, if the web option **forpaper** is used, color is turned off and ‘Exercise’ appears in black.)

► **Caveat:** There is one problem you might watch for. There is an optional argument to the `solution` environment. When  $\text{\LaTeX}$  searches the source looking for the optional parameter, which may not exist, it expands macros looking for a `'[`. This causes problem when you have a solution that begins with a math display environment and  $\text{\LaTeX}$  prematurely expands such an environment.

**Exercise 2.** Write an equation of a line that crosses the  $x$ - and  $y$ -axes at 1.

To prevent  $\text{\LaTeX}$  errors that will stop the compilation, just place a `\relax` prior to the math environment. The code for the previous exercise is

```
\begin{exercise}
Write an equation of a line that crosses
the  $x$ - and  $y$ -axes at 1.
\begin{solution}\relax
\begin{equation*}
\boxed{x+y=1}
```

```

\end{equation*}
\end{solution}
\end{exercise}

```

This is only necessary if the solution **does not** begin with text.

#### • Exercises with Parts: The `exercise*` Environment

Beginning with version 6.07, the `exercise*` environment is used to create exercises with multiple parts.<sup>11</sup>

A companion environment to `exercise*` is the `parts` environment, use use to enclose the multiple parts of the question. The `parts` environment takes one optional argument, the number of columns to be used. The argument must be a positive integer greater than 1, in this case, a tabular environment is used, with the number of columns equal to the specified argument. If no optional parameter is given, then a list environment is used.

```

\begin{exercise*}
Preamble for your multi-parted question.
\begin{parts}                % begin listing of the parts
\item First question.
\begin{solution}
Solution to first question.
\end{solution}
...
...
\item Final question.
\begin{solution}
Solution to the final question.
\end{solution}
\end{parts}                % end listing of parts
\end{exercise*}

```

The following exercise illustrates this option. This example appears in `webeqtst.tex`.

**Exercise 3.** Suppose a particle is moving along the  $s$ -axis, and that its position at any time  $t$  is given by  $s = t^2 - 5t + 1$ .

- (a) Find the velocity,  $v$ , of the particle at any time  $t$ .
- (b) Find the acceleration,  $a$ , of the particle at any time  $t$ .

There is also an option for listing multipart questions in tabular form.

**Exercise 4.** Simplify each of the following expressions in the complex number system. *Note:*  $\bar{z}$  is the conjugate of  $z$ ;  $\operatorname{Re} z$  is the real part of  $z$  and  $\operatorname{Im} z$  is the imaginary part of  $z$ .

- |                   |           |
|-------------------|-----------|
| (a) $i^2$         | (b) $i^3$ |
| (c) $z + \bar{z}$ | (d) $1/z$ |

<sup>11</sup>For users of `exerquiz`, this is equivalent to the `*`-option with the `exercise` environment. The `exercise*` environment is the preferred form but the `*`-option works as before.

The syntax is the same as an exercise with multipart.

```
\begin{exercise*} % <- star indicates multipart
Simplify each...
\begin{parts}[2] % <- optional argument indicates tabular
\item  $i^2$ 
\begin{solution}  $i^2 = -1$  \end{solution}
&
\item  $i^3$  \begin{solution}  $i^3 = i i^2 = -i$  \end{solution}
\\
\item  $z + \bar{z}$ 
\begin{solution}  $z + \bar{z} = \operatorname{Re} z$  \end{solution}
&
...
\end{solution}
\end{parts}
\end{exercise*}
```

► This problem style does not obey the `solutionsafter` option. (See the section entitled ‘[The solutionsafter Option](#)’ on page 66).

✂ The sample file `webeqst.tex` contains this particular example.

## 19.2. Options of the exercise Environment

### • Leaving Vertical Space instead of a Solution

The exercise environment can be used for test construction. Initially, you may want to pose a questions and leave space beneath for the student to write in an answer.

The solutions environment has an optional parameter for inserting a vertical space.

```
\begin{exercise}
This is the question.
\begin{solution}[1in] % <-- optional vertical skip
This is the solution.
\end{solution}
\end{exercise}
```

This vertical space only appears when the `nosolutions` option is in effect.

Within the context of test construction, write the test (including the solutions), then publish it with the `nosolutions` option (leaving vertical spaces as appropriate), then publish the key with the `solutionsafter` option.<sup>12</sup>

► The optional parameter for the solution is ignored for exercises with parts having a tabular format ([Example 4](#) is an example of a tabular multipart exercise).

<sup>12</sup>If `solutionsafter` and `nosolutions` both appear in the option list, `solutionsafter` overrides `nosolutions`.

### • Hiding some Solutions

A subset of the solutions can be hidden by using the ‘h’ option. This option is an option of the exercise environment, as well as an option of `\item`, when there is an exercise with parts. For example, the following code

```
\begin{exercise}[h] % <- hide solution
Give an example of a set that is \textit{clopen}.
\begin{solution}
The real number line is both closed and open in the
usual topology of the real line.
\end{solution}
\end{exercise}
```

yields the following exercise.

**Exercise 5.** Give an example of a set that is *clopen*.

Notice that there is no hypertext link to the solution; indeed, the solution was not even written to the `\jobname.sol` file.

The ‘h’ option works with exercises with parts as well. Just apply the ‘h’ option to the `\item`:

```
\begin{exercise*}
A particle has position  $s=t^2 - 5t + 1$  at time  $t$ .
\begin{parts}

\item Find the velocity,  $v$ , at time  $t$ .
\begin{solution}
 $v = 2t-5$ .
\end{solution}

% This solution will not be included in the solutions
% section at the end of the document.
\item[h] Find the acceleration,  $a$ , at time  $t$ .
\begin{solution}
 $a = 2$ .
\end{solution}
\end{parts}
\end{exercise*}
```

The results of this code follow:

**Exercise 6.** A particle has position  $s = t^2 - 5t + 1$  at time  $t$ .

- (a) Find the velocity,  $v$ , at time  $t$ .
- (b) Find the acceleration,  $a$ , at time  $t$ .

Part (a) is hypertext linked to its solution, whereas part (b) is blue, indicating there is no link there.

- ▶ Multipart exercises in the tabular format behave the same way; use `\item[h]` to “hide” a solution.
- ▶ There is also an ‘H’ option as well. Specifying ‘H’ also hides the solutions. See the next two sections for a discussion of revealing the solutions marked by either ‘h’ or ‘H’ to understand the distinction between the two.

#### • The `nohiddensolutions` Option

Hidden solutions can be included in the document by either removing the ‘h’ option everywhere and re- $\LaTeX$ ing, or by simply using the `nohiddensolutions` of `exerquiz`.

```
\usepackage[nohiddensolutions]{exerquiz}
```

This option overrides the local ‘h’ option throughout the document.

- ▶ When the `solutionsafter` option of `exerquiz` is invoked, the hidden solutions are also revealed. To keep the solutions hidden, in this case, you should use ‘H’ option instead of ‘h’. See the next section.

#### • The `noHiddensolutions` Option

In addition to the ‘h’, you can also use the ‘H’ option with exercises. The solution will be hidden with ‘H’, but will not be revealed when either the `nohiddensolutions` or the `solutionsafter` options are used.

The ‘H’ option can be overridden by using the `noHiddensolutions` of `exerquiz`.

```
\usepackage[noHiddensolutions]{exerquiz}
```

This option overrides the local ‘h’ option throughout the document.

#### • The `exercise environment Counter`

The counter for the `exercise` environment is `eqexno`, and will number your exercises consecutively throughout the document. Should you want the counter to be reset after each section, place in the preamble of your document the following lines:

```
\makeatletter
\@addtoreset{eqexno}{section}
\makeatother
```

#### • The `nosolutions` Option

Some educators may initially want to post a series of exercises on the Web without the solutions. Then, at a later date, repost the exercises with the solutions included. For this application there is the `nosolutions` option for the `exerquiz` package.

```
\documentclass{article}
\usepackage[pdfTeX]{web} % dvipsone, dvips or dvipdfm
\usepackage[nosolutions]{exerquiz}
```

For this kind of application, it might make sense to publish the exercises with the `forpaper` option.

### • The solutionsafter Option

For additional flexibility with how you want the solutions to the exercises presented, there is a `solutionsafter` option with `exerquiz`. Should you invoke this option,

```
\documentclass{article}
\usepackage[dvipson]{web} % dvips or pdftex
\usepackage[solutionsafter]{exerquiz}
```

the solutions to the exercises appear just *after* the exercise question. For example,

**Exercise 7.** Let  $V$  be a vector space, show that the zero vector,  $\mathbf{0}$ , is unique.

*Solution:* Let  $\mathbf{0}'$  be a vector that satisfies the axiom of being a zero of the vector space  $V$ . We want to show  $\mathbf{0} = \mathbf{0}'$ . Since  $\mathbf{0}$  is a zero, we have  $\mathbf{0} + \mathbf{0}' = \mathbf{0}'$ . But we are assuming  $\mathbf{0}'$  is a zero vector as well, hence,  $\mathbf{0}' + \mathbf{0} = \mathbf{0}$ . Finally,

$$\mathbf{0}' = \mathbf{0} + \mathbf{0}' = \mathbf{0}' + \mathbf{0} = \mathbf{0}$$

and this completes the proof.

Exercise 7

As you can see in the above example, the word *Solution:* is typeset at the beginning of the solutions. The command `\renameSolnAfterTo` can be used for conveniently changing the solution after label, for example, `\renameSolnAfterTo{\textbf{Proof:}}` changes the label to **Proof:**, and `\renameSolnAfterTo{}` produces no label. These changes will be local to the group in which they are made, or global if they are not made in a group.

The command `\resetSolnAfterToDefault` sets the label text back to the default. The default label is

```
\newcommand{\eq@exsolafterDefault}{\textit{Solution}:}
```

The option `solutionsafter` is global; all exercises are typeset this way—unless you change it within the document using the macros `\SolutionsAfter` and `\SolutionsAtEnd`. This manual was typeset without the `solutionsafter` option. The above example was typeset as follows:

```
\SolutionsAfter % show solution following exercise
\begin{exercise}
Let  $V$  be a vector space, show ...
\begin{solution}
.....
\end{solution}
\end{exercise}
\SolutionsAtEnd % turn back on solutions at of document
```

Normally, a typical document might have all solutions at the end of the document (the default behavior), or all solutions following each exercise (`solutionsafter` option). Mixtures of these two types can be obtained by using the two commands `\SolutionsAfter` and `\SolutionsAtEnd`.

This feature might be an easy way of typesetting examples. See the example in the section entitled '[Redesigning the exercise Environment](#)' on page 67.

- ▶ The `solutionsafter` option has no effect on multipart exercises in *tabular form*; I haven't been able to find a convenient way of displaying the solutions after the questions when the questions are in tabular form.

📄 See the file [webeqst.pdf](#) for examples.

- **Moving the Solution Set**

The solution set, by default, comes last in the file. You can move its positioning by including the command `\includeexersolutions` at any point *after* the last exercise. You'll note that I have moved the solutions in this file *before* the **References** section, as indicated, for example, by its position in the table of contents.

### 19.3. Redesigning the exercise Environment

You can customize the `exercise` environment to suit your own needs. To customize, you need to change some or all of the following six commands. In the listing below, the  $\LaTeX$  definition of each follows a short description.

1. `\exlabel`: This command expands to the name of the exercise label, the default string is 'Exercise'.

```
\newcommand\exlabel{Exercise}
```

2. `\exlabelformat`: Typesets the exercise label; use it to introduce additional type style such as boldface, italic, small caps, etc.

```
\newcommand\exlabelformat{%
  {\scshape\exlabel\ \theeqexno.}}
```

3. `\exlabelso1`: Expands to the name of the exercise label in the solutions section. Usually its value is the same as `\exlabel`.

```
\newcommand\exlabelso1{\exlabel}
```

4. `\exsllabelformat`: The format of the solutions label, `\bfseries\exlabel` is the default.

```
\newcommand\exsllabelformat
  {\noexpand\textbf{\exlabelso1\ \theeqexno.}}
```

5. `\xrtnlabelformat`: This is the label you click on to return from the solution of the exercise.

```
\newcommand\xrtnlabelformat{\exlabelso1\ \theeqexno}
```

6. `\exsectitle`: The section title of the solutions to the exercises.

```
\newcommand\exsectitle{Solutions to \exlabel s}
```

7. `\exsecrunhead`: The running header for the solution section for the exercises.

```
\newcommand\exsecrunhead{\exsectitle}
```

- ▶ The counter `eqexno` is used to count exercises. When an exercise environment starts, this counter is incremented. The value of this counter is used in the definitions of `\exlabelformat`, `\exsllabelformat` and `\xrtnlabelformat`. Still, the use

of `eqexno` is optional; for example, you might want to state a problem just as ‘Special Exercise’, without an associated exercise number.

Below is an example of redefining the `exercise` environment. We define a `problem` environment based on the `exercise` environment.

```
\newenvironment{problem}{%
\renewcommand\exlabel{Problem}
\renewcommand\exlabelformat{\textbf{\exlabel\ \theeqexno.}}
\renewcommand\exsllabelformat
{\noexpand\textbf{\exlabel\ \theeqexno}}
\renewcommand\extrnlabelformat{${\blacktriangleleft}$}
\renewcommand\exsecrunhead{\exsectitle}
\begin{exercise}}{\end{exercise}}
```

See any standard L<sup>A</sup>T<sub>E</sub>X reference on how to define a new environment, for example [3].

Here is an example of the new `problem` environment:

**Problem 8.** This is a question.

The code for this problem was simply:

```
\begin{problem}
This is a question.
\begin{solution}
This is the solution.
\end{solution}
\end{problem}
```

► Two of these commands, `\exsllabelformat` and `\extrnlabelformat`, must be handled with special care. Formatting such as `\textbf` or `\scseries` must be preceded by a `\noexpand`. These commands are written to a file, and must be prevented from expanding.

When you use the `exercise` environment, the counter `eqexno` is automatically incremented by default. The `exercise` does have an optional argument for inserting your own counter.

```
\begin{exercise}[<ctr>]
.....
\end{exercise}
```

Where `<ctr>` is a counter already defined. This option is useful if you want to use the `exercise` environment to create a new environment with its own numbering scheme, as the following example illustrates.

This example demonstrates how to define an `example` environment with its own counter. For examples, we don’t want the solutions to appear at the end of the file, so we’ll use the commands `\SolutionsAfter` and `\SolutionsAtEnd`. All changes are local.

```

% put a counter in preamble
\newcounter{exampleno}
\newenvironment{example}{%
\renewcommand\exlabel{Example}
\renewcommand\exlabelformat
  {\textbf{\exlabel\ \theexampleno.}}
\renewcommand\extrnlabelformat{${\square$}
\SolutionsAfter
\begin{exercise}[exampleno]}%
{\end{exercise}
\SolutionsAtEnd}

```

Now we simply type

```

\begin{example}
What is  $2+2$ ?
\begin{solution}
It is well known that  $2+2=4$ .
\end{solution}
\end{example}

```

to obtain

**Example 1.** What is  $2 + 2$ ?

*Solution:* It is well known that  $2 + 2 = 4$ . □

The changes are local to the new `example` environment. If we have another exercise, we get a correctly numbered exercise.

**Exercise 9.** What is  $2 + 2$ ?

► The command `\exsolafter` typesets the solution label to the exercise in the case the `solutionsafter` option is in effect. The default definition of `\exsolafter` is `\textit{Solution}:`. You can redefine it as follows:

```
\renewcommand\exsolafter{\textit{L"osung}:}
```

The command `\renameSolnAfterTo` is a convenience macro for changing this label. We can redefine the label above like so: This redefinition yields:

```
\renameSolnAfterTo{\textit{L"osung}:}
```

**Example 2.** What is  $2 + 2$ ?

*Lösung:* It is well known that  $2 + 2 = 4$ . □

► There is a special option to the exercise environment as well,

```

\begin{exercise}[0]
.....
\end{exercise}

```

When the optional argument is 0 rather than a counter. In this case, no counter is associated with the environment. For example,

```
\newenvironment{project}{%
\renewcommand\exlabel{Project}
\renewcommand\exlabelformat{\textbf{\exlabel. }}
\renewcommand\exsllabelformat
{\noexpand\textbf{\exlabel\ Hint:}}
\renewcommand\exrtnlabelformat{${\blacktriangleleft}$}
\begin{exercise}[0]}%
{\end{exercise}}
```

Thus, we obtain,

**Project.** Find a shorter proof of Fermat's Last Theorem. Do not look at the project hints until you have finished the project.

The code:

```
\begin{project}
Find a shorter proof of \textsf{Fermat's Last Theorem}. Do not
look at the project hints until you have finished the project.
\begin{solution}
There, you didn't need my help after all.
\end{solution}
\end{project}
```

Note that the solutions are typeset at the end of the file in the 'Solutions to Exercises' section. At this time, there is no feature for sorting out these different types of environments; they are all `exercise` environments, which is what they are.

## 20. The `shortquiz` Environment

The `shortquiz` environment is used to create multiple choice question and math/text fill-in questions with immediate response. The discussion of math and text fill-in questions is post-phoned to [Section 22](#), entitled [Objective Style Questions](#). The environment allows redefinition to customize the look of your quizzes. (See '[Redesigning the short quiz Environment](#)' on page 79.)

### 20.1. Basic Usage

The syntax for the environment (tabular version) is as follows:

```
\begin{shortquiz}                % begin shortquiz
...Question goes here...
\begin{answers}{num_cols}        % begin proposed answers
...
\Ans0 <an incorrect answer> &   % a wrong answer
...
```

```

\Ans1 <a correct answer> & % the right answer
...
\end{answers} % end listing of answers
\end{shortquiz} % end shortquiz

```

The parameter `num_cols` is the number of columns you want to typeset for your multiple choice responses. The environment sets up a `tabular` environment if `num_cols` is greater than 1, and a `list` environment if `num_cols` is 1.

This type of quiz is suitable for asking a short series of question of the reader, perhaps after explaining some concept. Quizzes can be used to direct the reader's attention to an important point.

► Here is an example of the `shortquiz` environment. Responses are graded without comment using JavaScript.

**Quiz** Which of the following is the  $\frac{d}{dx}\sin(x^3)$ ?

- (a)  $\sin(3x^2)$       (b)  $\cos(x^3)$       (c)  $3x^2 \cos(x^3)$       (d)  $3x^2 \cos(3x^2)$

The verbatim listing follows:

```

\begin{shortquiz} % begin shortquiz environment
Which of the following is the  $\frac{d}{dx}\{\sin(x^3)\}$ ?
\begin{answers}{4} % 4 columns of answers
\Ans0  $\sin(3x^2)$  & % \Ans0 is a false answer
\Ans0  $\cos(x^3)$  &
\Ans1  $3x^2\cos(x^3)$  & % \Ans1 is the correct answer
\Ans0  $3x^2\cos(3x^2)$ 
\end{answers} % end answers environment
\end{shortquiz} % end shortquiz environment

```

If `num_cols` is greater than 1, the `answers` sets up a `tabular` environment, `p{<width>}` to sets up the column widths. The `\parboxes` are typeset ragged right.

► Below is a two-column example in which the posed alternatives are rather long. The `answers` environment produces is a nicely aligned set of paragraphs.

**Quiz** Which of the following best describes Augustin Cauchy?

- (a) He developed the Calculus while his University was closed for the plague.      (b) Given credit for first using the functional notation  $f(x)$ .  
(c) He created the “bell-shaped curve” and first used the method of least squares.      (d) He first formulated a precise definition of the limit and continuity of a function.  
(e) Gave a rigorous definition of the definite integral—an integral that now bears his name.      (f) His notation for the derivative and the integral is used even to this day.

Here is the same example in which the `num_cols` is set to 1; in this case, a `list` environment is used.

**Quiz** Which of the following best describes Augustin Cauchy?

- (a) He developed the Calculus while his University was closed for the plague.
- (b) Given credit for first using the functional notation  $f(x)$ .
- (c) He created the “bell-shaped curve” and first used the method of least squares.
- (d) He first formulated a precise definition of the limit and continuity of a function.
- (e) Gave a rigorous definition of the definite integral—an integral that now bears his name.
- (f) His notation for the derivative and the integral is used even to this day.

✂ See the sample file `webeqtst.tex` for examples.

► The space between the label and the text of a multiple choice question can be adjusted using two commands, `\setMClabelsep` and `\resetMClabelsep`. The default horizontal space between the label and text is a hard space ‘\ ’. The first command takes as its argument anything that creates a horizontal white space, for example, `\setMClabelsep{\quad}` or `\setMClabelsep{\kern20pt}`. If `\setMClabelsep` is set in a group, its value will revert to the default on exit from the group. The second command `\resetMClabelsep` resets the space back to its default ‘\ ’.

The space above the beginning of the answer environment can be adjusted using the length `\aboveanswerskip`. The default value for this is

```
\setlength\aboveanswerskip{3pt}
```

and this can be reset to any desired vertical skip.

#### • shortquiz with Radio Buttons

The short quizzes (with multiple choices) can also be laid out using radio buttons rather than typeset lettering. Use a `shortquiz*` environment, with an optional argument the value of which is a unique name (which will be used to construct the titles of the radio buttons).<sup>13</sup> If you do not provide an optional argument name, one will programmatically be provided for you.

For example, the following code

```
\begin{shortquiz*}[KublaKhan1]
Was it in Xanadu did Kubla Kahn a stately pleasure dome decree?
\begin{answers}{4}
\Ans1 True & \Ans0 False
\end{answers}
\end{shortquiz*}
```

yields the following question:

**Quiz** Was it in Xanadu did Kubla Kahn a stately pleasure dome decree?

<sup>13</sup>In previous versions of `exerquiz`, there was a `*`-option that was used to signal the use of radio buttons. This option is still available, but the `shortquiz*` environment is preferred.

True                      False

Check the functionality of this question, and contrast it with the same question.

**Quiz** Was it in Xanadu did Kubla Kahn a stately pleasure dome decree?

True                      False

We have inserted two new commands prior to this last short quiz `\sqTurnOffAlerts` and `\sqCorrections` to change response feedback. The former turns off the alerts and the latter turns on the corrections: check for a correct answer and an cross for an incorrect answer. (It doesn't make sense to `\sqTurnOffAlerts` without `\sqCorrections`; `\sqCorrections` can be used without turning off the alerts.)

► These two commands only apply to a short quiz that uses radio buttons. You can reverse these two commands with `\sqTurnOnAlerts` and `\sqNoCorrections`, respectively. These settings are the defaults of the `shortquiz` with check boxes.

#### • shortquiz with Solutions

Another type of quiz that is easy to implement in PDF is the multiple choice quiz with immediate response with solution given. This too is a `shortquiz` environment:

```
\begin{shortquiz}
...Question goes here...
\begin{answers}[[<name>|*]{<num_cols>}
...
\Ans0 <an incorrect answer> &
...
\Ans1 <a correct answer> &
...
\end{answers}
\begin{solution}
...Solution to correct answer goes here...
\end{solution}
\end{shortquiz}
```

The `<name>` is a name used to create a hypertext jump to the solution; `<name>` will be the “named destination.” If `*` is used, `exerquiz` automatically generates the name. Use `<name>` when you want to refer to a solution in another part of the document; otherwise, use `*`, for convenience.

As before, `<num_cols>` is the number of columns to typeset the answers in.

The following example illustrates the quiz with solution.

**Quiz** Define a function  $f(s) = 4s^3$  and another function  $F(t) = t^4$ . Is  $F$  an antiderivative of  $f$ ?

(a) Yes                      (b) No

The verbatim listing:

```

\begin{shortquiz}
Define a function  $f(s)=4s^3$  and another
function  $F(t)=t^4$ . Is  $F$  an antiderivative of  $f$ ?
\begin{answers}[quiz:anti]{4}
\Ans1 Yes &\Ans0 No
\end{answers}
...solution not shown...
\end{solution}
\end{shortquiz}

```

Here is a listing of the same quiz using the `*` to automatically assign a destination label.

```

\begin{shortquiz}
Define a function  $f(s)=4s^3$  and another
function  $F(t)=t^4$ . Is  $F$  an antiderivative of  $f$ ?
\begin{answers}*{4}
\Ans1 Yes &\Ans0 No
\end{answers}
...solution not shown...
\end{solution}
\end{shortquiz}

```

#### • The `\bChoices`/`\eChoices` Commands

Beginning with `exerquiz` version 6.03, the command pair `\bChoices` and `\eChoices` are defined to help lay out the list of choices for a multiple choice question. This pair of macros is not really an environment, the `\eChoices` really does nothing other than to act as an ending marker. The idea behind this pair of macros is twofold:

1. To provide a convenient way of listing alternate choices for a multiple choice questions, the “environment” makes it easy to change the number of columns, or to change from tabular to list (or visa-versa);
2. To lay out the alternatives in a uniform manner, which will make it easy to develop techniques of randomizing the alternatives.

