AcroTEX Software Development Team

The AcroTEX eDucation Bundle

D. P. Story

Prepared: May 10, 2007

Version 6.05

Table of Contents

Preface

1. Introduction

The AcroTeX eDucation Bundle, read "AcroTeX Education Bundle", is a collection of LaTeX macro files, along with various support files and sample files. The overall theme of this bundle 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.

Currently, there are three components to the bundle, with others planned:

1. The web package is used to create an attractive, easy-on-the-eye page layout suitable for the `www` (or classroom/conference presentations).
2. The exerquiz package makes it very easy to create interactive exercises and quizzes.
3. The eforms package provides support for PDF form fields.
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, and the demo file `execJStst.tex`, which features an animation built completely from LaTeX source lines.
5. 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`, and try the test file for this package, `jslib_ex.tex`.
6. The eqExam Package is a stand alone LaTeX package for creating exams, quizzes, homework assignments. 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 AcroTeX Bundle should be useful to educators who want to post interactive materials for their students on the `www`.

Here is an important point that should be emphasized early in this manual. AcroTeX only supports three ways of producing a `pdf` document: (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 document from AcroTeX; this will not work! (You will get the `pdf` document but not much functionality.)

Please contact me at `dpstory@acrotex.net` should you encounter any problems, or have suggestions to make.

► See 'Getting Started' on page 8 for instructions on how to get up and running.

1.1. A Brief History

The web and exerquiz packages were written in preparation for a two-day workshop on LaTeX/PDF that I gave at the College of the Redwoods, April 30-May 1, 1999, at the invitation of David Arnold.

The workshop forced me to take many of the basic macros that I had developed in plain TEX and convert them to LATEX.

Significant additions to the exerquiz were made immediately following the 20th Annual Conference of the TEX User's Group (tug), in August, 1999, Vancouver, British Columbia, which I attended.

The insdljs package was written for the 22nd Annual Conference of the TEX User's Group (tug), in August 2001, at the University of Delaware, Newark, Delaware.

The execJS environment was created as a result of some work I did for the Seybold SF Convention, 2002 (see article at planetPDF); the techniques were first presented to the PDF public at this convention. My complete presentation from that conference is again available from planetPDF.

1.2. Thanks

Noel Vaillant, www.probability.net, deserves my thanks for his enthusiasm for the web style file and his initial work on it inspired me to make a serious effort at writing a LATEX package.

Thanks also goes out to Jean-Michel Sarlat for writing a French version of the web and exerquiz packages. See his Syracuse Web site. He urged me to include a language option. Thanks also goes to Michael Wiedmann who suggested a language option many months earlier, but I'm afraid it landed on deaf ears at the time. These two provided the translations for the french and german options. (January 1, 2000)

My thanks to Heiko Oberdiek, who took a close look at insdljs. He made several suggestions, and urged me to make some significant improvements to this package.

2. Getting Started

There has been a new package added to the AcroTEX Bundle, the insdljs package. This package allows the document or package author to write JavaScripts to the document level JavaScript section of a PDF document. Exerquiz now uses insdljs to place its JavaScripts into the PDF document.

► The LATEX package files for AcroTEX Bundle consist of web.dtx, web.ins, exerquiz.dtx and the installation files for each of these packages. Additionally, acrotex.ins installs the whole distribution.

1. Place all these files in the same directory. This directory must be in the search path of your LATEX system, perhaps in a separate folder called acrotex.
2. The whole bundle 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.

   Important: See the next section, Unpacking the AcroTEX Bundle for information on unpacking the bundle.
3. Place the sample files either in the same folder as the AcroTEX program files, or in another folder of your choosing. See the section titled 'Sample Files' on page 9 for more details on these.

After reading the manual you are then ready to write your own set of tutorials, exams, exercises and quizzes.