Here is an example of usage:

```

\begin{answers}{4}                                %<- 4 columns
\bChoices[2]                                       %<- use only first 2 columns
  \Ans0 a choice\eAns                               %<- choices delimited by
  \Ans1 another choice\eAns                         %   \Ans...\eAns
  \Ans0 still another choice\eAns
  \Ans0 another\eAns
  \Ans0 incoming\eAns
  \Ans0 more choices\eAns
  \Ans0 another still\eAns
  \Ans0 too many\eAns
  \Ans0 choices\eAns
\eChoices
\end{answers}

```

Note that the choices begin with the `\Ans` macro followed by a 0 or 1, as described in the previous pages. The end of the choice is delimited by the `\eAns` command (which is undefined, but is used as a argument delimiter). Note that it is not necessary (and would be a mistake) to place the column delimiter '&', or the end of line command '\\', this is done by the `\bChoices` command.

The commands `\bChoices` and `\eChoices` can be used within the `answers` environment as part of a `shortquiz` or a `quiz` environment.

Concerning the example above, the argument of the `answers` environment is 4 which means we are going to use the tabular environment with 4 columns. Now within those 4 columns we are going to only use the first 2 columns (this is the optional argument of `\bChoices`). If the optional argument is removed from `\bChoices`, the choices are typeset in the with 4 columns.

If the argument of `\answers` is changed to 1, the optional argument of `\bChoices` is ignored, and the alternatives are typeset in a one column list environment.

By changing the two parameters (the one for `\answers` and the one for `\bChoices`) you can easily modify how the alternatives are typeset.

► `\rowsep`: Space between rows (or items in list environment) can be adjusted with the `\rowsep` command. If you say `\rowsep{3pt}`, an additional 3pt space is added between rows or items in a list environment. The default is 0pt. The default is controlled through the command `\rowsepDefault`, like so,

```
\rowsepDefault{0pt}
```

For example,

```
\begin{answers}{4}
\rowsep{3pt}
\bChoices[2]
  \Ans0 a choice\eAns
  \Ans1 another choice\eAns
  \Ans0 still another choice\eAns
  \Ans0 another\eAns
  \Ans0 incoming\eAns
\eChoices
\end{answers}
```

Do not place this command inside the `\bChoices`/`\eChoices` pair, the above example shows the optimal placement, just before `\bChoices`. This separation `\rowsep` is reset to the default at the end of the listing. To get a uniform separation between rows across the document, redefine `\rowsepDefault` in the preamble, like so,

```
\rowsepDefault{3pt}
```

► No commands should appear *between* the pair `\bChoices` and `\eChoices` and *outside* an `\Ans` and `\eAns` pair. All such tokens are gobbled up.

**Limitations:** The command `\eAns` acts as a macro parameter delimiter: Everything between `\Ans` and `\eAns` is taken in as the argument of a macro; therefore, verbatim content generated by `\verb` is not permitted by  $\text{\LaTeX}$ . If you want your choices to contain verbatim text, use the style as described in 'Basic Usage' on page 70.

### • The questions Environment

The `questions` environment was designed to work with the `quiz` environment—taken up in [Section 21](#) below—but it works equally well with `shortquiz`.

Using the `questions` environment, quizzes defined by `shortquiz`, with/without solutions, can be mixed together and combined to make a “mini-quiz”. For example,

**Quiz** Determine the LCD for each of the following.

$$1. \frac{3x}{2y^2z^3} - \frac{2}{xy^3z^2}.$$

(a) LCD =  $2xy^5z^5$

(b) LCD =  $2y^3z^3$

(c) LCD =  $2xy^3z^3$

(d) LCD =  $2xy^3z^5$

$$2. \frac{x+y}{3x^{3/2}y^2} - \frac{x^2+y^2}{6xy^4}.$$

(a) LCD =  $18x^{3/2}y^4$

(b) LCD =  $6x^{3/2}y^4$

(c) LCD =  $18xy^4$

(d) LCD =  $6xy^4$

The first question is given without a solution, the second has a solution attached to it. An abbreviated verbatim listing follows.

```
\begin{shortquiz}
Determine the LCD for each of the following.
\begin{questions}
\item $\dfrac{3x}{2y^2z^3}-\dfrac{2}{xy^3 z^2}$$.
\begin{answers}{2}
...
\end{answers}
\item $\dfrac{x+y}{3 x^{3/2}y^2}
-\dfrac{x^2+y^2}{6 x y^4}$$.
\begin{answers}[quiz:LCB]{2}
...
\end{answers}
\begin{solution}
If you erred on this one, ... ..
\end{solution}
\end{questions}
\end{shortquiz}
```

**Important:** The `questions` environment can be nested three deep.

Between items, you can say `\pushquestions`, make a comment, to continue with more questions, type `\popquestions`. Here is an example without using `shortquiz` environment.

1. This is my first question.

2. This is my second question.

We now begin a multipart question.

3. Answer each of the following:

(a) First part of the question.

Be sure to concentrate on this next question, its a hard one; recall that  $1 + 1 = 2$ .

(b) Second part of the question.

(i) This is a subpart of this part.

The questions environment can be nested up to three deep.

(ii) This is a second subpart of this part.

4. This is the fourth question.

A partial listing of the above environment is presented below.

```
\begin{questions}
  \item This is my first question.
  \item This is my second question.
\pushquestions
We now begin a multipart question.
\popquestions
  \item Answer each of the following:
  ...
\end{questions}
```

## 20.2. Options of the shortquiz Environment

### • The forpaper Option

The forpaper option has already been described. The solutions to shortquiz questions are not typeset on separate pages, but are separated by a `\medskip`.

Following up on the pre-test angle discussed in “[Redesigning the shortquiz Environment](#)”, page 79, a single document can be constructed that can be published on-line, or published for paper distribution. This feature may be useful to some educators.

### • The solutionsafter Option

The solutionsafter option works as described for the exercise environment. The option just sets a Boolean switch. This switch can be controlled with `\SolutionsAfter` and `\SolutionsAtEnd`. Here is a simple example.

**Quiz** In what year did Columbus sail the ocean blue?

(a) 1490

(b) 1491

(c) 1492

(d) 1493

*Solution:* Columbus sailed the ocean blue in 1492. Some say he discovered San Salvatore, others say he first sited Cat Island in the Bahamas. ■

In the example above, `\SolutionsAfter` was placed before the shortquiz environment and the `\SolutionsAtEnd` was placed just after.

This option may be useful in publishing an answer key to a multiple choice quiz. The quiz and solutions can be created together. The quiz can be published, then later, the quiz with complete solutions.

### • The proofing Option

For proofreading, use the proofing option of exerquiz.

```
\usepackage[proofing]{exerquiz}
```

When used, a symbol, defined by the command `\proofingsymbol`, will mark the correct answers, as defined in your source file. The command `\proofingsymbol` may be redefined, its definition is

```
\newcommand\proofingsymbol{\textcolor{webgreen}{\bullet}}
```

This option works for the `quiz` environment defined below (page 83), as well.

### • The showgrayletters Option

When `showgrayletters` is used, the form field check boxes that appear in a `quiz` or in a `shortquiz` will have a gray capital letter A, B, C, etc. underneath it. This letter can then be referred to in the text or the solution.

This option is global and is controlled by a switch, `\ifaebshowgrayletters`. The gray letter feature can be turned on and off locally: To turn on this feature, insert the command `\graylettersOn` at some appropriately chosen point in the document; to turn off the gray letter feature insert `\graylettersOff`.

In support of the `showgrayletters` option is a new command `\REF`. `\REF` acts like the  $\TeX$  command `\ref` with the `hyperref` modifications, but it converts the reference to uppercase. When `\ref` would typeset the letter 'a', for example, `\REF` would typeset the letter 'A'. `\REF`, like `\ref`, typesets a `hyperref` link. `Hyperref` defines a `*` version of `\ref`; `\ref*` typesets the reference, but does not create a link; `\REF*` does the same. When `\aebshowgraylettersfalse` is in effect, `\REF` does not capitalize the reference.

The `showgrayletters` feature and the `\REF` command work equally well for the `quiz` environment.

► Below is an example of this.

**Quiz** Answer this if you can!

- |                              |                                  |
|------------------------------|----------------------------------|
| A This is a possible answer. | B Try this one (the correct one) |
| C This is an answer.         | D Another alternative.           |

Reference both (A) and (B).

The code is

```
\graylettersOn
\begin{shortquiz*}[TestSQ]
Answer this if you can!
\begin{answers}{2}
\bChoices
  \Ans0\label{testsqFirst} This is a possible answer.\eAns
  \Ans1\label{testsqSecond} Try this one (the correct one).\eAns
  \Ans0 This is an answer.\eAns
```

```

        \Ans0 Another alternative.\eAns
    \eChoices
    \end{answers}
    \end{shortquiz*}
    \noindent Reference both (\REF{testsqFirst}) and (\REF*{testsqSecond}).
    \graylettersOff

```

Notice that the gray letters was not turned to off until after the usage of `\REF`.

**Important:** The gray letters are typeset into the document. Do not use a background color for the checkboxes as this color will cover up the gray letters. The default background color checkboxes is transparent, keep it that way.

The command `\insertGrayLetters` is used to insert the gray letters; this command can be redefined as desired. See the source file `exerquiz.dtx`. Also see the definition of `\bottomOfAnsStack`.

- **Moving the Solution Set**

The solution set, by default, comes last in the file. You can move its positioning by including the command `\includequizzesolutions` at any point *after* the last exercise. You'll note that I have moved the solutions in this file before the **References** section, as indicated, for example, by its position in the table of contents.

### 20.3. Redesigning the shortquiz Environment

The `shortquiz` environment can be redesigned to better suit your needs. In the paragraphs below, we describe how you can change titles and form elements.

- **Changing Titles**

You can temporarily change the title for the `shortquiz` environment by redefining the macro `\sqlabel`; for example, the default definition of this macro is

```
\newcommand\sqlabel{\textcolor{red}{Quiz.}}
```

The syntax for redefining `\sqlabel` is

```
\renewcommand\sqlabel{...new code goes here...}
```

You can redefine the *default* label as well; the default label is the title label that `shortquiz` uses when `\sqlabel` is *not present*. The default label is `\eq@sqlabel` and must be redefined using the macro `\renewcommand`. The best place for this to be done is the preamble. The syntax:

```

\makeatletter    % make 'at'=@ a normal letter
\renewcommand\eq@sqlabel{...new code goes here...}
\makeatother    % make 'at'=@ something special(other)

```

To change the entire document to use 'Exam' instead of 'Quiz', make the following changes in the preamble:

```

\makeatletter
% change default quiz title to 'Exam'
\renewcommand\eq@sqlabel{\textcolor{red}{Exam.}}
% change quiz solutions return label
\renewcommand\eq@sqs|rtnlabel{End Exam}
% change solutions label
\renewcommand\eq@sqs|label{%
  \string\textbf{Solution to Exam:}}
\renewcommand\eq@sqs|sectitle{Solutions to Exams}
% change default running header for solutions
\renewcommand\eq@qs|secrunhead{Solutions to Exams}
\makeatother

```

► The above commands are ‘global’—they are in effect throughout the entire document. You can temporarily change these labels using the `\sqlabel`, `\sqs|rtnlabel`, `\sqs|label` and `\sqs|sectitle`. Note that you cannot temporarily change the running label `\eq@qs|secrunhead`, this should be set in the preamble.

Should you want to make a series of multiple choice questions (using the `questions` environment) and combine them into a sort of review or pretest, a useful idea would be to number the solutions. The counter that maintains the question number is called `questionno`. You can then, for example, define

```

\renewcommand\eq@sqs|label{%
  \string\textbf{Solution to Question \thequestionno:}}

```

✎ See the sample file `webeqtst.tex` for examples.

#### • Modifying Form Elements

For quizzes that use radio buttons (see page 72 above), the appearance of the radio buttons can be controlled using the “every” mechanism as described in the document `eFormMan.pdf` on *eForm Support* for the `AcroTeX Bundle`. The radio buttons can be modified using the command `\everysqRadioButton`.

Prior to the short quiz below, the following command was executed

```

\everysqRadioButton{\BC{.690 .769 .871}\BG{.941 1 .941}}

```

**Quiz** Was it in Xanadu did Kubla Kahn a stately pleasure dome decree?

True                      False

Return to the defaults, if desired, by then emitting

```

\everysqRadioButton{}

```

► The short quiz can also have fill-in questions and various other controls, these are described in Section 22.5, *The shortquiz Environment*. There too, methods of modifying the appearance of the form elements are discussed.

## 20.4. `\titleQuiz` and `\fancyQuizHeaders`

✎ The demo file for these features is `title_quiz.tex`.

- `\titleQuiz`

To assign a title to your short quiz use `\titleQuiz`. This title appears just after the typeset word “Quiz”.

**Quiz Two Question MC** Answer each of these to the best of your ability.

1. Math is fun.
  - (a) True
  - (b) False
2. Statistics is fun.
  - (a) True
  - (b) False

The partial verbatim code for the above quiz follows:

```
\titleQuiz{Two Question MC}
\begin{shortquiz}\label{myCoolSQ}
Answer each of these to the best of your ability.
\begin{questions}
...
\end{questions}
\end{shortquiz}
```

The format of the title is controlled by the command `\titleQuizfmt`, the default definition is `\newcommand\titleQuizfmt{\bfseries}`. The value of the `\titleQuiz` command can act as an argument for `\titleQuizfmt` if you redefine `\titleQuizfmt` to have an argument. For example,

```
\renewcommand\titleQuizfmt[1]{\bfseries(#1)}
```

Now, the title will appear in a set of parentheses. The title and formatting are enclosed in a `\mbox`, so font and style changes are localized to that `\mbox`.

If a quiz does not have a title, as defined by `\titleQuiz`, the title defaults to an empty title.

The definition of `\titleQuiz` is such that the command `\makeatletter` is executed before the argument for `\titleQuiz` is read. This means that the special character ‘@’ can be used within the argument of `\titleQuiz`. For example, the `shortquiz` environment maintains a text macro `\@shortquizCnt` which contains the current short quiz count number. Thus you can say `\titleQuiz{Short Quiz \@shortquizCnt}`, the title of the short quiz will include a short quiz number as well.

The `\titleQuiz` is used for quizzes as well as with short quizzes. It is possible to have different formatting for each of these two environments by using the `\ifQuizType` command. The command takes two arguments:

```
\ifQuizType{<quiz_code>}{<shortquiz_code>}
```

For example, we could define `\titleQuizfmt` as follows:


```
\renewcommand\titleQuizfmt{\ifQuizType{\bfseries}{\slshape}}
```

or with an argument, as mentioned above, you could define `\titleQuizfmt` with an argument, like so,

```
\renewcommand{\titleQuizfmt}[1]{\ifQuizType{\bfseries(#1)}{\slshape[#1]}}
```

#### • `\titleQuiz*`

For `shortquiz` environment—not for the `quiz` environment—there is a ‘\*’ optional first parameter. When the ‘\*’ option is taken, the quiz label (`\sqlabel`) is redefined to be the title of the quiz text.

 The `title_quiz.tex` demo file illustrates the ‘\*’ option.

Thus, if we title a quiz with the ‘\*’, we get

**My Cool Quiz** Answer each of these to the best of your ability.

1. Math is fun.
  - (a) True
  - (b) False
2. Statistics is fun.
  - (a) True
  - (b) False

The formatting is that defined by the `\titleQuizfmt`, which is `\bfseries`. In addition to the `\ifQuizType` command, there is also the `\ifstaroption` command. It takes two parameters

```
\ifstaroption{<code if \titleQuiz*>}{<otherwise>}
```

This allows us to not only format for quizzes, but for short quizzes, with or without the ‘\*’ option. If we redefine `\titleQuizfmt` to be

```
\renewcommand{\titleQuizfmt}{\ifQuizType{\bfseries}%
{\ifstaroption{\color{red}}{\bfseries}}}
```

we format the quiz title as `\bfseries` is either a quiz or a short quiz without ‘\*’ option, and as `\color{red}` with a short quiz with the ‘\*’ option. The results are seen in the next quiz, with this redefinition:

**My Swave Quiz** Answer each of these to the best of your ability.

1. Math is fun.
  - (a) True
  - (b) False
2. Statistics is fun.
  - (a) True
  - (b) False

#### • Cross-Referencing

Quizzes and shortquizzes conform to the cross-referencing system of  $\LaTeX$ . For example, we used the label `\label{myCoolSQ}` in the last quiz, the one on page 81. The title of the quiz is **Two Question MC**. The code for the past two sentences is

```
For example, we used the label \verb!\label{myCoolSQ}! in the last
quiz, the one on page~\pageref{myCoolSQ}. The title of the quiz is
\nameref{myCoolSQ}.
```

- `\fancyQuizHeaders`

The `\fancyQuizHeaders` command, a companion feature to `\titleQuiz`, enhances the labeling of the quiz in the solutions section of the document. If `\fancyQuizHeaders` is executed in the preamble, or prior to a quiz, the solution header has the following default format:

**Two Question MC: Question 1.** The answer is true, of course, math is fun!

The above assumes the title of the quiz is the one given in the previous section.

When a quiz has no title, the header reverts to **Solution to Quiz:** followed by the question number.

► You can turn off fancy quiz headers by using `\restoreDefaultQuizHeaders`.

You can modify the fancy headers with a combination of setting `\setsoInspace` and/or redefining the command `\eq@fancyQuizHeadersfmt`. First `\setsoInspace`: This is a hook into the space immediately following the header. To insert a `\medskip` after the header, for example, you can say `\setsoInspace{\par\medskip\noindent}` a `\medskip` is created after the solution header and the solution above appears as

**Two Question MC: Question 1.**

The answer is true, of course, math is fun!

The definition of `\eq@fancyQuizHeadersfmt` is

```
\def\eq@fancyShrtQuizHeadersfmt{\eq@fancyQuizHeadersfmt}
```

The `\eq@fancyQuizHeadersfmt` is the quiz counterpart to this `shortquiz` construct. The fancy short quiz solutions header defaults to the same header used for a quiz. You can copy and paste the definition of `\eq@fancyQuizHeadersfmt` into the definition of the command `\eq@fancyShrtQuizHeadersfmt`, then modify as described in '`\fancyQuizHeaders`' on page 99.

## 21. The quiz Environment

Use the `quiz` environment to create graded quizzes. In this case, several (many) questions are bundled together. The student takes the quiz and responses are recorded by JavaScript. Upon completion of the quiz, the total score is reported to the student.

The `quiz` environment can generate multiple choice questions and math/text fill-in questions. The discussion of math and text fill-in questions is postponed to [Section 22](#) on page 103

There are two types of quizzes, the `link-style` and `form-style`. In [Section 21.5](#), we see that the `quiz` environment can also correct the quizzes.

The `quiz` environment takes one required parameter, the `quizfieldname`, which is a string.

```
\begin{quiz}{quizfieldname}
...
\end{quiz}
```

**Important:** Exerquiz uses the `quizfieldname` to build Acrobat form fields, and also JavaScript variables; therefore, there are restrictions on the choice of the name. The name must be a (JavaScript) *identifier*, the rules to be an identifier are these.

The first character shall be a letter, an underscore (`_`), or a dollar sign (`$`); subsequent characters may be any letter or digit, or an underscore or a dollar sign.

Do not use other symbols, like a colon (`:`) or a period (`.`) as part of the field name, these will confuse the JavaScript interpreter, and may cause the quiz not to function as designed.

The `quiz` environment consists of a series of nested environments. Inside the `quiz` environment is the `questions` environment (an enumerated list), and within that environment is the `answers` environment. Symbolically, we can express this as

$$\text{quiz} \supseteq \text{questions} \supseteq \text{answers}$$

The term ‘answers’ is, perhaps, not sufficiently descriptive; ‘alternatives’ would be more appropriate, but it requires more typing. `{:-{}`

► The `answers` environment requires one parameter, the `num_cols`. If `num_cols` is 1, a `list` environment is created; otherwise, a `tabular` environment is used.

This (`tabular`) environment has the following syntax:

```
\begin{quiz}{quizfieldname}
The preamble to the questions goes here.
\begin{questions}
\item State first question...
\begin{answers}{4} % <- num_cols = 4
\Ans0 ... &\Ans1 ... &\Ans0 ... &\Ans0 ...
\end{answers}
...
\item n th question....
\begin{answers}{4} % <- 4 column format
\Ans0 ... &\Ans1 ... &\Ans0 ... &\Ans0 ...
\end{answers}
\end{questions}
\end{quiz}
```

► Following the `quiz`, or anywhere in the document, place the macro `\ScoreField`, defined in `exerquiz`, to display the results of the quiz:

```
\ScoreField{quizfieldname}
```

**Important.** The value of the parameter of the macro `\ScoreField` must match the `quizfieldname` defined in the argument of the `quiz` environment.

► There is a convenience macro, `\currQuiz`, that holds the name of the current quiz. Thus, we could have instead typed:

```
\ScoreField\currQuiz
```

Read the paragraph entitled ‘[The Score Field](#)’ on page 96 for more details on this macro.

### 21.1. Options of the quiz Environment

In addition to the options listed in the section 20.2, page 77, the following is designed for quizzes.

- **The option `noquizsolutions`**

For online quizzing, where results are stored in some way (database, e-mail, text file) the presence of the solutions in the same file as the questions is a breach in security of the quiz. Using the `noquizsolutions` removes the solutions to a `quiz` and `shortquiz` from the document under construction.

### 21.2. The answers Environment

The simplest type of question that can be posed is the multiple choice question. For this question type, the `answers` environment is used. The basic syntax is

```
\item A question
\begin{answers}[[<name>]|*]{<num_cols>}
\Ans[<partial_pts>]{0|1}....
\end{answers}
```

The `[<name>]`, an optional argument, is a name used to create a hypertext jump to the solution; `[<name>]` will be the “named destination.” If `*` is used, `exerquiz` automatically generates the name. Use `<name>` when you want to refer to a solution in another part of the document; otherwise, use `*`, for convenience.

If either the `[<name>]` or `*` is used, a `solution` environment should follow.

The `<num_cols>` is the number of columns to typeset the answers in. When `<num_cols>` is 1, the answers are set in a list environment; otherwise, a tabular environment is used. In the latter case, the usual table syntax is used. (Use `&` to separate columns, and `\\` to terminate a row.)

The `\Ans` command has one required argument, a 1 to mark that this answer is a correct answer, and a 0 to mark that this answer is not true. `\Ans` also has an optional argument to pass partial points rewarded for a wrong answer.

This environment is illustrated extensively in this section. See also the `manswers` environment, ‘[The manswers Environment](#)’ on page 90, for posting questions with multiple correct answers.

### 21.3. Basic Usage

In this section we discuss the two `quiz` styles: [Link-Style Quiz](#) and [Form-Style Quiz](#).

A paragraph is devoted to some modifications that can be made at the beginning and end of the quiz. In addition, a [proofing](#) option is also described.

### • Link-Style Quiz

This style uses links to record the choices to the alternatives. The link method takes up less space in the pdf file than does the form-style.

Below is an example of a link-style quiz. Instructions should be given to guide the student in operating the quiz correctly.

**Instructions.** You must click on 'Begin Quiz' to initialize the quiz. Not doing so brings forth an error message. When finished, click on 'End Quiz'.

**Begin Quiz** Using the discriminant,  $b^2 - 4ac$ , respond to each of the following questions.

1. Is the quadratic polynomial  $x^2 - 4x + 3$  irreducible?  
(a) Yes (b) No
2. Is the quadratic polynomial  $2x^2 - 4x + 3$  irreducible?  
(a) Yes (b) No
3. How many solutions does the equation  $2x^2 - 3x - 2 = 0$  have?  
(a) none (b) one (c) two

**End Quiz**

- ▶ While you are taking the test, and before you click on 'End Quiz', you can change your answers.

```
\begin{quiz}{qzdiscr} % qz:discr=quiz field name
Using the discriminant,  $b^2-4ac$ , respond to each of the
following questions.
\begin{questions}
\item Is the quadratic polynomial  $x^2-4x + 3$  irreducible?
\begin{answers}{4}
\Ans0 Yes &\Ans1 No
\end{answers}
\item Is the quadratic polynomial  $2x^2-4x+3$  irreducible?
\begin{answers}{4}
\Ans1 Yes &\Ans0 No
\end{answers}
\item How many solutions does the equation  $2x^2-3x-2=0$  have?
\begin{answers}{4}
\Ans0 none &\Ans0 one &\Ans1 two
\end{answers}
\end{questions}
\end{quiz}\par
\ScoreField\currQuiz % matching quiz field name
```

- ▶ The convenience text macro, `\currQuiz`, contains the name of the current quiz. This macro can be used as the argument of `\ScoreField`.

### • Form-Style Quiz

You may be thinking that such a quiz format—one in which the student cannot see the choices made—is not very good. It is perhaps adequate for two or three quick questions. For a longer quiz format, one would like to see a “checkbox” format. A quiz with a checkbox format can be obtained using the `quiz*` environment:<sup>14</sup>

```
\begin{quiz*}{quizfieldname}
...
...same format as before...
...
\end{quiz*}
```

Here is the same sample quiz with the form-style option. The only change in the code is the insertion of the `*`-option.

**Instructions.** You must click on ‘Begin Quiz’ to initialize the quiz. Not doing so, brings forth an error message. When finished, click on ‘End Quiz’.

**Begin Quiz** Using the discriminant,  $b^2 - 4ac$ , respond to each of the following questions.

1. Is the quadratic polynomial  $x^2 - 4x + 3$  irreducible?  
     Yes                      No
2. Is the quadratic polynomial  $2x^2 - 4x + 3$  irreducible?  
     Yes                      No
3. How many solutions does the equation  $2x^2 - 3x - 2 = 0$  have?  
     none                      one                      two

**End Quiz**

- ▶ Before completing the quiz, a student can easily change alternatives.
- ▶ This type is more suitable for longer quizzes. The choices the student makes are visually recorded for the student to review and change before clicking on ‘End Quiz’. A partial verbatim listing:

```
\begin{quiz*}{qzdiscrf}
Using the discriminant,  $b^2-4ac$ , respond to each of the
following questions.
\begin{questions}
.....
.....
\end{questions}
\end{quiz*}\par
\ScoreField{qzdiscrf}
```

See the sample file `webeqtst.tex` and for examples.

<sup>14</sup>In previous versions of `exerquiz`, there was a `*`-option for this purpose. The `quiz*` has replaced the `*`-option and is the preferred method, though the `*`-option is still defined so as not to break previous code.

### • Using Circular Radio Buttons with MC Questions

As seen with the examples of the previous section, the form-type quizzes have a rectangular check box for the user to make his selection. Acrobat also offers a circular radio button that can be used to make a multiple choice.

To create circular radio buttons for a MC question execute `\useMCCircles` before the quiz, or, perhaps, in the preamble of your document.

To return to the default rectangular choice boxes, execute `\useMCCRects`. This setting is the default, there is no need to execute it unless `\useMCCircles` has already been executed.

Executing `\useMCCircles` does not affect the `answers` environment, page 90), so you can have circular radio buttons for multiple choice, and rectangular choice boxes for multiple selection problems.

✎ The sample file `quiz_w_circ.tex` illustrates these two commands.

### • Overriding the 'quiztype' Parameter

You can globally declare that all quizzes to be a link-type or form-type by using the command `\quiztype`. Placing `\quiztype{f}` in the preamble (or prior to any quiz) will cause all quizzes following that command to be form-type quizzes. Similarly, executing `\quiztype{l}` will produce all link-type quizzes.

You can make the environments revert back to their normal behavior by using the command `\defaultquiztype`, which cancels out the `\quiztype` command.

✎ The sample file `quizpts.tex` illustrates these collections of commands.

*Within a quiz*, you can say `\useForms`, to change the style to a visible checkbox for multiple choice or multiple selection questions; use `\useLinks` to change the style to links with alphabetized alternatives. Using `\restoreFLTypeDefault` restores the next multiple choice/selection question back to the default determined by the quiz environment (`quiz` versus `quiz*`), or as set by the `\quiztype` command.

### • The BeginQuiz and EndQuiz Form Buttons

The default setup the `quiz` environment is to have hypertext links for the 'Begin Quiz' and 'End Quiz'. You can also redefine this linking and use a form button instead. Prior to your quiz, use the following code, if desired.

```
\useBeginQuizButton
\useEndQuizButton
```

Answer each of the following. Passing is 100%.

- Who created  $\TeX$ ?
 

(a) Knuth	(b) Lamport	(c) Carlisle	(d) Rahtz
-----------	-------------	--------------	-----------
- Who originally wrote  $\LaTeX$ ?
 

(a) Knuth	(b) Lamport	(c) Carlisle	(d) Rahtz
-----------	-------------	--------------	-----------

Revert back to link-style as follows:

```
\useBeginQuizLink
\useEndQuizLink
```

The commands `\useBeginQuizButton` and `\useEndQuizButton` each have an optional argument that can be used to modify the appearance of the buttons.

```
\useBeginQuizButton[\textColor{0 0 1}]
```

would create a ‘Begin Quiz’ button with blue text for the button label.

#### • The proofing Option

For proofreading, use the proofing option of `exerquiz`.

```
\usepackage[proofing]{exerquiz}
```

When used, a symbol, defined by the command `\proofingsymbol`, will mark the correct answers, as defined in your source file. The command `\proofingsymbol` can be redefined, its definition is

```
\newcommand\proofingsymbol{\textcolor{webgreen}{\bullet}}
```

This option works for the `shortquiz` environments defined above (page 70), as well.

#### • The showgrayletters Option

The `showgrayletters` option is valid for the `quiz` and `shortquiz` environments. See the section entitled ‘The `showgrayletters` Option’ on page 78 for a detailed explanation of this option.

#### • Setting the Threshold

The default behavior of the `quiz` environment is that a student can begin the quiz and finish the quiz without answering any or all of the questions. This is called a `lowThreshold` and is the default behavior.

The document author can set a `highThreshold` by re-defining the `\minQuizResp` macro. The default definition is

```
\newcommand\minQuizResp{lowThreshold}
```

However, if you make the definition

```
\renewcommand\minQuizResp{highThreshold}
```

the student is required to answer all the questions of a quiz.

Actually, `lowThreshold` and `highThreshold` are JavaScript functions that are called when the “End Quiz” button is clicked. If the threshold is not met, an alert box appears informing the user of this.

The document author can write a custom threshold function and place its name in the `\minQuizResp` macro. See the `exerquiz` source code—in either `exerquiz.dtx` or `exerquiz.sty`—for the `highThreshold()` function for an example of how to do this.

## 21.4. The `manswers` Environment

Beginning with Exerquiz v.6.2, the `manswers` environment is defined to support multiple selection questions. (`manswers` stands for multiple answers). This is in contrast with the `answers` environment in which there is only one correct answer; for `manswers`, there are several correct answers.

✂ See the example file for this feature is `manswers.tex`.

The syntax is the same as that of the `answers` environment, ‘[The answers Environment](#)’ on page 85. Below is example fragment, taken from `manswers.tex`

```
\item\PTs{5} Select which people were the President
of the United States.
\begin{manswers}{4}
  \bChoices[2]
    \Ans[-1]{0} Henry Clay\eAns
    \Ans[-1]{0} Ben Franklin\eAns
    \Ans[1.5]{1} Andrew Jackson\eAns
    \Ans[1.5]{1} Ronald Reagan\eAns
    \Ans[-1]{0} George Meade\eAns
    \Ans[2]{1} Grover Cleveland\eAns
    \Ans[-1]{0} John Jay\eAns
    \Ans[-1]{0} Paul Revere\eAns
  \eChoices
\end{manswers}
```

The arguments of the `\Ans` command is the same as those of `\Ans` in the `answers` environment.

When the `\ScoreField` is used to display results, a `manswers` question is correct if and only if all correct answers are checked, and none of the incorrect answers are checked.

When the `\PointsField` is used (see ‘[Adding Points to a Quiz](#)’ on page 100), and points are assigned, as in the above code fragment, the optional argument of `\Ans` is used to assign points for each response (if there is no optional argument, it defaults to 0). When a student selects a choice, the point value in the optional argument is added to the point score. In the case of choosing Ronald Reagan, 1.5 points are added to the point score, while -1 point is added to the point score for choosing John Jay. The minimal point score for each question is zero, unless `\negPointsAllowed` command is executed in the preamble, see ‘[\negPointsAllowed](#)’ on page 102.

## 21.5. Correcting the Quizzes with JavaScript

Beginning with `exerquiz`, version 1.2, you can now correct quizzes created by the `quiz` environment. To correct the quizzes, simply include an additional element into your quiz, a correction button. The correction button is installed using the macro `\eqButton`.

► The following is a link-style quiz.

**Begin Quiz** Using the discriminant,  $b^2 - 4ac$ , respond to each of the following questions.

1. Is the quadratic polynomial  $x^2 - 4x + 3$  irreducible?
  - (a) Yes
  - (b) No
2. Is the quadratic polynomial  $2x^2 - 4x + 3$  irreducible?
  - (a) Yes
  - (b) No
3. How many solutions does the equation  $2x^2 - 3x - 2 = 0$  have?
  - (a) none
  - (b) one
  - (c) two

**End Quiz**

**Legend:** A ✓ indicates a correct response; a ✗, indicates an incorrect response, in this case, the correct answer is marked with a ●.

A partial verbatim listing of this quiz follows:

```
\begin{quiz}{qzdiscr11} Using the discriminant,  $b^2-4ac$ ,
respond to each of the following questions.
\begin{questions}
.....
.....
.....
\end{questions}
\end{quiz}

\ScoreField{qzdiscr11}\eqButton{qzdiscr11}
```

- ▶ The macro `\eqButton` is used to create a nice “correction” button. JavaScript is used to correct the quiz. The only required argument is the field label that uniquely defines the field in which the total score is placed. See the section entitled ‘The ‘Correction Button’ on page 96 for more details on how to use this macro.
- ▶ The `\eqButton` will not work until the user has clicked on ‘End Quiz’. The user can re-take the quiz simply by clicking on ‘Begin Quiz’, the form fields and JavaScript variables will be cleared.
- ▶ It is possible to take this form data and submit it to a CGI script for processing. (The data can be saved to a database, for example.) However, there is no built-in capability for this in the `exerquiz` package.

The same quiz can be written in form-style simply by inserting the `*`-option.

**Instructions.** You must click on ‘Begin Quiz’ to initialize the quiz. Not doing so, brings forth an error message. When finished, click on ‘End Quiz’.

**Begin Quiz** Using the discriminant,  $b^2 - 4ac$ , respond to each of the following questions.

1. Is the quadratic polynomial  $x^2 - 4x + 3$  irreducible?  
     Yes                      No
2. Is the quadratic polynomial  $2x^2 - 4x + 3$  irreducible?  
     Yes                      No
3. How many solutions does the equation  $2x^2 - 3x - 2 = 0$  have?  
     none                    one                       two

**End Quiz**

► In the partial verbatim listing that follows, notice the field name has been changed from `qzdiscr1l`, which is the name of the quiz previous to his one, to `qzdiscr1f`. Different quizzes must have unique field names.

```
\begin{quiz*}{qzdiscr1f} Using the discriminant,  $b^2-4ac$ ,
respond to each of the following questions.
\begin{questions}
.....
.....
.....
\end{questions}
\end{quiz*}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

► Notice that in this example, the `\ScoreField` and the `\eqButton` are positioned following the ‘End Quiz’; this makes the design more compact and nicer looking.

#### • The `nocorrections` Option

Including the corrections adds quite a bit more JavaScript code to the .pdf document, this feature is ‘on’ by default. If you have a document in which you do not want to have the option of offering corrected quizzes, then just specify `nocorrections` in the option list of `exerquiz`.

There are also a couple of macros you can use to override the option switch, they are `\CorrectionsOn` and `\CorrectionsOff`. Each remains in effect until the other is invoked.

► If the `nocorrections` option is taken, then the `\eqButton` does not appear for a quiz.

#### • Showing Partial Credit Markup

When a quiz is corrected, it is possible to show how much credit was given.

- ✎ The demo file for this feature is `pc_test.tex`.
- ✎ The example file `pc_test1.tex` illustrates how to have a multiple selection problem and a grouped problem with no partial credit. The student must get all parts correct to get full credit for that problem, otherwise, no credit is assigned.

Place the command `\showCreditMarkup` before a quiz you want the partial credit markup. The command `\hideCreditMarkup` hides this markup (it switches off the previous `\showCreditMarkup`); place this command before a quiz for which you don't want the markup. The default is not to show any partial credit markup.

When a question item introduces a series of question, but poses no question itself, place `\multipartquestion` prior to that item, for example

```
\multipartquestion
\item\PTs{6} Answer each of these questions
\begin{question}
  \item\PTs{3} Answer this
  \item\PTs{3} Answer that
\end{question}
```

Other relevant commands are `\aeb@creditmarkup` and `\aeb@creditmarkupfmt`; the first one defines a little text field to hold the partial-credit data and the second one sets the placement of the text field. These can be redefined if the default definitions are not to your liking. The markup in red that appears in the left margin shows the number of points the student received for that problem, for example **3 pts**. The string `pts` (and its singular counterpart, `pt`) is language dependent, in English it refers to “points” (or “point”). You can change this string to an abbreviation of points (point) using the commands `\ptsLabel` and `\ptLabel`. The default definitions are `\ptsLabel{pts}` and `\ptLabel{pt}`, respectively.

**Important:** Adobe Reader 5.0 or later is required for this feature.

## 21.6. Quizzes with Solutions

In addition to scoring and marking the quizzes, you can also (optionally) provide solutions as well. To enter a solution to a multiple choice question, use a `solution` environment, and attach a named destination to the `answers` environment. A partial verbatim listing follows the next example.

**Begin Quiz** Answer each of the following. Passing is 100%.

- Who created  $\TeX$ ?

Knuth                      Lamport                      Carlisle                      Rahtz

- Who originally wrote  $\LaTeX$ ?

Knuth                      Lamport                      Carlisle                      Rahtz

**End Quiz**

After the quiz is completed and the corrections button is pressed, the corrections appear. The correct answer has a green filled circle or a green check; this circle is now outlined by a green rectangle to indicate that this is a link to the solution. Click on the green dot and jump to the solution!

Solutions do not have to appear. Some problems can have solutions, while others do not. The ones with the solutions have the green boundary to indicate a link to the solution.

Here is a partial listing of the above example.

```
\begin{quiz*}{qzTeX1} Answer each of the following.
Passing is 100\%.
\begin{questions}
\item Who created \TeX?
\begin{answers}[knuth]4
\Ans1 Knuth &\Ans0 Lamport &\Ans0 Carlisle &\Ans0 Rahtz
\end{answers}
\begin{solution}
Yes, Donald Knuth was the creator of \TeX.
\end{solution}
....
\end{questions}
\end{quiz*}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

- ▶ Notice that in the `answers` environment, an optional parameter `[knuth]` appears. The value of this parameter is a unique name for the solution to the quiz. Notice as well that the `solution` environment follows, and is not nested within the `answers` environment.

## 21.7. How to Modify the quiz Environment

There are four ways the appearance of the quizzes can change:

- change the titles
- change the ‘check’ appearance
- change the text field in which the score appears,
- change the appearance of the ‘Correction’ button.

This section discusses each of these four in turn.

### • The Quiz Titles

It is possible to redefine the quiz titles and other labels if desired.

- ▶ Locally:

```
\renewcommand\bqlabel{Begin Exam}
\renewcommand\eqlabel{End Exam}
```

- ▶ Globally:

```
\makeatletter
\renewcommand\eq@bqlabel{Begin Exam}
\renewcommand\eq@eqlabel{End Exam}
\makeatother
```

- **The check appearance**

The appearance of the ‘check’ can be chosen using the `\symbolchoice` macro of the `exerquiz` package. The permissible values for the argument of `\symbolchoice` are `check` (the default), `circle`, `cross`, `diamond`, `square`, and `star`.

This quiz was generated by inserting `\symbolchoice{diamond}` before the quiz. The multiple choice field is actually a radio button field. The appearance of these radio buttons can be modified using the command `\everyqRadioButton`.

```
\symbolchoice{diamond}
\everyqRadioButton{\textColor{0 0 1}
  \BC{.690 .769 .871}\BG{.941 1 .941}}
```

**Begin Quiz** Answer each of the following. Passing is 100%.

1. Who created  $\TeX$ ?

Knuth                      Lamport                      Carlisle                      Rahtz

2. Who originally wrote  $\LaTeX$ ?

Knuth                      Lamport                      Carlisle                      Rahtz

**End Quiz**

If desired, we can return to the defaults:

```
\symbolchoice{check} % restore the default
\everyqRadioButton{}
```

► The `\symbolchoice` can also be introduced into the flow of the code through the `\everyqRadioButton`:

```
\everyqRadioButton{\symbolchoice{diamond}\textColor{0 0 1}
  \BC{.690 .769 .871}\BG{.941 1 .941}}
```

After the quiz, we could return to the defaults by

```
\everyqRadioButton{}
```

► See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” command.

- **Change color of Correction Marks**

The colors used to mark the quiz can be changed by redefining the commands `\checkColor`, `\crossColor` and `\correctColor` in the *preamble* or before. Below are the defaults:

```
\renewcommand\checkColor{["RGB", 0, .6, 0]}
\renewcommand\crossColor{color.red}
\renewcommand\correctColor{["RGB", 0, .6, 0]} % webgreen
```

The colors are inserted into the field using JavaScript, so the color definitions are in the color space of the JavaScript object model.

### • The ‘Correction’ Button

The ‘Correction’ button is defined by the `\eqButton` has two parameters.

```
\eqButton[mod_appear]{field_name}
```

The second parameter is the field name that contains the total score for the quiz (see the above examples). It also has one optional argument that can be used to modify the appearance of the button.

In addition to the optional parameter to modify the appearance of `\eqButton`, there is also a “global” mechanism for modifying the appearance of the button field. These are

**Global Modification:** `\everyButtonField` and `\everyeqButton`

The first one modifies the appearance of every quiz button field, the second can be used to modify all `\eqButtons`.

► See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” command.

### • The Score Field

The score field is the text field to which the quiz (and its underlying JavaScript) reports the score. This field can be constructed using the `\ScoreField` macro

```
\ScoreField[mod_appear]{field_name}
```

In the simplest case, `\ScoreField` takes one argument, as above, the `field_name` of the associated quiz. The expansion of `\ScoreField` produces a read-only text field that is 1.5 inches in width with a red border. The initial text that appears in the field is the expansion of the macro `\eqScore`. The expansion of `\eqScore` depends on the language option: `\eqScore` expands to ‘Score:’ by default, to ‘Punkte:’ for the german option and to ‘Score :’ for the french option.

The macro `\ScoreField` also has an optional parameter that can be used to modify the appearance of the text field. Should the document author want to change the basic look of the text field produced by `\ScoreField`, just introduce the changes through this optional parameter.

In addition to the optional parameter for modifying the appearance of the text field, `\ScoreField`, there is also a “global” mechanism for modifying the appearance of the button field. These are

**Global Modification:** `\everyeqTextField` and `\everyScoreField`

The first one modifies the appearance of every quiz text field, the second can be used to modify all `\ScoreFields`.

► See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” command.

**Begin Quiz** Answer each of the following. Passing is 100%.

1. What TeX System does Thomas Esser maintain?

MiKTeX                  csTeX                  teTeX                  fpTeX

2. What TeX System does Fabrice Popineau maintain?

MiKTeX                  csTeX                  teTeX                  fpTeX

3. What TeX System does Christian Schenk maintain?

MiKTeX                  csTeX                  teTeX                  fpTeX

**End Quiz**

The new part is the customized scoring and correction button. Here is a verbatim listing of the `\ScoreField` and `\eqButton` macros.

```
\ScoreField[\BC{0 0 1}]{qz:TeXc}%
  \eqButton[\BC{0 0 1}        % blue border color
  \CA{TeX}                    % Button text
  \RC{Users}                 % rollover text
  \AC{Group}                 % pushed text
  \textFont{TiRo}            % text font: Times Roman
  \textSize{10}              % text size: 10 point
  \textColor{0 0 1}         % blue text
  \W{1}\S{I}                 % border width 1, inset button
  ]{qz:TeXc}
```

✂ This example—as well as others—appears in `webeqtst.tex`, a test file that accompanies the **AcroTeX Bundle**.

## 21.8. `\titleQuiz` and `\fancyQuizHeaders`

✂ The demo file for these features is `title_quiz.tex`.

### • `\titleQuiz`

To assign a title to your quiz use `\titleQuiz`. This title appears just after the “Begin Quiz” button

**Begin Quiz Two Question MC** Answer each of these to the best of your ability.

1. Math is fun.

(a) True                  (b) False

2. Statistics is fun.

(a) True                  (b) False

**End Quiz**

The partial verbatim code for the above quiz follows:

```

\titleQuiz{Two Question MC}
\begin{quiz}{MyMCQuiz}\label{MyMCQuiz}
Answer each of these to the best of your ability.
\begin{questions}
...
\end{questions}
\end{quiz}\quad\ScoreField\currQuiz\eqButton\currQuiz

```

The format of the title is controlled by the command `\titleQuizfmt`, the default definition is `\newcommand\titleQuizfmt{\bfseries}`. The value of the `\titleQuiz` command can act as an argument for `\titleQuizfmt` if you redefine `\titleQuizfmt` to have an argument. For example,

```
\renewcommand\titleQuizfmt[1]{\bfseries(#1)}
```

Now, the title will appear in a set of parentheses. The title and formatting are enclosed in a `\mbox`, so font and style changes are localized to that `\mbox`.

If a quiz does not have a title, as defined by `\titleQuiz`, the title defaults to an empty title.

The definition of `\titleQuiz` is such that the command `\makeatletter` is executed before the argument for `\titleQuiz` is read. This means that the special character '@' can be used within the argument of `\titleQuiz`. For example, the `quiz` environment maintains a text macro `\quizCnt` which contains the current quiz count number. Thus you can say `\titleQuiz{Quiz \@quizCnt}`, the title of the quiz will include a short quiz number as well.

The `\titleQuiz` is used for short quizzes as well as with quizzes. It is possible to have different formatting for each of these two environments by using the `\ifQuizType` command. The command takes two arguments:

```
\ifQuizType{<quiz_code>}{<shortquiz_code>}
```

For example, we could define `\titleQuizfmt` as follows:

```
\renewcommand\titleQuizfmt{\ifQuizType{\bfseries}{\slshape}}
```

or with an argument, as mentioned above, you could define `\titleQuizfmt` with an argument, like so,

```
\renewcommand{\titleQuizfmt}[1]{\ifQuizType{\bfseries(#1)}{\slshape[#1]}}
```

#### • Cross-Referencing

Quizzes and shortquizzes conform to the cross-referencing system of  $\LaTeX$ . For example, we used the label `\label{MyMCQuiz}` in the last quiz, the one on page 97. The title of the quiz is **Two Question MC**. The code for the past two sentences is

```

For example, we used the label \verb!\label{MyMCQuiz}! in the last
quiz, the one on page~\pageref{MyMCQuiz}. The title of the quiz is
\nameref{MyMCQuiz}.

```

If the quiz does not have a title, as defined by the `\titleQuiz` command, a generic label of 'Quiz' is generated. The default label name is

```
\newcommand\eq@defaultQuizLabelName{Quiz}
```

This definition can be change using `\renewcommand`.

- `\fancyQuizHeaders`

The `\fancyQuizHeaders` command, a companion feature to `\titleQuiz`, enhances the labeling of the quiz in the solutions section of the document. If `\fancyQuizHeaders` is executed in the preamble, or prior to a quiz, the solution header has the following default format:

**Two Question MC: Question 1.** The answer is true, of course, math is fun!

The above assumes the title of the quiz is the one given in the previous section.

When a quiz has no title, the header reverts to **Solution to Quiz:** followed by the question number.

► You can turn off fancy quiz headers by using `\restoreDefaultQuizHeaders`.

You can modify the fancy headers with a combination of setting `\setsoInspace` and/or redefining the command `\eq@fancyQuizHeadersfmt`. First `\setsoInspace`: This is a hook into the space immediately following the header. To insert a `\medskip` after the header, for example, you can say `\setsoInspace{\par\medskip\noindent}` a `\medskip` is created after the solution header and the solution above appears as

**Two Question MC: Question 1.**

The answer is true, of course, math is fun!

The definition of `\eq@fancyQuizHeadersfmt` is

```
\def\eq@fancyQuizHeadersfmt{\protect\color{blue}\protect
  \textbf{\ifx\@titleQuiz\@empty\ifnum\@eqquestiondepth>0
    Solution to Quiz:\fi\else
    \@titleQuiz:\protect\
    \ifnum\@eqquestiondepth=0\else
    Question\fi
  \fi}\space
\ifcase\@eqquestiondepth
\ifx\@titleQuiz\@empty Solution to Quiz:\fi\or
\arabic{eqquestionnoi}.\or
\arabic{eqquestionnoi}(\alph{eqquestionnoi})\or
\arabic{eqquestionnoi}(\alph{eqquestionnoi})%
(\roman{eqquestionnoi})%
\fi}}%
```

Ugly isn't it. We need to set the header in all cases: with or without a title; multiple questions within a `questions` environment, or a single question with no `questions` environment.

A simple re-definition might be

```
\def\eq@fancyQuizHeadersfmt{\protect\color{blue}\protect
\textbf{\ifx\aeB@@titleQuiz@empty
\ifnum\@eqquestiondepth>0
Solution to Quiz\fi\else
\aeB@@titleQuiz\protect\
\ifnum\@eqquestiondepth=0\else Question%
\fi\fi\space
\ifcase\@eqquestiondepth
\ifx\aeB@@titleQuiz@empty Solution to Quiz\fi
\or\arabic{eqquestionnoi}.\or
\arabic{eqquestionnoi}(\alph{eqquestionnoi})\or
\arabic{eqquestionnoi}(\alph{eqquestionnoii})%
(\roman{eqquestionnoiii})%
\fi}}%
}
```

The colons (:) are removed and a `\\` inserted in the fourth line; that line now reads, in part, `\else\\Question\fi`, instead of `\else Question\fi`.

The results of these changes, in combination with the earlier change of

```
\seto\space{\par\medskip\noindent}
```

are

Two Question MC  
Question 1.

The answer is true, of course, math is fun!

Other (minor) changes such as color, style, font can be done in this way.

As a reminder, you can use the web package's switch, `\ifeqforpaper` to further refine these headers. You can have one style of solution headers for paper, and another style for the screen.

### 21.9. Adding Points to a Quiz

The questions on a quiz, especially a quiz meant for credit, may not have the same weight. A point scheme, therefore, has been created; several additional text fields in support have also been defined.

Here is a simple two question example to illustrate:

**Begin Quiz** Answer each of the following. Passing is 100%.

- (4<sup>pts</sup>) If  $\lim_{x \rightarrow a} f(x) = f(a)$ , then we say that  $f$  is...  
differentiable          continuous          integrable
- (6<sup>pts</sup>) Name *one* of the two people recognized as a founder of Calculus.

**End Quiz**

Answers:

Points:

Percent:

⚡ See the sample file `quizpts.tex` for a more elaborate version of this question, as well as the source code.

- `\PTs#1`: This macro takes one argument, the number of points to be assigned to the current problem. Place this command immediately after the `\item` in the questions environment. For example, in the above quiz we had

```
\item\PTs{6} Name \emph{one} of the two people recognized
as a founder of Calculus.
```

- `\PTsHook#1`: This macro, which takes one argument, can be used to typeset the points assigned. and is called by `\PTs`. The argument is what is to be typeset. The value assigned the current problem by `\PTs` is contained within the macro `\eqPTs`. In the quiz above, we had

```
\PTsHook{($\eqPTs^{\text{pts}}$)}
```

- There are three other commands that create text fields to display results from a quiz with points assigned:

- `\PointsField[#1]#2`: The number of points earned for the quiz, the total points are also reported. The parameter #2 is the base name of the quiz.
- `\PercentField[#1]#2`: The percentage score for the quiz. The parameter #1 is the base name of the quiz.
- `\GradeField[#1]#2`: The letter grade of the performance on the quiz. The parameter #2 is the base name of the quiz. The values placed in this field are determined by the macro `\eqGradeScale`.

- `\eqGradeScale`: This macro sets the grade scale of a quiz, the default definition is

```
\newcommand\eqGradeScale{"A", [90, 100], "B", [80, 90],
"C", [70, 80], "D", [60, 70], "F", [0, 60]}
```

The ways things are defined now, there can be only one grade scale per document. The value of `\eqGradeScale` is a matrix with an even number of elements. The odd numbered elements are the grades; the even number elements are intervals of percentages (percentages of the total number of points on the quiz). If the percentage of the score falls into a particular range, the corresponding grade is assigned.

Note, obviously, you can redefine this command. The letter grades do not actually have to be grades, they can be little messages to the student upon completion of the quiz.

```
\renewcommand\eqGradeScale{%
  "Excellent Work.",[90, 100],
  "Solid Effort.",[80,90],
  "Fair.",[70,80],
  "Needs improvement, better work expected.",[60,70],
  "Learning still in progress.",[0,60]
}
```

#### • `\negPointsAllowed`

The `answers` and `manswers` environments allow for partial credit, and even assigning of negative points for wrong answers (see ‘[The manswers Environment](#)’ on page 90); consequently, it is possible for a student taking a quiz to get a negative points for the quiz. By default `exerquiz` reports a point score of zero to `\PointsField` if the points are negative.

If you want `exerquiz` to report the true point score, use the command `\negPointsAllowed` in the preamble. This will allow the reporting of a negative point total.

✎ See the example file for this feature is `manswers.tex`. Compare the results of when the default behavior is used, versus the results when `\negPointsAllowed` is used in the preamble.

### 21.10. Floating a Quiz

The quiz structure is to have a ‘Begin Quiz’ button or link at the beginning of the quiz and an ‘End Quiz’ button or quiz at the end. In some settings, you might want to place the ‘Begin Quiz’ button on another page, with, perhaps, some instructions. For this purpose `exerquiz` defines three commands `\DeclareQuiz`, `\floatQuiz`, `\startQuizHere`, `\endQuizHere` and `\dockQuiz`.

The `\DeclareQuiz` command takes one argument, which is the name of the quiz, for example `\DeclareQuiz{myQuiz}`. This command takes it argument and creates a text macro `\currQuiz`, which expands to the quiz name. `\currQuiz` can then be used in the `quiz` environment command argument for the quiz name. This makes it easy to change the names without having to search through and make all changes.

The command `\floatQuiz` declares the current quiz, as set by the `\DeclareQuiz` command, is floating, which means the ‘Start Quiz’ and ‘End Quiz’ link-text or buttons are separated from their usual placement and are allowed to “float.” The third command `\startQuizHere` inserts the ‘Begin Quiz’ link text or form button, while the `\endQuizHere` command inserts the ‘End Quiz’ link or form button.

For example,

Solve each of the following problems. To begin the quiz, click on the button.

Answer these questions truthfully.

1. Do you understanding the concept of `\floatQuiz`?
  - (a) Yes
  - (b) No
2. Do you understanding the concept of `\dockQuiz`?
  - (a) Yes
  - (b) No

After you have finished the quiz, click on the `\floatQuiz` button. Click the `\dockQuiz` button if you wish and your quiz will be marked. Click on the choice with the green border to jump to the solution.

To return to the default behavior of the quiz environment place `\dockQuiz` following the quiz. Quizzes that come after the `\dockQuiz` will then have the standard quiz format.

▶ See the example file `floatquiz.tex` for the  $\LaTeX$  code of the above example.

## 22. Objective Style Questions

Beginning with version 2.0 of `exerquiz`, objective style questions can be posed. Single questions can be posed in the `oQuestion` environment, multiple questions can be placed in either the `shortquiz` or the `quiz` environments. This section discusses this type of question and all of its supporting commands.

### 22.1. Math and Text Questions

`Exerquiz` distinguishes between two types of open ended or objective questions:

1. A mathematical question that requires a mathematical expression as the answer.
2. A question that requires a text answer.

▶ The demo file `jquiztst.tex` is an important source of examples and instruction for the mathematical type question; additionally, the file `jtxttst.tex` has many examples for the text type question.

#### • The Mathematical Question

At this stage in the development of `exerquiz`, a (mathematical) question can be posed that requires an answer that is a function of one or more declared variables  $x$ ,  $y$ ,  $z$ , etc. Thus, when the declared variables  $x$ ,  $y$ ,  $z$  are given a value, the answer is reduced to a number.

For example, the answer to the question “Differentiate  $\frac{d}{dx} \sin^2(x)$ ”, is a function in one variable  $x$ , it can be evaluated numerically and can, therefore, be posed:

▶ Differentiate  $\frac{d}{dx} \sin^2(x) =$

See ‘`\RespBoxMath: The Math Question`’ on page 104 for details.

In contrast, consider the question: “Name the probability distribution popularly referred to as the ‘bell-shaped curve’”. The answer to this question cannot be reduced to a numerical value. This question can be posed as an **text objective question**, or, it does lend itself to a multiple choice question, however.

- **The Text Question**

You can also pose questions that require a text answer:

- ▶ Name the probability distribution popularly referred to as the “bell-shaped curve”.

See `\RespBoxTxt: The Text Question` on page 107 for details.

## 22.2. The oQuestion Environment

The `oQuestion` environment is a very simple environment for posing a *single* question and will be used in this section to discuss in detail the macros for posing mathematical and text open questions.

The syntax for the `oQuestion` environment is

```
\begin{oQuestion}{<field_name>}
<A math or text open ended question.>
\end{oQuestion}
```

The environment takes one required argument, a unique name for the question. This name, `field_name`, is used by other supporting macros.

- `\RespBoxMath: The Math Question`

The `\RespBoxMath` command is used for posing an objective question. This command must appear in the `oQuestion`, `shortquiz` or `quiz` environments. In this section we discuss only the `oQuestion` environment.

The following is a minimal example. Additional enhancements will be discussed in subsequent sections.

- ▶ Differentiate  $\frac{d}{dx} \sin^2(x) =$

The code for the above example is

```
\begin{oQuestion}{sine1}
\redpoint Differentiate $\dfrac{d}{dx} \sin^2(x) =
\RespBoxMath{2*\sin(x)*\cos(x)}{4}{.0001}{[0,1]}$
\end{oQuestion}
```

The `\RespBoxMath` need not appear in math mode. The definition of the `\redpoint` command, a command written for this document follows.

```
\newcommand\redpoint{\par\removeLastskip\vskip\medskipamount\noindent
\makebox[\parindent][l]{\large\color{red}$\blacktriangleright$}}
```

You can also pose multivariate questions as well, for example

$$\triangleright \frac{\partial}{\partial y} 4x^2y^3 =$$

The code for the above example is

```
\begin{oQuestion}{multivariate}
\redpoint $\dfrac{\partial}{\partial y} {4 x^2 y^3 }
= \RespBoxMath{12*x^2*y^2}(xy){4}{.0001}{[0,1]x[0,1]}$
\end{oQuestion}
```

See the file `multivar.tex` for more examples quizzes involving multivariate problems.

The algorithm used for determining the correctness of the answer entered by the user is very simple: The user's answer and the correct answer are evaluated at randomly selected points in an interval, then compared. If any of the comparisons differ by more than a pre-selected amount, an  $\epsilon$  value, if you will, the user's answer is declared incorrect; otherwise, it is considered correct.<sup>15</sup>

The command `\RespBoxMath` takes ten parameters, five optional and five required:

```
\RespBoxMath[#1]#2(#3)[#4]#5#6#7#8[#9]*#10
```

#### Parameters:

- #1** : Optional parameter used to modify the appearance of the text field. See the section entitled [The 'Correction' Button](#) for examples, and `exerquiz.dtx` for a listing of all controlling macros.
- #2** : The correct answer to the question. This must be a numerical value, or a function of one or more variable(s).
- #3** : An optional parameter, *delimited by parentheses*, that defines the independent variable; `x`, is the default value. Note that this parameter is set off by parentheses. For a multivariate question, just list the variables in juxtaposition, `(xyz)`.

Beginning with version 5.5 of `exerquiz`, an alternate method is to delimit with commas `(x,y,n)` and include the type of the variables `(r:x,r:y,i:n)`, where "r" means a real variable and "i" means an integer variable. When a type is not specified explicitly, "r" is assumed. The variables must be either of the old style (no commas, no typing) or the new style. Do not mix the styles.

See the [example](#) in ['Some Enhancements'](#) on page 110 of the section below and see the demo file `integer_test.tex` to demonstrate the new method for specifying variables, found in the `d\j{s}lib_examples` folder.

<sup>15</sup>The idea for evaluating user input in this way comes from Drs. Wlodzimierz Bryc and Stephan Pelikan of The University of Cincinnati.

**#4** : Optional, a named destination to the solution to the question. There are two forms: the `[mydest]` an explicit destination for the solution to the problem, or a `'*`, in which case, the name is `[\curr@quiz.\thequestionno]` is automatically assigned.

If this parameter appears, a solution must follow the question, enclosed in a `solution` environment.

**#5** : The number of samples points to be used, usually 3 or 4 is sufficient.

**#6** : Precision required, a *non-negative* “ $\epsilon$ ” value, if you will. If the precision is set to zero, `reldiffCompare` is used as the default compare, and the precision is set to a small positive value, `1E-14`, by default. The small value is may reset with `\defaultRDPrecision`. The default definition, `\defaultRDPrecision{1E-14}`.

**#7** : Parameters **#7** and **#8** are used to define the interval from which to draw the sample points. There are two forms: (1) **#7** is the left-hand endpoint of the interval and **#8** is the right-hand endpoint (the use of **#7** and **#8** in this form is deprecated); (2) the interval is defined by standard interval notation, `[a,b]`. For a multivariate question—one where parameter **#2** lists more than one variable, separate the intervals for each variable by a `'x'`, `[0,2]x[1,2]x[3,4]`.

**#8** : (1) **#8** is the right-hand endpoint of the interval (the use of this parameter is deprecated); (2) in the second case, **#8** is not used.

**#9** : This optional parameter is the name of a customized comparison function.

Beginning with version 5.5 of `exerquiz`, this argument can also be a JavaScript object with at most two properties: `priorParse` and `comp`. `priorParse` is used to insert additional JavaScript into `ProcResp` prior to processing the user's answer; this allows additional “filtering” of the user's response. The value of `priorParse` can either be a single function, or an array of functions. These functions take `UserAns` as its argument and return either `null`, if `UserAns` is not acceptable, or `true`, if it is ok for processing. The value of `comp` is the name of the function to be used to compare answers.

See the demo file `integer_tst.tex` for examples of usage.

**#10**: (Only detected if following an asterisk, `'*`) The name of a JavaScript function that is to be used to process the user input.

► For the above example,

```
\RespBoxMath{2*\sin(x)*\cos(x)}{4}{.0001}{[0,1]}
```

no optional parameter is specified. The correct answer written in valid JavaScript is given by `2*\sin(x)*\cos(x)`; evaluation of the user's answer is done by randomly selecting 4 points from the interval `[0,1]`; if the evaluation at any of the 4 points differs from the evaluation of the correct answer at the same point by more than  $\epsilon = 0.0001$ , the user's answer is considered wrong.

Once you choose the question to ask, you must then select the values of the parameters for `\RespBoxMath`.

► **Some Comments:**

1. The correct answer can be written either with valid JavaScript, or in the same syntax a user would enter the answer with. The functions and operators are pretty much as expected. See the demo file `jquiztst.tex` for some discussion how authors and users should enter their answers.
2. The interval from which the sample points are taken needs to be chosen with care. The interval must, obviously, be a subset of the domain of the answer function. Choose an interval away from any singularities the answer may have.

✎ See the file [jquiztst.pdf](#) for various examples of the math questions.

By using the optional first parameter, you can modify the appearance of the field “locally.” There is also a “global” mechanism as well:

**Global Modification:** `\everyeqTextField`, `\everyRespBoxMath`

The first one modifies the appearance of every quiz text field, and the second can be used to modify all fields created using `\RespBoxMath`.

► See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” command.

• **`\RespBoxTxt`: The Text Question**

You can also pose a question that takes a simple text response. The basic command for posing this type of question is `\RespBoxTxt`. Consider the example given earlier:

► Name the probability distribution popularly referred to as the “bell-shaped curve”.

The underlying JavaScript compares the user’s response against acceptable alternatives, as supplied by the author of the question. If there is a match, the response is deemed correct.

The code for this example is

```
\begin{oQuestion}{exTxt1}
\redpoint Name the probability distribution popularly
referred to as the ‘bell-shaped curve’.\
\RespBoxTxt{0}{0}{4}{Normal}{Normal Distribution}%
{Gaussian}{Gaussian Distribution}
\end{oQuestion}
```

The command `\RespBoxTxt` takes five or more parameters.

```
\RespBoxTxt[#1]#2#3[#4]#5<plus listing of alternatives>
```

**Parameters:**

- #1** : Optional parameter used to modify the appearance of the text field. See the section entitled [The ‘Correction’ Button](#), page 96 for examples, and `exerquiz.dtx` for a listing of all controlling macros.
- #2** : This required parameter is a number that indicates the filtering method to be used. Permissible values of this parameter are
- 1**: (The default) The author’s and user’s answers are not filtered in any way. (Spaces, case, and punctuation are preserved.)
  - 0**: The author’s and user’s answers are converted to lower case, any white space and non-word characters are removed.
  - 1**: The author’s and user’s answers are converted to lower case, any white space is removed.
  - 2**: The author’s and user’s answers are stripped of any white space.
- See the JavaScript function `eqFilter` in `exerquiz.dtx` for the program code details. Additional filtering options may be added.
- #3** : This parameter a number that indicates the compare method to be used. Permissible values of this parameter are
- 0**: (The default) The author’s and user’s answers are compared for an exact match. (These answers are filtered before they are compared.)
  - 1**: The user’s response is searched in an attempt to get a substring match with the author’s alternatives. Additional comparison methods may be added.
- See the JavaScript function `compareTxt` in `exerquiz.dtx` for the program code details.
- #4** : Optional, a named destination to the solution to the question. If this parameter appears, then a solution must follow the question, enclosed in a `solution` environment. If the fourth parameter is a ‘\*’, then an automatic naming scheme is used instead.
- #5** : This required parameter is the number of alternative answers that are acceptable. The alternative answers are listed immediately after this parameter. (The example above specified that 4 alternatives follow.)

✎ See the file [jtxttst.pdf](#) for examples of the differences between various combinations of filtering rules and comparison methods.

By using the optional first parameter, you can modify the appearance of the field “locally”. There is also a “global” mechanism as well:

**Global Modification:** `\everyeqTextField` and `\everyRespBoxTxt`

The first mechanism modifies the appearance of every quiz text field, the second can be used to modify all fields created using `\RespBoxTxt`.

► See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” command.

- **\RespBoxTxtPC: The Text Question with Partial Credit**

Exerquiz can now create text fill-in questions that awards credit each time one of the key words are found in the student's input string.

The command `\RespBoxTxtPC` is the one that creates a text fill-in question. Its syntax is

```
\RespBoxTxtPC[#1]#2[#3]#4[<num1>]{<word1>}. . . [ <num_n> ] { <word_n> }
```

The new command `\RespBoxTxtPC` behaves differently than `\RespBoxTxt`. JavaScript performs a regular expression search for each word listed (the word, can be a regular expression), if the word is found (the search is successful), the total credit for this problem is incremented by the amount associated with the word.

**Parameters:**

- #1: Optional parameter used to modify the appearance of the text field.
- #2: This required parameter is a number indicating the filtering method to be used. Permissible values of this parameter are
  - 1: (The default) The author's and user's answers are not filtered in any way. (Spaces, case, and punctuation are preserved.)
  - 0: The author's and user's answers are converted to lower case, any white space and non-word characters are removed.
  - 1: The author's and user's answers are converted to lower case, any white space is removed.
  - 2: The author's and user's answers are stripped of white space.
  - 3: Same as -1, but a case insensitive search is performed. *This is the recommended value for this function.*

See the JavaScript function `eqFilter` in `exerquiz.dtx` for program code details. Additional filtering options may be added.

**Recommendation:** For `\RespBoxTxtPC`, use either -1 (case sensitive search) or 3 (case insensitive search).

- #3: Optional, a named destination to the solution to the question. If this parameter appears, then a solution must follow the question, enclosed in a `solution` environment. If the third parameter is a '\*', then an automatic naming scheme is used instead.
- #4: This required parameter is the number of alternative answers that are acceptable. The alternative answers are listed immediately after this parameter.

The parameters that follow #4 are of the form `[<num>]{<word>}`. The `[<num>]` is the amount of credit the user gets if his answer contains `<word>`. Actually, `<word>` can be a regular expression so that a more sophisticated search criteria can be set up. This is illustrated in the examples above, and explained below.

A `\RespBoxTxtPC` question is deemed correct (and marked with a green rectangle, in the case of a quiz) if at least one of the words is found.

If a word does not have a partial credit [`<num>`] before it, that word has a partial credit of zero, and may as well not be included in the list of words.

- ✂ The demo file for this section is `jttxtst_pc.tex`. The file contains extensive examples, and discusses techniques of searching using regular expressions.

### 22.3. Some Enhancements

There are several enhancements to the math (using `\RespBoxMath`) and text (using `\RespBoxTxt`) open-ended question beyond the minimal examples given earlier. These enhancements can be used within the `oQuestion`, the `shortquiz`, and the `quiz` environments.

- **Including an Answer Key with `\CorrAnsButton`**

The correct solution can be included in the question as well; just include the command `\CorrAnsButton`. This command takes one parameter, the correct answer that will be viewed when the user clicks on the button.

The example below also illustrates the (optional) third parameter of `\RespBoxMath`. Here we pose the question in the variable  $t$  rather than the default variable of  $x$ .

- ▶ Differentiate

$$\frac{d}{dt} \sin^2(t) =$$

The listing follows:

```
\begin{oQuestion}{sine2}\\[1ex]
\redpoint Differentiate $\dfrac{d}{dt} \sin^2(t) = $
\RespBoxMath{2*\sin(t)*\cos(t)}(t){4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*\sin(t)*\cos(t)}
\end{oQuestion}
```

The `\CorrAnsButton` takes one parameter, the correct answer. This answer is (usually) the same as the one given as the second argument (the optional argument is the first) in the `\RespBoxMath` command.

- ▶ The `\CorrAnsButton` also controls access to the (optional) solution, see the next section.

- **Including a Solution**

In addition to a correct answer, you can also include a solution to the question. Insert the optional fourth parameter—fourth for both `\RespBoxMath` and `\RespBoxTxt`—into the parameter list giving the name of the destination to the solution. Follow the question by a `solution` environment containing the solution.

The user `Shift-Clicks` on the `\CorrAnsButton` to jump to the solution.

- ▶ Differentiate

$$\frac{d}{dt} \sin^2(t) =$$

The listing follows:

```
\begin{oQuestion}{sine3}\[1ex]
\redpoint Differentiate $\dfrac{d}{dt} \sin^2(t) = $
\RespBoxMath{2*\sin(t)*\cos(t)}(t)[sine3]{4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*\sin(t)*\cos(t)}
\begin{solution}
\[
\frac{d}{dx}\sin^2(x) = 2\sin(x)\cos(x) = \sin(2x)
\]
\end{solution}
\end{oQuestion}
```

- ▶ The `\CorrAnsButton` works the same way for the `shortquiz` and the `quiz` environments.

Actually, `\CorrAnsButton` has an optional argument. The full syntax is

```
\CorrAnsButton{<display_answer>}[*{<JS_function>}]
```

When the `*` is detected following the `<display_answer>`, the `<JS_function>` is used to evaluate `<display_answer>`, before displaying it. There is one built-in function called `EvalCorrAnsButton`, the definition of which is

```
function EvalCorrAnsButton(fieldname,theanswer)
{
  theanswer = eval(theanswer);
  DisplayAnswer(fieldname,theanswer);
}
```

Other JavaScript functions may be defined using the `insDLJS` environment, then passed to `\CorrAnsButton` with the optional argument.

An example of this optional parameter is

- ▶ Perform the indicated operation, and enter result into the box.

$$9 + 8 =$$

The listing follows:

```
\begin{oQuestion}{add1}
\redpoint Perform the indicated operation, and enter
result into the box.\[1ex]
$ 9 + 8 = $
\RespBoxMath{9 + 8}{1}{.0001}[[0,1]]\kern1bp
\CorrAnsButton{9 + 8}*{EvalCorrAnsButton}\kern1bp
\sqTallyBox
\end{oQuestion}
```

- **Including a Tally Box**

The macro `\sqTallyBox` is used to keep a running total of the number of wrong answers a user has entered into the response box.

For example,

- ▶ Differentiate

$$\frac{d}{dx} \sin^2(x) =$$

The listing follows:

```
\begin{oQuestion}{sine4}
\redpoint Differentiate\[\[1ex]
$\dfrac d{dx} \sin^2(x) =$
\RespBoxMath{2*\sin(x)*\cos(x)}{4}{.0001}{0}{1}\kern1bp
\CorrAnsButton{2*\sin(x)*\cos(x)}\kern1bp
\sqTallyBox
\end{oQuestion}
```

- ▶ The tally box can be used within the `oQuestion` and `shortquiz` environments; in the `quiz` environment, no tally box is used.

- **Clearing the Fields**

For the `oQuestion` and the `shortquiz` environments, you can clear the response box fields by placing insert `\sqClearButton`.

- ▶ Differentiate

$$\frac{d}{dx} \sin^2(x) =$$

The listing follows:

```
\begin{oQuestion}{sine5}
\redpoint Differentiate\[\[1ex]
$\dfrac d{dx} \sin^2(x) =$
\RespBoxMath{2*\sin(x)*\cos(x)}{4}{.0001}{0}{1}%
\CorrAnsButton{2*\sin(x)*\cos(x)}\kern1bp
\sqTallyBox\kern1bp\sqClearButton
\end{oQuestion}
```

Notice that I've inserted a `\kern1bp` to separate the two fields `\sqTallyBox` and `\sqClearButton`. This is to keep their borders from overlapping.

## 22.4. More on Math Fill-in Questions

This section covers rules for author and user input into a math fill-in question.

### • Comments on Authoring

When authoring a math fill-in, that is, when using `\RespBoxMath` to construct a question, several things must be kept in mind.

- The **second parameter** of `\RespBoxMath`,

$$\backslash\text{RespBoxMath}[\#1]\#2(\#3)[\#4]\#5\#6\#7\#8[\#9]*\#10$$

is the correct answer parameter. You need not compute the correct answer, you can let the JavaScript of `exerquiz` do that. For example, if the answer is  $\sqrt{2}$  you can enter `sqrt(2)`. All the rules of entering user input outlined in the next section on ‘Comments on User Input’ holds for this second parameter as well.

If you choose to restrict user input in some way, perhaps using the `limitArith` option of `dljslib`, these limitations only apply to the user input, not to the author’s input in parameter #2. See ‘`limitArith`’ on page 151 for more information on how to limit arithmetic operations.

- The question must be posed so as to make it clear to the student what is expected by way of input into the text field, an expression, an equation, and so on.

### • Comments on User Input

When responding to a Math Fill-in question, the student responds by typing in an answer. The following notation is expected by `exerquiz`.

- Use `*` to indicate multiplication: Type `4*x` for  $4x$ .

If the `ImplMulti` option of the `dljslib` package is taken, then the student need only type in `4*x` for  $4x$ . `Exerquiz` attempts to insert the multiplication symbol `*` by parsing the user input. For information on, see the section on ‘`ImplMulti`’ on page 142.

- Use `^` to indicate powers: Type `4*x^3` for  $4x^3$ ; `12*x^-6` for  $12x^{-6}$ . Again, if the `ImplMulti` option is taken, entering `12x^-6` is sufficient.
- Use parentheses to delimit the argument of a function; that is, type `sin(x)`, rather than `sin x`.
- Use parentheses to define the *scope* of an operation: Type `4*x*(x^2+1)^3` for  $4x(x^2 + 1)^3$ ; `4^(2*x+1)` for  $4^{2x+1}$ ; `(sin(x))^2` for  $(\sin(x))^2$ . *Do not* type an expression like `sin^2(x)` for  $\sin^2(x)$ , type `(sin(x))^2` instead.

If the `ImplMulti` option of the `dljslib` package is taken, the use of `*` is unnecessary, but also the simplified notation for powers of functions is supported, now you can type `sin^2(x)` for  $\sin^2(x)$ .

- Brackets `[ ]` or braces `{ }` can be used to delimit a mathematical expressions.
- Functions that are recognized:

- Trig:  $\sin$ ,  $\cos$ ,  $\tan$ ,  $\cot$ ,  $\sec$ ,  $\csc$ .
- Inverse Trig:  $\arcsin$  (or  $\text{asin}$ ),  $\arccos$  (or  $\text{acos}$ ),  $\arctan$  (or  $\text{atan}$ ).
- Logarithms:  $\ln$  for natural logs, and  $\log$  for common logs. For example,  $\ln(x)$  or  $\log(x)$ .
- Exponential: The natural exponential function,  $e^x$ , can be entered as  $\text{exp}(x)$  or as  $e^x$ .
- The absolute function,  $\text{abs}(\cdot)$  can also be written in the usual way  $|\cdot|$ ; thus, you can type either  $\text{abs}(x)$  or  $|x|$ .
- Misc.:  $\text{sqrt}$ , usage  $\text{sqrt}(x)$  for  $\sqrt{x}$  (or, use exponential notation:  $x^{(1/2)}$ ).  
When the `combinatorics` option of `dljslib` is taken, the binomial coefficient function  $C(n, r)$ , the permutation function  $P(n, r)$ , and the factorial function  $\text{fact}$ , are defined. See '`combinatorics`' on page 153 for additional details.

A variation of these listings should appear as instructions to the student on how to enter math fill-in questions.

When a user enters a response, some attempt is made to determine whether the response is a valid mathematical expression. For example, if  $\text{san}(x)$  is entered, the function 'san' will not be recognized as a valid mathematical function; an error message is generated, and the user is not penalized for a possible typing error. The JavaScript routines will also check for unbalanced parentheses; thus,  $((x^4+1) + \sin(x)^2)$  will be flagged as a syntax error.

**Important:** The student must use the variables declared for the problem when entering mathematical expressions. If the problem statement involves the variable  $x$ , then  $x$  must be used; if the problem statement uses  $t$ , then  $t$  must be used. To enter an expression using  $t$  when an expression with variable  $x$  is expected will, no doubt, result in missed problem.

### 22.5. The shortquiz Environment

The objective question (with or without the presence of a correction box, `\corrAns-Button` or a tally box `\sqTallyBox`) can be mixed in with multiple choice questions.

Solutions to the questions can also be included using a `solution` environment. Click on the "Ans" button to get the answer to a question; shift-click on the "Ans" button to get the solution.

**Quiz** Answer each of the following. Passing is 100%.

1. If  $f$  is differentiable, then  $f$  is continuous.
  - (a) True
  - (b) False
2.  $\frac{d}{dx} \sin^2(x) =$
3. Name *one* of the two people recognized as a founder of Calculus.

- When using objective questions within a `shortquiz` environment, you must give a unique field name as an optional argument of the environment. The listing of this example follows:

```

\begin{shortquiz}[oQsq] % <-- unique field name
Answer each of the following. Passing is 100%.
\begin{questions}

\item If  $f$  is differentiable, then  $f$  is continuous.
\begin{answers}{4}
\Ans1 True & \Ans0 False
\end{answers}\hsfill\sqTallyBox

\item  $\frac{d}{dx} \sin^2(x) =$ 
\RespBoxMath{2*\sin(x)*\cos(x)}[sinsqx]{4}{.0001}{0}{1}%
\hsfill\CorrAnsButton{2*\sin(x)*\cos(x)}%
\kern1bp\sqTallyBox
\begin{solution}
\[
\frac{d}{dx} \sin^2(x) = 2 \sin(x) \cos(x) = \sin(2x)
\]
\end{solution}

\item Name one of the two people recognized
as a founder of Calculus.\vadjust{\kern3pt}\newline
\RespBoxTxt{2}{0}[newton]{5}{Isaac Newton}{Newton}{I. Newton}%
{Gottfried Leibniz}{Leibniz}\hsfill
\CorrAnsButton{Isaac Newton or Gottfried Leibniz}%
\kern1bp\sqTallyBox
\begin{solution}
Yes, Isaac Newton and Gottfried Leibniz are considered
founders of Calculus.
\end{solution}
\end{questions}
\end{shortquiz}
\begin{flushright}
\sqClearButton\kern1bp\sqTallyTotal %<-- total tally
\end{flushright}

```

#### Example Notes:

- Note the optional argument gives this collection of questions a common base name. All supporting macros use this name.
- The named destination to the solution is entered with parameter #5 of the command `\RespBoxMath`, and with parameter #4 of `\RespBoxTxt`.

- In this example, another built-in macro, `\sqTallyTotal` was used. This macro creates a text field that accumulates the totals of all the tally boxes.
- ▶ The `shortquiz` environment can also be used for a single objective question. Just don't use the `questions` environment within.

```
\begin{shortquiz}[anExample]
< an objective style question >
\end{shortquiz}
```

## 22.6. The quiz Environment

Objective questions can be mixed in with multiple choice questions within the `quiz` environment. When posing an objective style question in the `quiz` environment, use `\RespBoxMath` and `\RespBoxTxt`, and optionally include the `\CorrAnsButton` button and the `\AnswerField` text field.

Since the evaluation of the quiz is delayed until the user has finished the quiz, the `\sqTallyBox` macro is not applicable here.

Answer each of the following. Passing is 100%.

1. If  $f$  is differentiable, then  $f$  is continuous.  
True                      False
2.  $\frac{d}{dx} \sin^2(x) =$
3. Name *one* of the two people recognized as a founder of Calculus.

Answers:

- ▶ The buttons created by `\CorrAnsButton` are hidden until the user ends the quiz (and gets scored) and clicks on the corrections button (`\eqButton`). The `\CorrAnsButton` should not be included if there is no `\eqButton`.
- ▶ Pressing on the “Ans” button populates the text field created by `\AnswerField` with the author's correct answer. If there is a solution to the problem, the “Ans” button is outlined in green. Shift-click on the “Ans” button to jump to the solution.
- ▶ The `quiz` environment requires a field name. This same name is used by the objective style question as well.

The listing for the above example follows.

```
\begin{quiz*}{oQq}
Answer each of the following. Passing is 100%.
\begin{questions}

\item If  $f$  is differentiable, then  $f$  is continuous.
```

```

\begin{answers}{4}
\Ans1 True & \Ans0 False
\end{answers}

\item $\displaystyle\frac{d}{dx} \sin^2(x) = $
\RespBoxMath{2*\sin(x)*\cos(x)}{4}{.0001}{0}{1}%
\hfill\CorrAnsButton{2*\sin(x)*\cos(x)}%

\item Name \emph{one} of the two people recognized
as a founder of Calculus.\vadjust{\kern3pt}\newline
\RespBoxTxt{2}{0}[leibniz]{5}{Isaac Newton}{Newton}{I. Newton}%
{Gottfried Leibniz}{Leibniz}\hfill
\CorrAnsButton{Isaac Newton or Gottfried Leibniz}
\begin{solution}
Yes, Isaac Newton and Gottfried Leibniz are considered
founders of Calculus.
\end{solution}
\end{questions}
\end{quiz*}\quad\ScoreField{oQq}\eqButton{oQq}

\noindent Answers: \AnswerField{oQq}

```

The `\AnswerField` has the following syntax:

```
\AnswerField[<field_opts>]{<quiz_name>}
```

The parameter `<field_opts>` is the typical optional parameter for all eform fields; it is used to change the appearance of the field. The parameter `<quiz_name>` is the name of the quiz this field will hold the answers for.

You may have `\AnswerField` commands on several pages—possibly for different quizzes. By using the command `\resetAnsFieldOnClose`, the answer field for the current quiz will be cleared when the page containing it closes. You can revert back to the default behavior (answer fields not cleared) by executing `\noResetAnsFieldOnClose`.

► There are some additional grade reporting and statistical fields defined: `exerquiz` defines `\PointsField`, `\PercentField`, and `\GradeField`. The demo file `quizpts.tex` illustrates these commands; see also the section below entitled [Adding Points to a Quiz](#).

#### • The Prompt Button

In addition to the `\CorrAnsButton`, the document author can provide a prompt button (probably not the best descriptive term).

For some quizzes, the author might want to ask a series of questions where the answer to one question depends on the correct answer of a previous question. In this situation, you'd like to provide the correct answer so the student can make a good run at the next question. The `\@PromptQuestion` not only provides the answer to the question but it also makes the corresponding math fill-in read only, so that the student cannot change the answer already provided.

Ideally, the student first enters an answer, and once satisfied with the answer, can then get the correct answer for future questions.

- ▶ See the demo file `prompt_tst.tex` for an example of usage.

#### • Grouped Math/Text Fill-in Questions

Exerquiz defines a grouping environment, `mathGrp`, for math fill-in and *text fill-in* questions where the response to the question might require entering text into multiple math fill-in fields.

When you use the `mathGrp` environment to enclose a set of related math questions, you need to the `\CorrAnsButtonGrp` button, instead of the `\CorrAnsButton` button. The required argument for this button is a comma delimited list of the answers that appear within the grouped questions. The answers should be listed in the same order  $\TeX$  processes the math (or text) questions. The group of questions is processed as if it were a single question.

- ▶ For example...

(3<sup>pts</sup>) Compute the following cross product:

$$(3\vec{i} - 2\vec{j}) \times (\vec{i} + 5\vec{k}) = \quad \vec{i} + \quad \vec{j} + \quad \vec{k}$$

$\underbrace{\hspace{10em}}$   
ScoreField

$\underbrace{\hspace{10em}}$   
PointsField

Ans:

#### Notes:

- If you miss any one of the three answers, the `ScoreField` reports back ‘Score: 0 out of 1’. There is only one question there, to get it correct, you must answer all three inputs correctly.
- Points can be assigned to the individual responses and a score is given based on the validity of the inputs and the corresponding points. There is a default JavaScript function that scores the results. The document author can define a custom JavaScript function to have a more “exotic” method of evaluating the group. See the test file `grp_test.tex` for details.
- Notice that after you take the quiz and click on “Correct” button, the “Ans” button appears (as usual). If you click repeatedly on this “Ans” button, you can cycle through all answers to this question; the response box is highlighted (or put in focus) and the answer appears in the answer field provided.
- ▶ See the demo file `grp_test.tex` for the source code of this example, as well as more technical details of the `mathGrp` environment.

## 22.7. Modifying Form Elements

All form elements have a first optional parameter for modifying their appearance, and they have an associated “every” command for global modifications as well.

Global Modification Commands	
For the shortquiz Environment	
<code>\Ans</code>	<code>\everysqRadioButton</code>
<code>\sqTallyBox</code>	<code>\everysqTallyBox</code>
<code>\sqTallyTotal</code>	<code>\everysqTallyTotal</code>
<code>\sqClearButton</code>	<code>\everysqClearButton</code>
For the quiz Environment	
<code>\useBeginQuizButton</code>	<code>\everyBeginQuizButton</code>
<code>\useEndQuizButton</code>	<code>\everyEndQuizButton</code>
<code>\Ans</code>	<code>\everyqRadioButton</code>
<code>\ScoreField</code>	<code>\everyScoreField</code>
<code>\eqButton</code>	<code>\everyeqButton</code>
<code>\AnswerField</code>	<code>\everyAnswerField</code>
<code>\PointsField</code>	<code>\everyPointsField</code>
<code>\PercentField</code>	<code>\everyPercentField</code>
<code>\GradeField</code>	<code>\everyGradeField</code>
For Both Environments	
<code>\RespBoxMath</code>	<code>\everyRespBoxMath</code>
<code>\RespBoxTxt</code>	<code>\everyRespBoxTxt</code>
<code>\CorrAnsButton</code>	<code>\everyCorrAnsButton</code>

- In addition to these, there are other “every” commands that effect the appearance of the various buttons and text fields. The two commands `\everyeqButtonField` and `\everyeqTextField` are executed before every `exerquiz` button and text field (respectfully). These can be used to give a general uniform appearance for all the short quiz or quiz form elements; use the more specific version, as listed in the above table, to make additional refinements in appearance.
  - See the document [eFormMan.pdf](#) on *eForm Support* for complete documentation on how to modify a field using the optional first argument, and how to use the “every” mechanism.

## 22.8. Evaluating Equivalent Expressions

The `\RespBoxMath` macro, and its underlying JavaScript code, is designed to determine the correctness of an answer, numerical or symbolic. We can pose such questions as “find the derivative” of this function or “find the (definite or indefinite) integral” of that function, and so on. This is all accomplished by randomly choosing numbers from a specified interval and evaluating both the user’s and author’s answer to the question; if they are close to each other, then the user’s answer is called correct. The system works pretty well.

The `exerquiz` scheme does not perform well with such questions as “fully simplify the radical  $\sqrt{72}$ ,” or “factor the polynomial  $2x^2 - 4x + 2$ ,” in both cases, the student can simply type in the given expression and `exerquiz` will call them equal. Indeed it is true that  $\sqrt{72}$  equates to  $\sqrt{72}$  and the polynomial  $2x^2 - 4x + 2$  equates to  $2x^2 - 4x + 2$ , as is shown by evaluating randomly selected numbers from an interval, `exerquiz-style!`

To illustrate, we use the problem of simplifying  $\sqrt{72}$ .

**Quiz** Simplify the following radical expression by factoring out all perfect squares.

1. Simplify  $\sqrt{72}$  by factoring out all perfect squares.

$$\sqrt{72} =$$

`Exerquiz` performs its magic by a combination of the usual `exerquiz` processing and new pre- and post-parsing techniques, where appropriate code is passed through of the ninth parameter of `\RespBoxMath`.

The verbatim listing follows.

```

1  \item Simplify  $\sqrt{72}$  by factoring out all perfect squares.
2  \begin{equation*}
3    \sqrt{72} = \RespBoxMath{6sqrt(2)}{1}{.0001}{[0,1]}[{}%
4      priorParse:\Array(nodect,%
5        \preDenyForm(%
6          /sqrt\refac(\reany+)\redigit+/, \Msgii),%
7        \preReqForm(%
8          /\rechrclass(+)*\redigit+sqrt\refac(\reany+)/, \Msgi)%
9      ),%
10     postParse:\Array(\postDenyForm(/sqrt(\refac(72)|\refac(8)))/)%
11   ]]\CorrAnsButton{6sqrt(2)}\kern1bp\sqTallyBox
12 \end{equation*}

```

It becomes a bit complicated here. :-) The ninth parameter can take a JavaScript object as its value (The object is opened on line (3) with a left brace (`{`) and is closed in line (11) with a right brace (`}`). This object has three properties: `comp` for defining the compare function (not used in this example); `priorParse`, which is used to inspect the user’s input to filter out unwanted responses prior to evaluation; `postParse` is used to look at the user’s answer *after* the `exerquiz` evaluation of random points stage. At the `postParse` stage, it has already been determined if the user’s answer is correct or incorrect, `postParse` only applies to answers that are *correct*.

**The `priorParse` property.** See also the (brief) description of `priorParse` as part of the [ninth parameter](#), page 106. There are new two built-in filter functions which are accessed through the helper commands `\preDenyForm` and `\preReqForm`. As shown in the verbatim text, they are both enclosed in `\Array()`.

- `\preDenyForm`: The idea behind this filter function is to define regular expressions which if the pattern is present, processing would stop for the user input, and an alert message would appear; that is, we *deny* the user the use of certain defined

patterns. There is no penalty, the user has a chance to change his/her answer. `\preDenyForm` takes up to two arguments, enclosed in parentheses:

```
\preDenyForm(regex|array_regexps, cMsg)
```

The first argument is a regular expression or an array of regular expressions. The second argument, `cMsg`, is a string that will be displayed in the alert box if the pattern or patterns appear in the user's answer.

When the first argument is an array of regular expressions, the same message (`cMsg`) will appear if any one of the patterns is present. The command may be repeated using additional regular expressions with different warning messages.

Let's have a look at the above example,

```
4 priorParse:\Array(nodect,%
5   \preDenyForm(%
6     /sqrt\refac(\reany+)\redigit+/, \Msgii),%
7   \preReqForm(%
8     /\rechrclass(+)*\redigit+sqrt\refac(\reany+)/, \Msgi)%
9   ),%
```

In line (4) we begin with `\Array` followed by `djslib` filter function `nodect`, this function denies the use of decimal numbers, it is important in this problem but not our central focus. In line (5) we begin with `\preDenyForm`, the arguments are on line (6). The correct answer is  $6\sqrt{2}$ , we want to force the student write the answer in this form, not, for example, as  $\sqrt{2}6$  or  $\sqrt{2}*6$ , while correct, it bad form. (We will deny them bad form!) This will deny them even if not correct, for example, if the student enters  $\sqrt{3}4$ .

To make it easier on the document author, I've defined a series of "helper commands," and oh, by the way, you should know how to use regular expressions. Let's now examine line (6). First argument is a single regular expression, the second is a command that defines the string message to be exhibited if this pattern is found (try it on the working example above to see what the message is).

Let's now look at the regular expression (`/.../` in line (6)).

```
6 /sqrt\refac(\reany+)\redigit+/
```

We take each in turn.

- `sqrt\refac(\reany+)`: we search for the string `sqrt` followed by the pattern `\refac(\reany+)`. `\refac` defines literal parentheses, within these parentheses can be anything `\reany+`. The whole sequence expands to the pattern `'sqrt\(\.+\\)`'.
- `\redigit+`: This command is a macro for one or more digits, it expands to `'\d+'`.
- Summarizing, we deny the student from typing in a square root of anything followed by one or more digits.

- `\preReqForm`: This is the positive version of `\preDenyForm`. It uses the same syntax

```
\preReqForm(regexp|array_regexps, cMsg)
```

The first argument is a regular expression or an array of regular expressions, the patterns are ones that we wish to see in the user's answer. If the user's answer does not fit into any of the patterns, an alert is emitted using the optional string declared in the second argument, `cMsg`. The second argument is optional, there is a default, generic string message that will be shown.

As with `\preDenyForm`, let's take a close look at the regular expression.

```
\rechrclass(+)*\redigit+sqrt\refac(\reany+)/
```

- `\rechrclass(+)*` defines a character class consisting of the characters '+' or '-' the \* means zero or more occurrences. (Probably I should use ? instead of \*, meaning 0 or 1 occurrences.) It expands to '[+-]\*'.
- `\redigit+` describes one or more digits, this would expand to '\\d+'.
- `sqrt\refac(\reany+)` is the same as before, it describes the square root of anything expression.

The `\preReqForm` only searches for the specified patterns. For example, the responses `sin(PI/2)6sqrt(2)` and `6sqrt(2)x` pass both the `\preDenyForm` (no digits follow the square root), and pass the `\preReqForm` because both answers have (or do not have) the patterns; in the former case, the answer is judged correct, and in the latter case, incorrect. You can try to improve the pattern like so

```
\rebstr\rechrclass(+)*\redigit+sqrt\refac(\reany+)\reestr/
```

The `\rebstr` (begin string anchor) expands to '^' and represents the beginning of the line, while `\reestr` (end string anchor) expands to '\$' represents the end of the line. In this case, the user's answer is required to begin with an optional plus or minus, followed by one or more digits, followed by a square root, followed by the end of the line.

Here is that same short quiz with the above changes (and \* replaced by ?).

**Quiz** Simplify the following radial expression by factoring out all perfect squares.

1.  $\sqrt{72}$

$$\sqrt{72} =$$

Now try `sin(PI/2)6sqrt(2)` or `6sqrt(2)x`, and other aberrant variations.

**The postParse property.** After the priorParse stage, exerquiz checks the answer for correctness, against the author’s answer, using randomly selected numbers. After that comes the postParse event. The syntax is the same except there is no warning message.

```
\preReqForm(regex|array_regexps)
```

If the user’s *answer is correct* at this stage *and* fits into one of the patterns as defined by the regular expressions, the answer will then be marked wrong.

Let’s look at the post parse code for this example, from line (10) on page 120, we see,


```
10 postParse:\Array(\postDenyForm(/sqrt(\refac(72)|\refac(8))/))%
```

Even though there is not an array of filter functions, a filter function with arguments must always be enclosed in an array.

Within the array, we execute our `\postDenyForm`, it has only one argument, a single regular expression: `/sqrt(\refac(72)|\refac(8))`.<sup>16</sup> There are three “correct answers” to this problem, they are  $\sqrt{72}$ ,  $3\sqrt{8}$ , and  $6\sqrt{2}$ , this latter one being the only correct answer to this question, for the instructions were to factor out *all* perfect squares. The regular expression here catches any answer containing  $\sqrt{72}$  or  $3\sqrt{8}$  and marks it wrong. :- (

#### List of RegExp helper macros.

```
\def\refac(#1){\(#1\)}\def\rediv{\\\/}\def\repow{\^^}
\def\redigit{\d}\def\rechrclass(#1){[#1]}\def\reany{.}
\def\remul{[\^*]}\def\rebstr{\^^}\def\reestr{[\^$]}
```

 Additional examples will (eventually) appear on the [AcroTeX Blog](#).

## 23. Randomizing the Multiple Choices

Beginning with version 6.1 of exerquiz, the choices of a multiple choice question can be randomized. The `random.tex` macro file by Donald Arseneau is used for this purpose.

 The demo file for this section is `randomize.tex`.

The randomization is only allowed if the `allowrandomize` option of `exerquiz` is used; otherwise, no randomization can occur.

The randomization is only defined for choices listed between the pair `\bChoices` and `\eChoices` (see ‘[The \bChoices/\eChoices Commands](#)’ on page 74). The command `\bChoices` now takes two optional key-value arguments:

- `nCols=<number>`: The number of columns to create, as described on 74. You can also use the old style by specifying just `<number>`. Thus, the commands `\bChoices[nCols=2]` and `\bChoices[2]` are equivalent.

<sup>16</sup>The use of alternation here is not needed, an answer involving  $\sqrt{72}$  would have been filtered out by the `\preReqForm` filter, do you know why?

- `random=<true|false>`: Specify this option to randomize choices. You can use the key word `random` instead of `random=true`. For example, the following commands will randomize the choices, `\bChoices[random]`, `\bChoices[nCols=2,random]`, or `\bChoices[2,random]`. The default is to not randomize the choices.

The following is an example of the `random` option of `\bChoices`.

```
\begin{shortquiz}
This is a quiz.
\begin{questions}
  \item Try to guess the correct answer.
  \begin{answers}{3}
    \bChoices[nCols=2,random]
    \Ans0 1 a choice\eAns
    \Ans1\label{eq} 2 another choice\eAns
    \Ans0 3 still another choice\eAns
    \Ans0 4 another\eAns
    \Ans0 5 incoming\eAns
    \Ans0 6 more choices\eAns
    \Ans0 7 another still\eAns
    \Ans0 8 too many\eAns
    \Ans0 9 choices\eAns
  \eFreeze
    \Ans0 10 None of these\eAns
  \eChoices
  \end{answers}
\end{questions}
\end{shortquiz}
```

Note the presence of the command `\eFreeze`. Any of the items listed after `\eFreeze` are not randomized, and are placed at the end of the list. So, for the example above, the first nine items will be randomized, whereas, the last item (None of these) will be placed at the end of the list.

Additional, there are five other commands that support the randomization feature.

```
\saveRandomSeed
\inputRandomSeed
```

A pseudo-random sequence of numbers requires an initial “seed value.” The the example file `random.tex` creates, by default, a seed value based on the data and time (the number of minutes since midnight); consequently, after every minute, the random sequence will change. By setting the value of the count register `\randomi`, the document author can also set the initial seed of the pseudo-random sequence.

The command `\saveRandomSeed` will write the last seed used in the source file to an auxiliary file (`\jobname_ran.sav`), while the command `\inputRandomSeed` inputs the seed stored in the `\jobname_ran.sav` back into the beginning of the source file. These two commands should be placed in the preamble.

By invoking both of these commands, a new pseudo-random sequence will be generated each time the source file is latexed.

Assuming a `\jobname_ran.sav` has already been created, by invoking the command `\inputRandomSeed` only (and not `\saveRandomSeed`), the seed already saved will be used for every subsequent compiling of the source document. Using the same seed is necessary in two situations:

1. When the quiz contains one or more `\label` commands, using the same seed gives you the same sequence every time you latex the document. This will give the auxiliary files a chance to come up to date so that any referencing of the label will be accurate.
2. When creating an online (or paper) quiz with randomization, which later you publish the solutions to, it is important that the randomization for the quiz document is the same as that for the solution document. By using `\inputRandomSeed` (rather than `\saveRandomSeed`), you get the same sequence for the solution document (unless you modify the source file, adding or removing questions that have randomization).

**Things to look for:** If `exerquiz` is not rearranging the order of the choices as you expect it to, it could be that `exerquiz` is reading an old `.sav` file. Either delete that file in your source folder, or comment out `\inputRandomSeed` in your document.

```
\useRandomSeed{<number>}
```

You may have several sections of the same class take a quiz with the questions rearranged for each. Save the seed value used by `exerquiz` to randomized the choices (open the `.sav` and copy and paste line you see into your document, for example, it could read `\randomi=132088850`. Then use `\useRandomSeed` to use that seed value for that class, for example

```
\useRandomSeed{132088850} % 11:00 class
% \useRandomSeed{634952429} % 12:30 class
```

Of course comment out `\inputRandomSeed`.

```
\turnOnRandomize
\obeyLocalRandomize
```

The command `\turnOnRandomize` overrides all local settings of `\bChoices` and causes all choice lists to be randomized. While `\obeyLocalRandomize` returns control to the local settings. For example,

```
\turnOnRandomize
...
\bChoices
  \Ans...\eAns
  \Ans...\eAns
...
\eChoices
```

will cause the choice list to be randomized, even though the `random` option was not specified. Whereas, in this code

```
\turnOnRandomize
...
\obeyLocalRandomize
...
\bChoices
  \Ans...\eAns
  \Ans...\eAns
  ...
\eChoices
```

the choices will not be randomized, because the `random` option was not specified; or they will be randomized if the `random` option is used.

**Limitations:** There are natural limitations on the use of `\bChoices` and `\eChoices` and consequently, there are limitations on the randomization. The content between `\Ans` and `\eAns` cannot have any verbatim text. This is usually not a problem for mathematical content, but could be a limitation for computer science where questions about syntax may be posed. I have in mind a work-around, but haven't pursued the problem as of yet.

## 24. Creating a Quiz Summary Table

This feature is designed for a quiz (as opposed to a `shortquiz`); and is probably most useful for a long quiz, spanning several pages. The `\displaySumryTbl` command creates a table that summarizes the user's effort for the quiz; `\displaySumryTbl` must follow the final `\end{questions}` and *must be on a separate page* from the last question of the quiz. An example of the generated table is shown below in [Figure 5](#).

 The demo file for this section is [sumry\\_tbl.pdf](#), found at the [AcroTeX Blog](#).

**Important:** The code for these commands are only input when the `exerquiz` option `usesumrytbls` is taken.

```
\begin{quiz*}{<quiz_name>}
Solve each of the following.
\begin{questions}
\item...
...
\item...
\end{questions}
```

```
\newpage %<--start new page
Before you complete your quiz by pressing 'End Quiz', review the list
below. A check mark indicates that you responded to that question;
otherwise, you did not respond to the question.
```

```
\begin{center}
\displaySumryTbl{\currQuiz}
\end{center}
```

If you are satisfied with your answers, press ‘End Quiz’ to complete the quiz.\medskip

```
\end{quiz*}\quad\ScoreField\currQuiz\eqButton\currQuiz
```

Question	Responded	Page	Question	Responded	Page
1	<input checked="" type="checkbox"/>	1	5	<input checked="" type="checkbox"/>	1
2(a)	<input type="checkbox"/>	1	6(a)	<input type="checkbox"/>	2
2(b)(i)	<input checked="" type="checkbox"/>	1	6(b)(i)	<input checked="" type="checkbox"/>	2
3	<input type="checkbox"/>	1	7	<input type="checkbox"/>	2
4(a)	<input checked="" type="checkbox"/>	1	8	<input type="checkbox"/>	2
4(b)(i)	<input type="checkbox"/>	1			

Figure 5: Quiz Summary Table

The table (as seen in [Figure 5](#)) shows a check mark for each question to which the student gave a response. The student can review the table, and optionally return to questions that had no response.

```
\displaySumryTbl[<key_vals>]{<quiz_name>}
```

**Parameter Description:** The second parameter is the quiz name of the quiz this table represents, in most cases, `\currQuiz` is used. The first parameter consists of key-value pairs, these are described below.

**Key-Value Pairs:** The first parameter takes key-value pairs

1. `ntables` is a key that has a value of 1 or 2, the default is determined by command `\smrytbl@ntables`, which is set to 2 in this package. If `ntables=1` only one table is created, and if `ntables=2` two tables are created, each containing half the information, as seen in [Figure 5](#).
2. `showmarkup`: If this switch is true, then markup fields are created that will hold the number of points the student received for each problem. The default is that there is no markup. Entering `showmarkup` in the option list will generate the markup fields. The code

```
\displaySumryTbl[ntables=1,showmarkup]{\currQuiz}
```

produces a single table with the points markup.

3. `nocorrections`: If this switch is true, corrections will not be shown when the user presses the ‘End Quiz’ button. Because of the way this key is implemented, if `nocorrections` is true, then it will be true for any “duplicate table,” that is, any table that references the same quiz.

When the user presses the ‘Corrections’ button, the borders of the are colored, and if `showmarkup` is used, the points earned for each problem are shown as well.

When corrections are made, a red border indicates the answer is wrong, a green border means the answer is correct, and a blue border the question was answered incorrectly, but partial credit was given. The colors are determined by the commands `\correctColor`, `\wrongColor`, and `\partialColor`. These are “global color” that need to set in the preamble; the quiz and shortquiz system uses the first two, change them will yield changes everywhere. However, you can change the `\partialColor` as this is the only place it is used.

```
\newcommand\partialColor{color.blue}
```

This is a JavaScript color, if you redefined it to be a nonstandard color, for example, this redefinition makes the partial credit color yellow.

```
\renewcommand\partialColor[["RGB", 1, 1, 0]]
```

**Important:** The table must be on a different page from the questions because there is an open page action that activates and populates the table based on the user’s responses.

Other related commands are

```
\renewcommand{\sumryTb1Q}{Question}
\renewcommand{\sumryTb1R}{Responded}
\renewcommand{\sumryTb1P}{Page}
```

The above three commands form the column titles of the table.

```
\renewcommand{\sumrytablesep}{\space}
```

Sets the separation between the tables, in the case that two tables are generated.

```
\newcommand{\sumryTb1ProbFmt}[1]{\textbf{\textcolor{blue}{#1}}}
```

The above command sets the  $\LaTeX$  formatting of the problems, the default is bold and blue, but may be redefined.

### 24.1. Placing the Quiz Summary Table Elsewhere

You may want the Quiz Summary Table in a place other than the default location (after the final `\end{questions}` and before the ‘End Quiz’). There is no problem putting it after the quiz and *before the next quiz*; however, if you want to put the table up front (that is, in front of the quiz) for whatever reasons, you need additional commands.

In the area just after `\end{questions}` and before ‘End Quiz,’ place the command

```
\writeProListAux
```

This command saves to the auxiliary file the two pieces of information the `\displaySumryTbl` needs to build the table. Then, in the location of your choice—usually outside of any quiz environment—enclose `\displaySumryTbl` in the `sumryTblAux` environment.


```
\begin{sumryTblAux}{<quiz_name>
\displaySumryTbl[<key-vals>]{<quiz_name>}
\end{sumryTblAux}
```

Here, `<quiz_name>` is the name of the quiz that this table represents. An example of this environment follows.

```
\begin{sumryTblAux}{\currQuiz}\small %<--make a little smaller
\renewcommand{\sumryTblQ}{\sffamily\bfseries Question}
\renewcommand{\sumryTblR}{\sffamily\bfseries Answered}
\renewcommand{\sumryTblP}{\sffamily\bfseries Page}
\begin{center}
  {\normalsize\sffamily\bfseries Quiz Summary Table}\\[3pt]
  \displaySumryTbl[showmarkup]{\currQuiz}
\end{center}
```

## 25. Bookmarking Exercises and Quizzes

It is unlikely that anyone would use this feature: Bookmarking exercises and quizzes.

 The demo file for this feature is `bkmrk_eq.tex`.

### 25.1. For Exercises

This “intelligent” bookmarking system requires the minimum of effort to use. For exercises, the command for bookmarking is `\expdfbookmark`,

```
\expdfbookmark{<bookmark_title>}
```

and should be placed immediately following the beginning of an exercise environment

```
\begin{exercise}\expdfbookmark{<bookmark_title>}
\begin{exercise*}\expdfbookmark{<bookmark_title>}
```

or, in the case of exercises with parts, and you want to bookmark an individual part to a question, place `\expdfbookmark` immediately after the `\item`, like so

```
\item\expdfbookmark{<bookmark_title>}
```

Every bookmark title, `<bookmark_title>`, has a suffix defined with `\exbookmarkfmt` or `\partbookmarkfmt`, depending on whether the entry is for the top-level exercise, or one of its parts. The default definitions are

```
\newcommand{\exbookmarkfmt}{\exlabel\space\theeqxno.\space}
\newcommand{\partbookmarkfmt}{(\thepartno)\space}
```

The command `\xlabel` expands to the language dependent word for ‘Exercise’, while `\theexno` expands to the exercise number, and `\thepartno` to the part letter. These commands can be redefined.

## 25.2. For Quizzes

To bookmark a `shortquiz` or `quiz`, use `\quizpdfbookmark`.

```
\quizpdfbookmark{<bookmark_title>}
```

It should be placed immediately following the beginning of the environments,

```
\begin{shortquiz}\quizpdfbookmark{<bookmark_title>}
\begin{quiz}{<quiz_name>}\quizpdfbookmark{<bookmark_title>}
```

► Unlike bookmarking for exercises, the bookmark command for quizzes does not support bookmarking individual questions within the quiz.

Every bookmark title, `<bookmark_title>`, for a `shortquiz` environment has a suffix defined with the `\sqbookmarkfmt` command, the default definition is

```
\newcommand{\sqbookmarkfmt}{Short Quiz \@shortquizCnt.\space}
```

The command `\@shortquizCnt` is a macro (not a counter) that contains the current short quiz count. This command can be redefined.

Every bookmark title, `<bookmark_title>`, for a `quiz` environment has a suffix defined with the `\qzbookmarkfmt` command, the default definition is

```
\newcommand{\qzbookmarkfmt}{Quiz \@quizCnt.\space}
```

The command `\@quizCnt` is a macro (not a counter) that contains the current quiz count. This command can be redefined.

► `\quizpdfbookmark` interacts with the command `\titleQuiz`. (for more information on this, see [Section 21.8](#) and [Section 20.4](#), pages 97 and 80, to learn about `\titleQuiz` for quizzes and short quizzes, respectively.) When a quiz has a defined title, as defined by the `\titleQuiz`, and the argument of `\quizpdfbookmark` is left empty, the value passed to `\titleQuiz` is used as the bookmark text. Thus, the code

```
\titleQuiz{Differentiation Quiz}
\begin{quiz}{MyDiffQuiz}\quizpdfbookmark{}
...
\end{quiz}
```

will bookmark the quiz under the name “Differentiation Quiz.”

## 25.3. Final Note

If the bookmarks for exercises, or parts of the exercises, or quizzes do not appear, you may have to increase the `tocdepth`. `Exerquiz` sets this at `\setcounter{tocdepth}{5}`.

## 26. Extending AcroTeX with `dljslib` and `insdljs`

The `exerquiz` Package, especially the math fill-in question, is quite programmable. In this section, we discuss two methods of extending the capabilities of the AcroTeX Bundle: (1) through the use of the package `dljslib`, which is a JavaScript library of extensions; (2) by writing your own custom extensions using the `insdljs` package for inserting JavaScripts into the PDF document.

### 26.1. Using the `dljslib` Package

The `dljslib` Package is actually a “library” of JavaScript functions. At the time of this writing, the library has JavaScripts that can process answers to math fill-in questions where an *equation* or a *vector* answer is expected. There is also a JavaScript compare function that properly evaluates an answer when an indefinite integral is expected. See the documentation that accompanies the package (by `latexing dljslib.dtx`) for details of how to use the library.

- ▶ For full documentation of `dljslib` package, see ‘[Powers and Exponents](#)’ on page 140.

### 26.2. Using the `insdljs` Package

With the `insdljs` package you can write your own JavaScript functions right in the  $\LaTeX$  source file. These custom JavaScripts are then inserted into the section of the PDF document where the document level JavaScripts reside. This package is a stand-alone package, and does not need `exerquiz`, though `exerquiz` now uses this package to insert its JavaScripts into the document.

- ▶ Refer to [eformman.pdf](#) (relative link [eformman.pdf](#)) for the `insdljs` and the `eforms` documentation. Also, see the sample file `insdljs_ex.tex` for examples that do not use `exerquiz`, and the file `jqzspec.tex`, for examples that does use `exerquiz`.

## 27. Submitting a quiz to a Web Server

Quizzes created by the `quiz` environment are entirely self-contained. They function within the Web browser (or from within the Adobe Reader) and do not communicate with any server. This kind of quiz is ideal for a do-it-yourself tutorial system, read by a well-motivated student who has the discipline to read the material and to take the quizzes in the spirit in which they are given.

However, some educators, myself included, may wish to use the quizzes created by the `quiz` environment for classroom credit. It is necessary, therefore, for the student to be able to submit quiz results to a Web server which, in turn, should store the results to a database.

In this section we discuss techniques of turning the quiz into something that can be submitted to a server.

- ▶ I have released the `eq2db` Package, a  $\LaTeX$  macro package and server-side script to process `exerquiz` quizzes. See ‘[Powers and Exponents](#)’ on page 138.

## 27.1. Technical Info for “Do It Yourself”

All one really has to do is to redefine the “End Quiz” link or button to submit the results of the quiz to the Web server and CGI of your choice. Since the quiz itself is scored, (optionally) marked, with (optional) answers and solutions provided, the CGI simply stores the quiz results to a database.

### • Redefining “End Quiz”

I’ve written the “End Quiz” link (button) to have various programming hooks available to the developer. The following code is common to both `\eq@EndQuizLink` and `\eq@EndQuizButton`, the macros that control the action of the end quiz link and button, respectively.

```
if (\minQuizResp(\thequestionno)) {\r\t
  var f = this.getField("ScoreField.\curr@quiz");\r\t\t
  if ( f != null )\r\t\t\t
    this.getField("ScoreField.\curr@quiz").value
      =(\eq@QuizTotalMsg);\r\t\t
  \eq@submitURL
  resetQuiz("\curr@quiz")\r\t
}
```

► The code is a mixture of  $\LaTeX$  macros and JavaScript. You can see from this code, that there is a submit hook macro provided, `\eq@submitURL`. Normally, this macro has a definition of `\@empty`. A developer needs only redefine this macro accordingly; one would use the Acrobat JavaScript method `this.submitForm()` to do this. See the *Acrobat JavaScript Scripting Reference* [1] for more detail about this method.

► The code flow above is as follows: (1) Execute this code if the threshold has been met. (See [Setting the Threshold](#).) The text macro `\curr@quiz` holds the base name of the current quiz.

(2) If the field `"ScoreField.\curr@quiz"` exists, then write the student’s score to that field. (This is the “Score: 2 out of 3” that you see in the demo quizzes.)

(3) We then submit with the macro `\eq@submitURL`. (This would do nothing if its value is `\empty`, the default value.) At this point we call a `resetQuiz("\curr@quiz")` which sets some values in an array to indicate the state of this quiz.

### • Gathering ID Information with `\textField`

► What kind of information would one submit to a CGI? Well, there is the usual information concerning the identity of the student (Name, SSN, etc.) and the course, section and so on.

This basic information can be gathered from the student by inserting text fields into the document to be filled in. Exerquiz provides the macro `\textField`<sup>17</sup> for this purpose. For example,

<sup>17</sup>You can also use `hyperref`’s `\TextField` command for this purpose as well.

```
\newcommand\FirstName[2]{\textField
  [\DV{First Name}\textFont{TiRo}\textSize{10}\textColor{0 0 1}]
  {IdInfo.Name.First}{#1}{#2}}
```

This defines a text field with a name of "IdInfo.Name.First", the two arguments are the width and height of the field that you want to create. E.g.,

```
\FirstName{100pt}{10pt}
```

creates a text field 100pt wide and 10pt high.

The `\textField` macro takes four parameters.

```
\textField[#1]#2#3#4
```

The first (optional) parameter can be used to custom design the field; the second is the name of the field; the third and fourth are the width and height of the field desired.

► See the file `eqformman.tex` on [AcroT<sub>E</sub>X](#) eForm support for complete documentation on `\textField`.

#### • Gathering Quiz Specific Information with `\eqSubmit`

In addition to ID information on the one taking the quiz, specific information about what quiz is being taken and where the results of the quiz are to be stored are needed as well.

Exerquiz provides a basic macro, called `\eqSubmit` that can be used to gather basic formation of this type. The definition of it and related commands are given below:

```
\newcommand\databaseName[1]{\def\db@Name{#1}}\def\db@Name{}
\newcommand\tableName[1]{\def\db@Table{#1}}\def\db@Table{}
\newcommand\eqCGI[1]{\def\eq@CGI{#1}}\def\eq@CGI{}
\newcommand\eqSubmit[3]
  {\eqCGI{"#1"}\databaseName{#2}\tableName{#3}}
```

The meaning of the parameters are self-explanatory.

Just prior to the quiz you can type:

```
\eqSubmit{http://www.myschool.edu/cgi-bin/myCGI.cgi}%
  {CalcIII}{Quizzes}
\begin{quiz*}{Quiz3} Answer each of the following.
\begin{questions}
...
...
\end{questions}
\end{quiz*}\quad\ScoreField\currQuiz\eqButton\currQuiz

\noindent
Answers: \AnswerField\currQuiz
```

► Any redefinition of `\eq@submitURL` would then include the values of some or all of these text parameters:

`\eq@CGI, \db@Name, \db@Table, \curr@quiz`

The last text macro is not gathered by `\eqSubmit`, it is known, however, at the time `\eq@submitURL` is expanded.

#### • Some Variables to Submit

When you submit a quiz to a server, the values of *all* fields are also submitted, unless you define specifically which fields are to be submitted.

In addition to the ID info, you would like also to submit the results of the quiz itself. The relevant variables are as follows:

1. The JavaScript variable `Score` has the number of correct responses as its value.
2. The  $\LaTeX$  counter variable `\thequestionno` has the count of the total number of questions in the quiz.
3. The JavaScript array `Responses` contains the responses of the student: multiple choice and fill-in responses. The contents of this array can be converted to a comma-delimited string by using the `toString()` method, `Responses.toString()`.

Now, how does one submit these values? The `\eq@submitURL` command can be used not only to submit the data, but to also populate certain *hidden* fields with this information. The hidden data is submitted along with the ID info to be processed. You can use the `\textField` to create hidden text fields for this purpose. See the next section for a discussion of how to create hidden text fields.

## 27.2. Features *apropos* to Submitting

#### • Assigning Points

See ‘[Adding Points to a Quiz](#)’ on page 100 for information on this subject.

#### • \NoPeeking

If you execute the command `\NoPeeking` in the preamble of your document, or prior to a quiz, then any quiz question with solution will be protected somewhat from prying eyes.

In this case, an open page action is placed on the first page of each solution. If the user (student) tries to view a quiz solution before doing the quiz, the [Adobe Reader](#) will automatically change the page to the page containing the quiz and place an alert box on the screen saying that viewing the solution before taking the quiz is not permitted.

To resort to the default behavior, use the `\AllowPeeking` command.

The previous quiz has been surrounded with a `\NoPeeking/\AllowPeeking` pair. If you go to one of the [solutions](#) to that quiz, you will see what happens. If nothing interesting happens, read the next red point.

- ▶ Protection is removed when you click on “End Quiz” and restored when you click on some “Begin Quiz”.

## 28. Functions and Syntax supported by Exerquiz

In addition to the JavaScript built in math functions, exerquiz defines a number of other functions useful in mathematical work. These functions are typically input in a math question created by `\RespBoxMath`, for example, or in the function input field of an AcroTeX document.<sup>18</sup>

Exerquiz also has its own syntax for entering algebraic expression in the `\RespBox-Math` field, these are document here as well.

### 28.1. Functions supported by Exerquiz

The following functions are built-in to JavaScript:

Function	Description	Usage
<code>abs</code> , <code> · </code>	absolute value function, preferred form, <code> · </code>	<code> -3 =3</code> , <code> x </code>
<code>sin</code>	the sine function	<code>sin(pi/2)=1</code> , see the definition of <code>pi</code> below
<code>cos</code>	the cosine function	<code>cos(1.5)</code> , <code>cos(2*x)</code>
<code>tan</code>	the tangent function	<code>tan(pi/4)=1</code>
<code>asin</code>	inverse sine function	<code>asin(x)</code>
<code>acos</code>	inverse cosine function	<code>acos(.2)</code>
<code>atan</code>	inverse tangent function	<code>atan(-4)</code>
<code>exp</code>	the natural exponential function, this function is also implemented as a <code>e^x</code> , where <code>e</code> is the natural number	<code>exp(-.5)</code> , <code>exp(-x^2)</code>
<code>log</code>	the common logarithm. In JavaScript <code>log</code> is the natural logarithm, <code>exerquiz</code> redefines this to be the common logarithm, use <code>ln</code> for natural logarithm	<code>log(10)=1</code>
<code>sqrt</code>	the square root function	<code>sqrt(4.5)</code>
<code>ceil</code>	the <code>ceil</code> function, <code>ceil(x)</code> is the smallest integer greater than or equal to <code>x</code>	<code>ceil(4.5)=5</code>
<code>floor</code>	the <code>floor</code> function, where <code>floor(x)</code> is the largest integer less than or equal to <code>x</code>	<code>floor(4.5)=4</code>
<code>max</code>	the maximum between two numbers	<code>max(2, 3.4) = 3.4</code>
<code>min</code>	the minimum between two numbers	<code>max(2, 3.4) = 2</code>
<code>random</code>	a pseudo-random number generator, this function is used internally by <code>exerquiz</code> , and is available to users, <code>random()</code> returns a number between 0.0 and 1.0	<code>random()</code>
<code>round</code>	a rounding function, rounds a number to the nearest integer	<code>round(4.5)= 5</code> , <code>round(-3.3)=-3</code>

<sup>18</sup>The AcroTeX Graphing Bundle, <http://www.math.uakron.edu/~dpstony/acroTeX.html>

The next set of functions are defined by `exerquiz`.

Function	Description	Usage
<code>cot</code>	the cotangent function	<code>cot(1)</code>
<code>sec</code>	the secant function	<code>sec(2)</code>
<code>csc</code>	the cosecant function	<code>csc(2)</code>
<code>arcsin</code>	the inverse sine function, same as <code>asin</code>	<code>arcsin(.5)</code>
<code>arccos</code>	the inverse cosine function, same as <code>acos</code>	<code>arccos(-.5)</code>
<code>arctan</code>	the inverse tangent function, same as <code>atan</code>	<code>arctan(1)</code>
<code>ln</code>	the natural logarithm	<code>ln(3.4)</code>
<code>sinh</code>	the hyperbolic sine function	<code>sinh(0)</code> , <code>sinh(x)</code>
<code>cosh</code>	the hyperbolic cosine function	<code>cosh(0)</code> , <code>cosh(x)</code>
<code>tanh</code>	the hyperbolic tangent function	<code>tanh(x)</code>
<code>coth</code>	the hyperbolic cotangent function	<code>coth(x)</code>
<code>sech</code>	the hyperbolic secant function	<code>sech(x)</code>
<code>csch</code>	the hyperbolic cosecant function	<code>csch(x)</code>
<code>asinh</code>	the inverse hyperbolic sine function	<code>asinh(0)</code> , <code>asinh(x)</code>
<code>acosh</code>	the inverse hyperbolic cosine function	<code>acosh(0)</code> , <code>acosh(x)</code>
<code>atanh</code>	the inverse hyperbolic tangent function	<code>atanh(x)</code>
<code>acoth</code>	the inverse hyperbolic cotangent function	<code>acoth(x)</code>
<code>asech</code>	the inverse hyperbolic secant function	<code>asech(x)</code>
<code>acsch</code>	the inverse hyperbolic cosecant function	<code>acsch(x)</code>
<code>sgn</code>	the sign function, <code>sgn(x)</code> is -1 if <code>x</code> is negative, 1 if <code>x</code> is positive, and 0 if <code>x</code> is zero	<code>sgn(-3.2) = -1</code>
<code>cis</code>	the polar representation of a complex number, $\text{cis}(\theta) = \cos(\theta) + i \sin(\theta)$ . This function is only defined if the <code>complex</code> option of <code>dljslib</code> is used.	<code>cis(pi/3)</code>

There are several other `exerquiz`-defined functions that are available through options of the `dljslib`, these are `C` (combinations), `P` (permutations), and `fact` (permutations). See the section titled '[Extending Numerical Functions](#)' on page 153 for details.

**Recognized Constants.** `Exerquiz` recognizes `PI` and `pi` as symbols for the mathematical constant  $\pi \approx 3.14159$ . The letter `e` is recognized as the natural number, which is approximately 2.7182818285.

## 28.2. Syntax supported by Exerquiz

`Exerquiz` attempts to make entering mathematics as easy and as natural as using a modern calculator. The following paragraphs outline the syntax of `exerquiz`. Authors

should reproduce some form of these rules to their students, and perhaps some practice documents for them to test themselves with.

**Grouping.** Algebraic expressions are grouped with matching parentheses ( ), brackets [ ], or braces { }.

**Spaces.** When the user enters a math expression, all spaces are removed before parsing of the expression begins. Students can write  $1+(2+3)$  or  $1 + ( 2 + 3 )$ , these are the same.

**Addition and Subtraction.** These operations are performed using + and -, as expected. For example,  $(x+1)-(x-2)$ ,  $x+\sin(x-\pi)$

**Multiplication and Division.** These operations are performed using \* and /, as expected. For example,  $2*x*(3*x+1)$  and  $(2*x-1)/(2*x+1)$ .

The use of the multiplication (\*), is required, by default, and its use can be very tedious to the student. If you include the `dljslib` package with the `ImplMulti` option, `exerquiz` automatically inserts the multiplication symbol (\*). The examples above, can be written as  $2x(3x+1)$  and  $(2x-1)/(2x+1)$ ; it is recommended, therefore, that the `ImplMulti` option should normally be used. (Note: The use of \* is still correct even when `ImplMulti` option is in effect.)

**Powers and Exponents.** Exerquiz uses ^ to indicate exponents, for example,  $x^2$  means  $x^2$ . Other examples are  $17^{(x+1)}$ ,  $(x^2+1)^3$ ,  $(\sin(x))^3$ , and  $(x+1)^{(2x+1)}$ .

Recall that the letter e is recognized as the natural number, so  $e^{-x^2}$  is the same as  $\exp(-x^2)$ . Normally, the use of e is preferred by students over the use of `exp`. For work in elementary mathematics, the existence of `exp` need not be even mentioned to the students.

The difficulty of entering powers of functions is another issue for students. As illustrated above, in entering  $(\sin(x))^3$  or more complex expression, students find a source of difficulty, pain, and anguish. Exerquiz allows entering powers in the calculator style, the student can enter  $\sin(x)^3$  for  $(\sin(x))^3$  (often written as  $\sin^3(x)$ ) or  $3^{\sin(x)}$  to mean  $3^{\sin(x)}$ . Expressions like  $2^2^3$  are interpreted as  $(2^2)^3$  (the typeset version is  $(2^2)^3$ ), which evaluates to 64. The use of parentheses is needed if the desired expression is  $2^{2^3}$ ; this expression needs to be typed as  $2^{(2^3)}$ , which evaluates to 256.

If the `ImplMulti` option of the `dljslib` package is in effect, powers of functions can be entered in a more mathematically traditional, for example  $\sin^2(x)$  is recognized as the same as  $\sin(x)^2$ , or  $(\sin(x))^2$ ; consequently,  $\sin^2(x)^2$  is the same as  $\sin(x)^4$ !

## The eq2db Package

As the name suggests, this package facilitates submitting an Exerquiz quiz to a CGI for storage in a database. The  $\LaTeX$  package itself does very little other than to define some useful commands that make it easy to convert a self-contained quiz into one that is submitted to server-side script.

The eq2db currently has two options, eqRecord and custom:

```
\usepackage[eqRecord]{eq2db}
```

The option eqRecord sets up the quiz to use an ASP (Active Server Page) that I have written. This ASP, named naturally, eqRecord.asp, takes the data and stores it to a database, such as Microsoft Access.

There is also a custom option. With this option, a developer can write  $\LaTeX$  code to set the quiz up for submission to a CGI used or written by the developer.

► For more details, see [eq2dbman.pdf](#),<sup>19</sup> the documentation for the distribution. The eq2db package is available for download at the [AcroTeX](#) home page:

<http://www.math.uakron.edu/~dpstory/webeq.html>

---

<sup>19</sup>The absolute URL is <http://www.math.uakron.edu/~dpstory/acrotex/eq2dbman.pdf>

## AcroTeX eForm Support

In this document, we describe the support for form elements in an AcroTeX document. The PDF Specifications indicate there are four different categories of fields for a total of seven types of fields.

1. Button Fields
  - (a) Push Button
  - (b) Check Box
  - (c) Radio Button
2. Choice Fields
  - (a) List Box
  - (b) Combo Box
3. Text Fields
4. Signature Fields

The AcroTeX Bundle does not support *signature fields*, this leaves six types of fields. Commands for creating each of the remaining six types will be discussed.

The hyperref Package (Rahtz, Oberdiek *et al*) provides support for the same set of form fields; however, not all features of these fields can be accessed through the hyperref commands. I was determined to write my own set of commands which would be sufficiently comprehensive and extendable to suit all the needs of the AcroTeX Bundle. All the quiz environments have been modified to use this new set of form commands, in this way, there is a uniform treatment of all form fields in AcroTeX Bundle.

► The documentation for eForm support is too voluminous to include in this already voluminous document. See [eformman.pdf](#) (relative link [eformman.pdf](#)) for complete details.

## The AeB JavaScript Library

The `dljslib` package—document level JavaScript library—is a companion to `exerquiz` and utilizes `insdljs` package. It is a library of JavaScript code that can be referenced in the preamble of a  $\LaTeX$  document, the code is then extracted from the package files, and inserted into the  $\LaTeX$  source using the `insDLJS` environment.

The code belonging to the library supports the processing of the quizzes and short quizzes produced by `exerquiz`; they extend the capabilities of the core features of the `exerquiz` quizzes. Many of the JavaScript functions in the library were contributed by users of AeB.

My deep appreciation goes out to the contributors to the AeB JavaScript library.

- The options `unordered`, `factors`, `point` and `intervals` were written by Dr. Bruce Wagner for an online grading system developed by Drs. Bruce Wagner and David Arnold, and Mr. Jacob Miles-Prystowsky. Dr. Robert Marik contributed to the `intervals` option, as well.
- The options `nodec`, `noBinFact` and `combinatorics` were contributed by Ross Moore and Frances Griffin, and were developed for their `MacQTeX` online testing system.<sup>20</sup>

The AeB JavaScript library continues to accept contributions from AeB developers.

## 29. Usage and Options

As with all  $\LaTeX$  package, you use `dljslib` as follows,

```
\usepackage[<options>]{dljslib}
```

With each of the `<options>`, the associated JavaScript code is brought into the  $\LaTeX$  source document. The JavaScript functions, then, are available to your `exerquiz` quizzes.

A list of the `<options>` of the `dljslib` package are listed below, and each are described in detail in the sections that follow.

1. `ImplMulti`: Use this option to allow implied multiplication, e.g., the user can input  $2x\sin(x)$  instead of  $2*x*\sin(x)$ . See '`ImplMulti`' on page 142.
2. `equations`: Use this option to process questions that take an equation as an answer. See '`equations`' on page 143.
3. `vectors`: Use this option to process questions that take a vector as an answer. See '`vectors`' on page 143.
4. `setSupport`: Use this option to process questions that take a set as an answer. See '`setSupport`' on page 144.
5. `unordered`: Use this option to process questions that take an unordered list as an answer. See '`unordered`' on page 145.


<sup>20</sup><http://rutherglen.ics.mq.edu.au/~macqtex/>

6. `factors`: Use this option to process questions that take a factored form of a polynomial. See '`factors`' on page 145.
7. `point`: Use this option to process questions that take an ordered pair  $(x, y)$  as an answer. See '`point`' on page 145.
8. `intervals`: Use this option to process questions that take an interval, or a union of intervals as an answer. See '`intervals`' on page 146.
9. `indefIntegral`: Use this compare function when evaluating the response to an indefinite integral. See '`indefIntegral`' on page 148.
10. `nodec`: Use this filter function to deny the use of decimal numbers in an answer. See '`nodec`' on page 150.
11. `noBinFac`: User this filter function to deny the use of combinatorial functions. See '`noBinFac`' on page 150.
12. `limitArith`: Use any of this collection of filter functions to limit the use of arithmetic operations in an answer. See '`limitArith`' on page 151.
13. `combinatorics`: Use this option to add combinatorial functions. The user may use these in an answer, unless denied by the `noBinFac` filter. See '`combinatorics`' on page 153.

For a document author that is building a system of quizzes and who regularly uses a subset of these options, sets of options can be grouped together into a custom option. Use the `libcusopt.opt` file to declare your own sets of options. These declarations should be a combination of existing options. For example, The declarations in the `libcusopt.opt` file

```
\DeclareOption{procrspgrp}{\includeOptions{%
    vectors,setSupport,equations,intervals,limitArith,%
    unordered,point,factors,ImplMulti}
}
```

creates a new option for `dljslib` called `procrspgrp` which bundles together all frequently used options. The file `libcusopt.opt` can be placed in the  $\LaTeX$  search path, but it's perhaps best to put it in your source file, so that the options are local to that folder. The command `\includeOptions` is defined in the `insdljs` package.

 A sample `libcusopt.opt` is distributed with the `AeB`, this sample file consists of the example given above, and is used by the sample file `dljslib_apw.tex`, which can be found in the folder `dljslib_examples`.

The JavaScript functions are organized into five categories:

1. **Parse Input Extensions** are support functions whose capabilities are built into the core `exerquiz` package. When `exerquiz` detects the presence of these support functions, it utilizes them seamlessly.

2. **Response Functions** are functions that take over the task of processing the user input. Discussion of these functions begins on page 142.
3. **Compare Functions** are functions that perform the comparison between the author's answer and the student's answer. Based on this comparison, the student's answer is judged correct or not. Refer to page 148 for the compare functions.
4. **Filter User's Responses** are functions that are called immediately after the user commits the response. A filter function can examine the response before passing it on to the response function. The filter function can cancel the user input if the input does not meet certain criterion. The filter functions begin on page 150.
5. **Extending Numerical Functions** extend the collection of built-in functions that perform numerical calculations. These numerical functions are discussed beginning on page 153.

### 29.1. Parse Input Extensions

This type of JavaScript functions are support functions whose capabilities are built into the core `exerquiz` package. When `exerquiz` detects the presence of these support functions, it utilizes them seamlessly.

#### • `ImplMulti`

The default behavior for processing a math fill-in question is to require the student to insert '\*' for multiplication and to enclose any function being raised to a power in parentheses, e.g.  $x * (\sin(x))^2$ . This becomes quite tiresome if the expression to be entered is complicated. The `ImplMulti` option enables the student to use *implied multiplication* and *simplified exponentiation*.

The `ImplMulti` option loads two JavaScript functions, `Ck4Products` and `Ck4Exponents`. The latter implements the notation,  $\sin^2(x)$  for  $\sin^2(x)$ , and is equivalent to  $(\sin(x))^2$ . The more complex exponent must be enclosed in parentheses, for example  $\ln^{(x+1)}(x)$ . The former function, `Ck4Products`, inserts the multiplication symbol, '\*', wherever multiplication is implied. For example,  $x\sin(xy)$  becomes  $x*y*\sin(x*y)$ .

**Important:** If loaded, `exerquiz` will automatically use these two functions in any math fill-in question, no further action is needed.

 Many of the demo files that come with `AeB` use the `ImplMulti` option, one such representative file is `jquizstst.tex`.

### 29.2. Response Functions

For fill-in questions, a response function is one that is called immediately after the user enters and commits an answer. The default response function that processes the user's response is `ProcResp`. The 10<sup>th</sup> argument of the `\RespBoxMath` command,

```
\RespBoxMath[#1]#2(#3)[#4]#5#6#7#8[#9]*#10
```

is used to specify another response function. This section discusses the alternate response functions available in the library, and gives examples of each.

### • equations

Use this option to process questions for which an equation is the expected answer. This option defines the JavaScript function ProcRespEq. When using a math fill-in question (the `\RespBoxMath` command), this function is placed in the optional 10<sup>th</sup> argument, the one that follows `*`. See '[\RespBoxMath: The Math Question](#)' on page 104 for a description of the parameters of `\RespBoxMath`

The following question goes inside a `quiz` or `shortquiz` environment.

```
\RespBoxMath{y = 4 * x - 3}(xy){4}{.0001}{[0,1]x[0,1]}*{ProcRespEq}
```

✂ The demo file `jqzequat.tex` includes the above example as well as other in the context of a working `shortquiz`, found in the `d\jslib_examples` folder.

Alert box message: May be redefined.

```
\newcommand\equationsAlertMsg{"An equation is expected"}
```

### • vectors

Use this option to process questions for which a vector is the expected answer. This option defines the JavaScript function ProcVec. When using a math fill-in question (the `\RespBoxMath` command), this function is placed in the optional 10<sup>th</sup> argument, the one that follows `*`. See '[\RespBoxMath: The Math Question](#)' on page 104 for a description of the parameters of `\RespBoxMath`.

The following question goes inside a `quiz` or `shortquiz` environment. The premise of these two questions is that the definitions of  $\vec{a}$ ,  $\vec{b}$  and  $f(t)$  are given:

```
\item $\vec{a} + \vec{b} = \RespBoxMath{<4, 4>}{1}{.0001}24*{ProcVec}$
\item $\vec{f}'(t) = \RespBoxMath{<e^t, 2*t, cos(t)>}(t){3}{.0001}{0}{1}*{ProcVec}$
```

The first question takes a three component vector of numerical values, the second takes a three component vector of function values.

✂ These two examples were taken from the demo file `jqzvector.tex`, which includes both `quiz` and `shortquiz` environments, found in the `d\jslib_examples` folder.

Alert box messages: May be redefined.

```
\newcommand\vectorsErrorMsgi{"I'm looking for a vector.
  You need to use proper vector notation,
  try using angle brackets <...>."}
\newcommand\vectorsErrorMsgii{"Angle brackets are not balanced.
  Check the expression you typed in."}
\newcommand\vectorsErrorMsgiii{"Incorrect number of components.
  The answer requires " + aCorrAns.length + " components."}
```

- **setSupport**

We introduce three JavaScript (response) functions for handling a set of answers or a comma delimited list of answers, these are `ProcRespSetNum`, `ProcRespSetSym`, and `ProcRespListFormula`. The first function is for numerical answers, the second for simple symbolic answers. These JavaScript response functions are passed to `\RespBoxMath` as the 10<sup>th</sup> argument.

✂ The demo file for these three response functions is `set_test.tex`; the examples below were taken from that file, found in the `dljslib_examples` folder.

The functions `ProcRespSetNum`, `ProcRespSetSym` can handle (math fill-in) questions whose answers are a *set of numbers (symbols)* or a comma delimited *list of numbers (symbols)*.

**ProcRespSetNum:** The function can handle (math fill-in) questions whose answers are a *set of numbers* or a comma delimited *list of numbers*.

This example takes a comma-delimited list as its expected response. After the student enters the response, the answer is formatted as a set, with enclosing braces, like so `{ ... }`. The student does not enter the braces, the JavaScript does after the comma-delimited list is entered and committed.

```
$(x+1)^2 (x+3) = 0$, $S = \RespBoxMath[\AddAAFormat{\formatAsSet}
\rectW{.75in}\textSize{0}]{-1,-3}{1}{.0001}{[0,1]}*{ProcRespSetNum}$
```

The formatting of answer is accomplished by passing `\AddAAFormat{\formatAsSet}` as an optional argument of `\RespBoxMath`. The command `\formatAsSet` expands to a JavaScript function that is defined when the `setSupport` option is taken. The little-known key `\AddAAFormat` is defined in `exerquiz` and is specifically designed for inserting special formatting into a math fill-in response. The use of the set formatting is optional.

This next example again uses `ProcRespSetNum`. The equation has repeated roots. To require the student to enter the repeated roots, just list them as part of the answer argument.

```
$(x+1)(2x-1)(x-2)^3 = 0$, $x = \RespBoxMath[\rectW{.75in}\textSize{0}]{
-1, 1/2, 2, 2, 2}{1}{.0001}{[0,1]}*{ProcRespSetNum}$
```

**ProcRespSetSym:** The function can handle (math fill-in) questions whose answers are a *set of symbols*.

This function can handle simple symbolic answers. The variable list should be the “universal set” of the problem.

```
\def\U{a,b,c,d,e,f,g} % define a universal set
$A \cap B = \RespBoxMath[\AddAAFormat{\formatAsSet}
\rectW{.75in}\textSize{0}]{c,d}{\U}{1}{.0001}{[0,1]}*{ProcRespSetSym}$
```

**ProcRespListFormula:** This function can handle a (math fill-in) question whose answer is a comma-delimited list of expressions. Similar to ProcVec, but the angle brackets are not used to specify the vector.

In this example, taken from `set_test.tex`, a vector response is required, but without the enclosing angle brackets.

```
\vec f'(t) = \RespBoxMath{e^t, 2*t, cos(t)}(t){3}
{.0001}{0}{1}*{ProcRespListFormula}$
```

You can also define your own formatting function, to format the input.

- **unordered**

This option defines the JavaScript (response) function ProcRespSetFormula that will grade an *unordered list* of formulas, such as  $x$ ,  $x^2$ ,  $x^3$ . For example,

```
Some question requiring an unordered list of responses.
\RespBoxMath{x,x^2,x^3}{10}{.0001}{{1,2}}*{ProcRespSetFormula}
```

A correct answer is a list of the monomials  $x, x^2, x^3$  in any order. As usual, this JavaScript response function is passed to `\RespBoxMath` as the 10<sup>th</sup> argument.

✂ The demo file that uses the `unordered` option is `d\jslib_apw.tex`, which can be found in the `d\jslib_examples` folder

- **factors**

The `factors` option defines ProcRespFactors, passed to `\RespBoxMath` as the 10<sup>th</sup> argument. The response function ProcRespFactors is used for grading polynomial factoring questions. For example, if a polynomial factors as  $-5x^2(x-4)(x+2)$ , then the answer is only unique up to the order of factors and a change of sign on an even number of factors. The student could respond with  $-5x^2(x+2)(x-4)$  or  $5x^2(-x+4)(x+2)$  or  $-5(x-4)(x+2)x^2$ . The function should grade all of these variations correctly; however, it will not grade  $-(x-4)(x+2)5x^2$  as correct. The leading coefficient, if there is one, must be placed at the beginning, which conforms to our usual conventions.

```
Completely factor the polynomial $x^3-4x^2 + 2x - 8$.
\RespBoxMath{(x^2+2)(x-4)}{10}{0.0001}{{0,1}}*{ProcRespFactors}
```

Note that ProcRespFactors is passed to `\RespBoxMath` as the 10<sup>th</sup> argument.

✂ The demo file that uses the `factors` option is `d\jslib_apw.tex`, which can be found in the `d\jslib_examples` folder

- **point**

The response function ProcPoint is an almost exact copy of your ProcVec, but uses parentheses instead of angle brackets as delimiters. Used with questions that have a point  $(x, y)$  as the answer.

```
Calculate the derivative of $\vec f(t) = (e^t, e^{t^2})$.
\RespBoxMath{(e^t, 2te^t)}(t){10}{0.0001}{{0,1}}*{ProcPoint}
```

`ProcPoint` is passed to `\RespBoxMath` as the 10<sup>th</sup> argument.

 The demo file that uses the `point` option is `d\j\slib_apw.tex`.

Alert box messages: May be redefined.

```
\newcommand\pointErrorMsgi{"I'm looking for a point.
  You need to use proper point notation."}
\newcommand\pointErrorMsgii{"Parentheses are not balanced."}
\newcommand\pointErrorMsgiii{"Incorrect number of components.
  The answer requires " + aCorrAns.length+ " components."}
```

#### • intervals

`ProcRespIntervals` is for grading questions that have an interval or union of intervals as the answer. Example of answers supported by this function are listed below:

```
(2,3)
[2,3]
(-inf,4)
(-inf,4]
(6,inf)
[6,inf)
(2,3)U[8,9)
(-inf,6)U(9,inf)
(-inf,6]U[7,8)U(9,inf)
```

Here `inf` for infinity and the capital letter `U` for the union symbol. Note also that intervals in a union can appear in any order. The

The `ProcRespIntervals` function has been modified by Robert Marik to allow intervals with end points `\ln(2)`, `\exp(4)` or `\sqrt{3}`. Thus, intervals of the form

```
(\ln(2),\ln(3)]U(\exp(10),inf)
```


are recognized by this procedure.

```
\RespBoxMath{(-inf,4]U[5,inf)}{10}
  {.0001}{[0,1]x[0,1]x[0,1]x[0,1]}*{ProcRespIntervals}
```

**Important:** Problems with unbalanced parentheses while authoring and building the PDF file. The code below

```
\intervalbox{(-inf,4)U[5,inf)}
```

the right parenthesis of the second interval does not have a matching left parenthesis. This causes problems with Acrobat Distiller. As a work around, escape the odd parenthesis (using `\eqbs`), like so `\intervalbox{(-inf,4)U[5,inf\eqbs)}`, where `\eqbs` is defined in the `insdls` package.

 The demo file that uses the `factors` option is `d\j\slib_apw.tex`.

- **complex**

This option extends the ability of `exerquiz` to process questions that are expecting a complex number as an answer.

**ProcRespComplex:** The `complex` option defines the function `ProcRespComplex`, which is specified in the usual way, as seen below.

```
$(3+3i)+(4+5i) = \RespBoxMath{7+8i}{3}{0.0001}{{[0,1]}}*{ProcRespComplex}$
```

The `complex` option also defines the `cis` function, the definition of which is  $\text{cis}(x) = \cos(x) + i \sin(x)$ . When `complex` is not loaded, this function is undefined, and will be caught by the parser of `exerquiz` as a syntax error. If `complex` is loaded and the user enters the `cis` function in a non-complex problem, the user's answer is not flagged as a syntax error and will most probably be scored as *wrong*. The `exerquiz` implementation of the `cis` function, does not, at this time, support powers of the `cis` function, the parser attempts to find expressions of the form `cis^`, flag it, and emit an alert message, see below. This implementation does not detect powers of the form  $(\text{cis}(x))^2$ , however.


The next two functions are due to Bruce Wagner, with modifications by D. P. Story.

**ProcRespListComplex:** This response function evaluates an *ordered* list of comma-delimited complex numbers. For example,

```
If $z=4(\cos x+i\sin x)$, compute $z^2$ and $z^3$, in that order.
\RespBoxMath{16*cis(2x),64*cis(3x)}{4}{0.0001}{{[0,1]}}*{ProcRespListComplex}
```

**ProcRespSetComplex:** This response function evaluates an *unordered* list of comma-delimited complex numbers. For example,

```
Find all real and complex solutions of the equation $x^2=-9$.
Express your answer(s) in rectangular form $a+bi$.
\RespBoxMath{3i,-3i}{4}{0.0001}{{[0,1]}}*{ProcRespSetComplex}
```

 The demo file that uses the `complex` option is `dljslib_complex.tex`.

Alert box messages: May be redefined.

```
\newcommand{\complexPowerAlertMsg}{%
  "Powers of i (for example, i^2, i^3) are not supported,
  replace powers of i with their complex equivalents."
}
\newcommand{\complexCisAlertMsg}{%
  "The cis function does not support exponents. Write,
  for example, cis^3(x) as cis(3*x), instead."}
\newcommand{\alertNotComplexMsg}{%
  "The expression is not in the form of a complex
  number, a+bi"}
\def\emptyCompComplexMsg(#1){%
```

```

    "You entered nothing for the component "
      +(#1+1)+" of your answer. Please enter a complex number."
  }

```

The latter definition takes a delimited argument, necessary since it executes within a verbatim environment where braces { and } are not grouping delimiters.

### 29.3. Compare Functions

Compare functions are used to compare the author's answer with the student's answer. Based on the comparison, the student's answer is judged correct, or not. The `exerquiz` package has two built-in compare functions: `diffCompare` (the default) and `reldiffCompare`. The former makes the comparison based on absolute differences

$$\text{diffCompare} = |\text{Author Ans} - \text{Student Ans}|$$

while the latter uses absolute relative differences to make the comparison

$$\text{reldiffCompare} = \frac{|\text{Author Ans} - \text{Student Ans}|}{|\text{Author Ans}|}$$

Comparison functions are passed through the 9<sup>th</sup> argument of `\RespBoxMath`, if no such argument is passed, `diffCompare` is used.

#### • `indefIntegral`

The answer to an indefinite integral is non-unique; however all answers differ by a constant. This compare function is used with indefinite integrals. Note the name of the function is `indefCompare`, this is the name you use to call the function. To use this compare function, insert its name in optional 9<sup>th</sup> argument,

```
\RespBoxMath[#1]#2(#3)[#4]#5#6#7#8[#9]*#10
```

that's the argument that follows the interval specification.

An example of usage of this function can be found in the sample file `jslib_ex.tex` that comes with the AcroTeX Bundle; from that file we have:

```


$$\int \sin(x) dx =$$

\RespBoxMath{-cos(x)}{4}{.0001}{[0,1]}[indefCompare]$

```

See the demo file `jslib_ex.tex` for an example of indefinite integration comparison. Numerous other files use indefinite integration as well.

#### • `satisfyEq`

The `satisfyEq` library of procedures supports the type of question where the student is asked to enter one or more points that satisfy an equation.

We pose a problem where the student is asked to enter an order pair or triple that satisfies an equation. For example

- ▶ Enter a point that lies on the line  $2x + 3y = 6$ , the answer is an ordered pair of numbers, for example (1, 2).

A point is

The verbatim listing follows:

```

1 \RespBoxMath[\rectW{1.5in}\textSize{0}]{2x+3y-6}(xy){1}
2   {.0001}{\viidna}*{ProcRespEvalEq}\kern1bp
3   \CorrAnsButton{various, such as (0,2)}

```

Line (2) shows the placement of one of the procedures, `ProcRespEvalEq`, that is part of `satisfyEq`.

There are four procedures in this set:


1. `ProcRespEvalEq`: Supports a single equation, the answer is a single ordered  $n$ -tuple, the entries are allowed to be zero.
2. `ProcRespEvalEqNonZero`: Supports a single equation, the answer is a single ordered  $n$ -tuple, the entries are *not allowed* to be zero.
3. `ProcRespEvalEqList`: Supports a single equation, the answer consists of  $k$ ,  $n$ -tuples, the components of the  $n$ -tuples are allowed to be zero. Each  $n$ -tuple is separated by a semi-colon (;).
4. `ProcRespEvalEqListNonZero`: Supports a single equation, the answer consists of  $k$ , ordered  $n$ -tuples, the components of the  $n$ -tuples are *not allowed* to be zero. Each  $n$ -tuple is separated by a semi-colon (;).

**Some Technical Details.** Recall the parameter list for `\RespBoxMath`,

```
\RespBoxMath[#1]#2(#3)[#4]#5#6#7#8[#9]*#10
```

`ProcRespEvalEq`, *et al*, are not based on random point generation, as are all the other `exerquiz` functions. Here, the user enters numerical data, the function then verifies the data entered satisfies the given equation. The role of one of the parameters has changed, others are ignored:

- The role of parameter #2 (the correct answer) has changed. Suppose the equation presented to the user is  $F = G$ .
  - For `ProcRespEvalEq` and `ProcRespEvalEqNonZero`, pass  $F - G$  as parameter #2.
  - For `ProcRespEvalEqList` and `ProcRespEvalEqListNonZero`, pass  $k;F-G$  as the parameter #2. For this type of problem, the author is asking the user to enter  $k$ ,  $n$ -tuples that satisfy the equation. Note  $k$  and  $F-G$  are delimited by a semicolon (;).
- The value of `epsilon` (parameter #6) is obeyed.
- Parameter #5, the number of iterations, is not used. Though this parameter is not used, #5 is required by the macro, so just use 1 as the number of iterations.
- Parameter #7, the domain of the variables, is not used. Again, since this is a required parameter, use `\viidna`; `\viidna` is a special command that can be used for this parameter, in this context only.
- The optional parameter #9 is not used and is ignored if present.

 Additional examples will (eventually) appear on the [AcroTeX Blog](#).

## 29.4. Filter User's Responses

Filters are JavaScript functions that process the user input before being passed to the response function. Filters are passed to `\RespBoxMath` through the 9<sup>th</sup> argument,

$$\int \sin(x) dx = -\cos(x)$$

```
\RespBoxMath{-cos(x)}{4}{.0001}{[0,1]}[indefCompare]$
```

This 9<sup>th</sup> argument can take on two formats: (1) the name of the compare function, as illustrated in `'indefIntegral'` above; (2) a JavaScript object. Use a JavaScript object as the 9<sup>th</sup> argument to specify a filter.

The JavaScript object that can be passed in the 9<sup>th</sup> argument has two properties `comp` and `priorParse`. The object has the form

```
{ comp: <compare>, priorParse: <JS_func> | <array_JS_funcs> }
```

The first property is used to specify a compare function, the second property specifies a filter or filters. The value of `priorParse` may be a JavaScript function to filter the user's response, or an array of JavaScript functions to filter the user input. (The array must be specified using the `\Array` command, illustrated below, see the paragraph `'NoProducts'`, on page 151.)

- **nodec**

The `nodec` option defines the JavaScript function `nodec` that attempts to prohibit the use of decimal numbers in the response. The function simply search the user response for any occurrence of the decimal point. (The German language uses the comma as a decimal point, you can set the decimal point to some other symbol that `nodec` searches for through the command `\setdecimalpoint`, the default definition is `\setdecimalpoint{.}`.)

```
$ 0.75 = \RespBoxMath{3/4}{1}{.0001}{[0,1]}[{priorParse: nodec}] $
```

Alert box message: May be redefined.

```
\newcommand\nodecAlertMsg{%
  "A decimal answer is not acceptable here
  Please express your answer using fractions, square roots,
  e, log, etc."}
```

- **noBinFac**

The `noBinFac` option defines the JavaScript filter function `noBinFac`, that disallows binomial coefficients and factorials in math fill-in questions. Binomial and factorial functions are defined through the `combinatorics` option, page 153.

The number of ways to choose four object from a set of ten is

```
\RespBoxMath{C(10,4)}{1}{.0001}{[0,1]}[{priorParse: noBinFac}]
\CorrAnsButton{C(10,4)}*{EvalCorrAnsButton}
```

The `noBinFac` filter is used, so the student is not allowed to enter binomial coefficients; however, the document author can use binomial coefficients, note the use in the answer (first argument) and in the `\CorrAnsButton`. The special syntax used for `\CorrAnsButton`, '[Including a Solution](#)' on page 110.

Illustrations of this filter can be found in the `dljs_comb.tex`.

Alert box messages: Mayb be redefined.

```
\newcommand\noBinFactBinCoeffAlertMsg{%
  "You may not use this notation here.
  Please evaluate the binomial coefficient.
  You may present your answer as a product rather
  than calculating a very large number."}
\newcommand\noBinFactPermAlertMsg{%
  "You may not use this notation here.
  Please evaluate the permutation.
  You may present your answer as a product rather
  than calculating a very large number."}
\newcommand\noBinFactFactAlertMsg{%
  "You may not use this notation here.
  Please evaluate the factorial.
  You may present your answer as a product rather
  than calculating a very large number."}
```

#### • `limitArith`

The `limitArith` option defines a number of filter functions:

```
DecimalsOnly, NoProducts, NoDivision, NoAddOrSub,
NoArithAllowed, NoExpAllowed, NoTrigAllowed, NoTrigLogAllowed
```

Examples of these filtering functions can be found in the demo file `limarth.tex`.

**DecimalsOnly** The `DecimalsOnly` is a function that takes only numbers, either integer or decimal. Usage

```
$ 2.3 + 4.5 = \RespBoxMath{6.8}{1}{.0001}{{[0,1]}
  [{}priorParse: DecimalsOnly ]}$
```

**NoProducts** The function `NoProducts` requires that `Imp]Multi` is taken. An example of usage is

```
$ 3/4 = \RespBoxMath{.75}{1}{.0001}{{[0,1]}
  [{}priorParse: \Array(NoDivision, NoProducts) ]}$
```

In the above example, we first call `NoDivision` then `NoProducts`. Several filtering functions can be executed by putting them into an array, using the `\Array` command.<sup>21</sup>

<sup>21</sup>The `\Array` command was originally named `\array`, it has been renamed to avoid clashing with the `amsmath` package.

**NoAddOrSub** This function, as the name implies, denies the use of both addition and subtraction. The algorithm I use here is that if a plus or minus follows anything but a left parenthesis, we assume it is a binary operation of addition or subtraction, respectively.

```
$ 5.1 - 3.2 =
  \RespBoxMath{1.9}{1}{.0001}{[0,1]}[{\priorParse: NoAddOrSub}]$
```

**NoArithAllowed** Deny user the use of all arithmetic functions, including exponents.

```
$ 6/5 = \RespBoxMath{1.25}{1}{.0001}{[0,1]}
  [priorParse: NoArithAllowed]$
```

**NoExpAllowed** Deny user the use of exponents.

```
$ 25^2 = \RespBoxMath{625}{1}{.0001}{[0,1]}
  [priorParse: NoExpAllowed]
```

**NoTrigAllowed** Deny user the use all trig functions, as well as the constants PI and pi, which are aliases for  $\pi$ . Example:

```
$ \sin(\pi/4) = \RespBoxMath{sqrt(2)/2}{1}{.0001}{[0,1]}
  [priorParse: NoTrigAllowed ]$
```

**NoTrigLogAllowed** Deny the use of trig and log functions.

```
$ \ln(e^2)\sin(\pi/4) = \RespBoxMath{sqrt(2)}{1}{.0001}{[0,1]}
  [priorParse: NoTrigLogAllowed ]\CorrAnsButton{sqrt(2)}$
```

**sciNotResp** Allow only a response in *normalized* scientific notation:  $aE([+|-]d)$ , where  $1 \leq |a| \leq 9$ , and  $d$  is one or more digits (0-9) prefixed by a -, or an optional +.

```
Convert 123.4 to scientific notation.
\begin{equation*}
  123.4 = \RespBoxMath{1.234E+2}*{1}{0}{[0,1]}[%
  {priorParse: sciNotResp }]\CorrAnsButton{1.234E+2}
\end{equation*}
```

**noNegExp** Does not allow negative exponents. For example,

```
\item Simplify $\displaystyle\frac{4}{r^4}\cdot\frac{r^3}{12}=
  \RespBoxMath{1/(3r)}(r)*{3}{.0001}{[0,1]}[{\%
  priorParse:\Array( nodedc, noNegExp )
  }]\CorrAnsButton{1/(3r)}$
```

**Alert box messages:** May be redefined.

```
\newcommand\DecimalsOnlyErrorMsg{"Enter only an integer,
  e.g., 17, or a decimal number, e.g. 12.4.
  Using arithmetic operations or built in function is
```

```

    not acceptable for this problem."}
\newcommand\NoProductsErrorMsg{"Multiplication is not allowed
for this problem."}
\newcommand\NoDivisionErrorMsg{"Division is not allowed
for this problem."}
\newcommand\NoAddOrSubErrorMsg{"Neither addition nor subtraction
is not allowed for this problem."}
\newcommand\NoExpAllowedErrorMsg{"The use of exponents is not
allowed for this problem."}
\newcommand\NoTrigAllowedErrorMsg{"The use of trig functions
in this problem is not allowed."}
\newcommand\NoTrigLogAllowedErrorMsg{"The use of trig and log
functions is not allowed in this problem."}
\newcommand{\sciNotSyntaxError}{"Enter the answer in scientific
notation."}
\newcommand{\sciNotNormalForm}{"The scientific notation entered
is not in normalized form."}
\newcommand{\NoNegExpMsg}{"No negative exponents permitted,
keep working!"}

```

### 29.5. Extending Numerical Functions

These JavaScript functions extend the collection of built-in functions that perform numerical calculations.


- **combinatorics**

This option defines three JavaScript functions: a binomial coefficient function  $\text{ch}(n, r)$ ; a number of permutations function  $\text{perm}(n, r)$ , and a factorial function  $\text{fact}(n)$ .

**User Accessible Functions** The user should use the names  $C$ ,  $P$  and  $\text{fact}$  to access the functions are  $\text{ch}$ ,  $\text{perm}$  and  $\text{fact}$ , respectively:

- $C(n, r) = \text{ch}(n, r) = \binom{n}{r} = \frac{n!}{r!(n-r)!}$
- $P(n, r) = \text{perm}(n, r) = n \cdot (n-1) \cdots (n-r)$
- $\text{fact}(n) = n \cdot (n-1) \cdots 3 \cdot 2 \cdot 1$

**Note:** If you place `\replaceExclPt{true}` in the preamble, the user is allowed to enter factorials using the traditional exclamation notation: for example,  $17!$  is the same as  $\text{fact}(17)$ , and  $(17)!$ ,  $[17]!$  or  $\{17\}!$  are all interpreted as  $\text{fact}(17)$ .

 An illustrations of the `combinatorics` option can be found in the `d1js_comb.tex`.

### 30. List of Options

Options of the Web/Exerquiz Packages	
Options of the Web Package	
<code>dvipsone</code>	dvi-to-ps driver by Y&Y, Inc.
<code>dvips</code>	dvi-to-ps driver
<code>pdftex</code>	tex-to-pdf application
<code>dvipdfm</code>	dvi-to-pdf application
<code>textures</code>	the Textures System for Mac
<code>designi</code> , <code>designii</code> , <code>designiii</code> , <code>designiv</code> , <code>designv</code>	these set screen design parameters
<code>usetemplates</code>	this option activates mechanism for creating colored backgrounds and overlays
<code>leftpanel</code>	creates a left navigation panel
<code>rightpanel</code>	creates a right navigation panel
<code>navibar</code>	inserts a menu bar at the bottom of each page
<code>latextoc</code>	displays the standard toc
<code>usedirectory</code>	causes the title page directory to appear on the title page
<code>forpaper</code>	this turns off color, and does not put solutions on separate pages.
<code>forcolorpaper</code>	Same as <code>forpaper</code> , but does not turn off color, useful for color printers.
<code>latexlayout</code>	web uses page layout for <code>article</code> class. For use with <code>forpaper</code> .
<code>tight</code>	redefines list environment parameters so lists don't take up so much space
<code>pro</code>	causes web to input enhanced control over various web elements
<code>nobullets</code>	forces the use of numbers for subsections, also executes the <code>latextoc</code> option
<code>usesf</code>	switches to sans serif as the default font, good for presentations
<code>unicode</code>	Passes the <code>unicode</code> option to <code>hyperref</code> .
<code>useui</code>	Passes the <code>useui</code> option to <code>eforms</code> .

Options of the Web/Exerquiz Packages (cont.)	
<code>dutch</code>	Dutch for web, passed to exerquiz. (Thanks to Henny Wilbrink)
<code>french</code>	French for web, passed to exerquiz. (Thanks to Jean-Michel Sarlat)
<code>german</code>	German for web, passed to exerquiz. (Thanks to Michael Wiedmann)
<code>italian</code>	Italian for web, passed to exerquiz. (Thanks to PierLuigi Zezza)
<code>norsk</code>	Norwegian for web, passed to exerquiz. (Thanks to Hans Fredrik Nordhaug)
<code>russian</code>	Russian for web, passed to exerquiz. (Thanks to Sergei V. Znamenskii)
<code>spanish</code>	Spanish for web, passed to exerquiz. (Thanks to Pedro Luis Luque)
<code>polish</code>	Polish for web, passed to exerquiz. (Thanks to Jerzy Mycielski)
<code>finnish</code>	Finnish for web, passed to exerquiz. (Thanks to Päivi Porras)
<code>catalan</code>	Catalan for web, passed to exerquiz. (Thanks to Ramon Ballester)
<code>czech</code>	Czech for web, passed to exerquiz. (Thanks to Robert Marik)
<code>brazil</code>	Brazilian Portuguese for web, passed to exerquiz. (Thanks to Koichi Sameshima)
<code>turkish</code>	Turkish for web, passed to exerquiz. (Thanks to Mahmut Koçak)
Options of the Exerquiz Package	
<code>pdftex</code>	tex-to-pdf application
<code>dvipdfm</code>	dvi-to-pdf application
<code>nosolutions</code>	removes the solutions to the exercises
<code>noquizsolutions</code>	removes the solutions to the quizzes
<code>nohiddensolutions</code>	overrides the 'h' (hidden) option for the exercises.
<code>noHiddensolutions</code>	overrides the 'h' and 'H' (hidden) options for the exercises.
<code>nocorrections</code>	removes the ability to correct the quizzes
<code>solutionsafter</code>	solutions to exercises are typeset just after the question

Options of the Web/Exerquiz Packages (cont.)	
<code>forpaper</code>	same function as in <code>web</code> . Needed when <code>exerquiz</code> is not used with <code>web</code>
<code>forcolorpaper</code>	same function as in <code>web</code> . Needed when <code>exerquiz</code> is not used with <code>web</code>
<code>preview</code>	shows the outline of all form fields in the dvi previewer
<code>nodljs</code>	turns off the insertion of DLJS
<code>exercisonly</code>	if document has only exercises, no doc level JS needed
<code>debug</code>	this option is passed on to the <code>insdljs</code> package
<code>proofing</code>	mark the correct answers for <code>shortquiz</code> & <code>quiz</code> for proof reading.
<code>showgrayletters</code>	Show gray letters under check boxes
<code>allowrandomize</code>	Allow the randomization of choices in multiple choice questions
<code>unicode</code>	Passes the <code>unicode</code> option to <code>hyperref</code> .
<code>useui</code>	Passes the <code>useui</code> option to <code>eforms</code> .
<code>usesumrytbls</code>	Inputs the code used to create quiz summary tables, <a href="#">Section 24</a> , page 126.
<code>dutch</code>	JavaScript messages in Dutch (Thanks to Henny Wilbrink)
<code>french</code>	JavaScript messages in French (Thanks to Jean-Michel Sarlat)
<code>german</code>	JavaScript messages in German (Thanks to Michael Wiedmann)
<code>italian</code>	JavaScript messages in Italian (Thanks to PierLuigi Zezza)
<code>norsk</code>	JavaScript messages in Norwegian (Thanks to Hans Fredrik Nordhaug)
<code>russian</code>	JavaScript messages in Russian (Thanks to Sergei V. Znamenskii)
<code>spanish</code>	JavaScript messages in Spanish (Thanks to Pedro Luis Luque)
<code>polish</code>	JavaScript messages in Polish (Thanks to Jerzy Mycielski)
<code>finnish</code>	JavaScript messages in Finnish. (Thanks to Päivi Porras)

Options of the Web/Exerquiz Packages (cont.)	
<code>catalan</code>	JavaScript messages in Catalan. (Thanks to Ramon Ballester)
<code>czech</code>	JavaScript messages in Czech. (Thanks to Robert Marik)
<code>brazil</code>	JavaScript messages in Brazilian Portuguese. (Thanks to Koichi Sameshima)
<code>turkish</code>	JavaScript messages in Turkish. (Thanks to Mahmut Koçak)

## Solutions to Exercises

**Exercise 1.** We evaluate by integration by parts:

$$\begin{aligned}
 \int x^2 e^{2x} dx &= \frac{1}{2} x^2 e^{2x} - \int x e^{2x} dx && u = x^2, dv = e^{2x} dx \\
 &= \frac{1}{2} x^2 e^{2x} - \left[ \frac{1}{2} x e^{2x} - \int \frac{1}{2} e^{2x} dx \right] && \text{integration by parts} \\
 &= \frac{1}{2} x^2 e^{2x} - \frac{1}{2} x e^{2x} + \frac{1}{2} \int e^{2x} dx && u = x^2, dv = e^{2x} dx \\
 &= \frac{1}{2} x^2 e^{2x} - \frac{1}{2} x e^{2x} + \frac{1}{4} e^{2x} && \text{integration by parts} \\
 &= \frac{1}{4} (2x^2 - 2x + 1) e^{2x} && \text{simplify!}
 \end{aligned}$$

Exercise 1

**Exercise 2.**

$$x + y = 1$$

Exercise 2

**Exercise 3(a)** Velocity is the rate of change of position with respect to time. In symbols:

$$v = \frac{ds}{dt}$$

For our problem, we have

$$v = \frac{ds}{dt} = \frac{d}{dt} (t^2 - 5t + 1) = 2t - 5.$$

The velocity at time  $t$  is given by  $v = 2t - 5$ . □

**Exercise 3(b)** Acceleration is the rate of change of velocity with respect to time. Thus,

$$a = \frac{dv}{dt}$$

For our problem, we have

$$a = \frac{dv}{dt} = \frac{d}{dt} (2t - 5) = 2.$$

The acceleration at time  $t$  is constant:  $a = 2$ . □

**Exercise 4(a)**  $i^2 = -1$  □

**Exercise 4(b)**  $i^3 = ii^2 = -i$  □

**Exercise 4(c)**  $z + \bar{z} = \text{Re } z$  □

**Exercise 4(d)**  $\frac{1}{z} = \frac{1}{z} \frac{\bar{z}}{\bar{z}} = \frac{\bar{z}}{z\bar{z}} = \frac{\bar{z}}{|z|^2}$  □

**Exercise 6(a)**  $v = 2t - 5$ . □

**Problem 8.** This is the solution.



**Exercise 9.** It is well known that  $2 + 2 = 4$ .

Exercise 9

**Project Hint:** There, you didn't need my help after all.



## Solutions to Quizzes

**Solution to Quiz:** The answer is ‘Yes’. The definition requires that

$$F'(x) = f(x) \quad \text{for all } x,$$

well, let’s check it out.

The definition of  $f$  is  $f(s) = 4s^3$  and so  $f(x) = 4x^3$ . The definition of  $F$  is  $F(t) = t^4$  and so, by the rules of differentiation,  $F'(t) = 4t^3$ . Thus,  $F'(x) = 4x^3$ . Therefore,

$$F'(x) = 4x^3 = f(x) \quad \text{for all } x,$$

as required by the definition. ■

**Solution to Quiz:** If you erred on this one, more than likely it was on the appropriate multiplicative constant: 6 not 18. At least that’s what I’m betting on.

The instructions of the LCD Algorithm said to *completely factor the denominator*. Here’s a list of the factors

$$\underbrace{3, x^{3/2}, y^2}_{\text{first term}}, \underbrace{2, 3, x, y^4}_{\text{second term}}$$

Let’s rearrange them

$$2, 3, 3, x, x^{3/2}, y^2, y^4$$

Now drop duplicate factors—that’s the 3. Oops! I did mention dropping identical factors, didn’t I?

$$2, 3, x, x^{3/2}, y^2, y^4$$

Now, group together all terms which have the same base, then drop, from each of these groups all terms but the one with the highest power. We obtain then,

$$2, 3, x^{3/2}, y^4$$

The LCD is the product of same:

$$\text{LCD} = (2)(3)x^{3/2}y^4 = 6x^{3/2}y^4.$$

*Solution Notes:* Alternative (a) will work as a common denominator, but it is not the least common denominator. If you use (a), you will be working with larger numbers than is really necessary. ■

**Solution to Quiz:** The answer is true, of course, math is fun! ■

**Solution to Quiz:** The answer is false, statistics is interesting but not fun. ■

**Solution to Quiz:** Yes, Donald Knuth was the creator of  $\text{\TeX}$ . ■

**Solution to Quiz:** Yes, Leslie Lamport was the creator of  $\text{\LaTeX}$ . ■

**Solution to Quiz:** The answer is true, of course, math is fun! ■

**Solution to Quiz:** The answer is false, statistics is interesting but not fun. ■

**Solution to Quiz:** You can’t understand it until I explain it! ■

**Solution to Quiz:**

$$\frac{d}{dx} \sin^2(x) = 2 \sin(x) \cos(x) = \sin(2x)$$

■

**Solution to Quiz:**

$$\frac{d}{dx} \sin^2(x) = 2 \sin(x) \cos(x) = \sin(2x)$$

■

**Solution to Quiz:** Yes, Isaac Newton and Gottfried Leibniz are considered founders of Calculus.

■

**Solution to Quiz:**

$$\frac{d}{dx} \sin^2(x) = 2 \sin(x) \cos(x) = \sin(2x)$$

■

**Solution to Quiz:** Yes, Isaac Newton and Gottfried Leibniz are considered founders of Calculus.

■

## References

- [1] Acrobat JavaScript Scripting Reference, Technical Note #5431, Version 6.0., Adobe Systems, Inc., 200  
<http://partners.adobe.com/asn/acrobat/docs.jsp> 132
- [2] Leslie Lamport, *LaTeX: A Document Preparation System* (2nd ed.), Addison-Wesley Publishing Company, 1994, ISBN 0-201-52983-1.
- [3] Michel Goossens, Frank Mittelbach and Alexander Samarin, *The LaTeX Companion*, Addison-Wesley Publishing Company, 1994, ISBN 0-201-54199-8. 68
- [4] Michel Goossens, Sebastian Rahtz, and Frank Mittelbach, *The LaTeX Graphics Companion*, Addison-Wesley Publishing Company, 1997, ISBN 0-201-85469-4.
- [5] Michel Goossens, and Rahtz, Sebastian, with Gurari, Eitan, Moore, Ross, and Sutor, Robert, *The LaTeX Web Companion*, Addison Wesley Longman, Reading, Massachusetts, USA, 1999. ISBN: 0-201-43311-7. 19
- [6] Helmut Kopka and Patrick W. Daly, *A Guide to LaTeX2e* (2nd ed.), Addison-Wesley Publishing Company, 1995, ISBN 0-201-43777-X.
- [7] Donald E. Knuth, *The TeXbook*, Addison-Wesley Publishing Company, 1987, ISBN 0-201-13448-9.
- [8] Thomas Merz, *Web Publishing with Acrobat/PDF*, Springer-Verlag Berlin Heidelberg New York, 1998, ISBN 3-540-63762-1.
- [9] D.P. Story, *Pdfmarks: Links and Forms*, AcroTeX Web Site, 1998,  
[www.math.uakron.edu/~dpstory/acrotex.html](http://www.math.uakron.edu/~dpstory/acrotex.html)
- [10] D.P. Story, *Using LaTeX to Create Quality PDF Documents for the WWW*, AcroTeX Web Site, 1998,  
[www.math.uakron.edu/~dpstory/acrotex.html](http://www.math.uakron.edu/~dpstory/acrotex.html) 15