2.1. Unpacking the AcroTEX Bundle

To install the AcroTEX Bundle, you must first "unpack" it. Unpacking is performed by "LATEXing" the file `acrotex.ins`. Simply execute `latex acrotex.ins` from the command line (the command line may vary depending on your TEX System), or if you use a TEX/LATEX friendly editor, open the file in the editor and `latex` it.

• Language Localizations

In `acrotex.ins`, the language localizations have been commented out. Just uncomment the language you intend to use.

▶ Also in the `exerquiz.ins` file is the line

```
%  \file{template.def}{\from{exerquiz.dtx}{copyright,template}}
```

Uncomment this line to get the template file, used for developing language localizations.

• 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.

Start **Acrobat Pro 8.1**[1] 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, this 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/7.0/JavaScripts
```

Follow the path to this folder. If the `JavaScripts` folder does not exist, create it. Finally, copy `aeb.js` into this folder. Close **Acrobat**, then start it again and look under the Tools menu to verify the presence of the sub-menu AcroTEX Presentation Bundle.

2.2. Sample Files

There are numerous sample/demo files that illustrate various features of the AcroTEX Bundle. View the `pdf` document `indexofex.pdf`, which is an "Index of AcroTEX Examples". This document contains a list of all examples, a short description of each, and links to the `pdf` document and source file.

[1] The file will cause no harm if you install it in an earlier version of Acrobat, as early as version 7.

2.3. 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 $\mathcal{AMS}$-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

2.4. LaTeXing 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` document, 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 `pdflatex`, and you have your nice `pdf` document, 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` document.

• For Distiller Users

If you use the distiller, as I do, the sophisticated features of AcroTeX Bundle require Acrobat 5.0 or higher. I've discontinued my attempt at supporting Acrobat 4.0.

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

```
\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. The `.fdf` files (e.g., `exerquiz.fdf`) contain the document level JavaScript that needs to be imported into your document.

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 (e.g., `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.

For distiller users, the AcroTeX eDucation Bundle has many exciting features—the `insdljs` and `dljslib` Packages—whose functionality requires the document author use Acrobat 5.0 or higher.

The Web Package

3. 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.

3.1. 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`, and `brazil`. 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.

3.2. 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 navigational 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 LaTeX to Create Quality PDF Documents for the WWW*", see reference [10].

4. 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 26 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, (http://odo.kettering.edu/dvipdfm/); and the commercial TEX system for the Mac, textures and TEXshop.

▶ 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.

4.1. 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 LATEX 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 TEX System for the dvipsone option and a MiKTEX System (www.miktex.org) for the dvips, pdftex and dvipdfm options.

4.2. 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.

4.3. The usesf Option

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

4.4. 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.

4.5. The `nobullets` Option

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

5. 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 sceen package pdfscreen.); or (2) use a screen design option. The next two sections address each of these in turn.

5.1. 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 LATEX 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.

5.2. Screen Design Options

For your convenience, I've included six options, designi, designii, …and (you guessed it) designvi. The first one roughly corresponds to the original screen dimensions of web. The other two set the screen dimensions at 4.5 in × 5 in and 5 in × 6 in (height × width), respectively. You can type

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

to obtain the standard web dimensions.

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

5.3. 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 5.1, page 14. 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 14.3, page 28, for more details on this command.

6. 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[dvipsone]{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 LaTeXing the file `hyperref.dtx`, or by getting a copy of the *The LaTeX Web Companion* [5] by Michel Goossens *et al.*

7. Running Headers

Beginning with version 3.3 of Web, 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).

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.

8. 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 in effect, most all colors can be specified through the `\selectColors` command, see 'Selecting Colors' on page 38.

8.1. Basic Information Commands

Just fill in the values of all the basic macros briefly described above. 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{Okaloosa-Walton College\\
   Department of Mathematics}
\email{dpstory@acrotex.net}
\version{1.30}
\copyrightyears{1999-2007}
```

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[2] 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, 16,use `\texorpdfstring` only with the `\title`, `\author`, `\subject` and `\keywords`, all of which are used in the DocInfo field of the Adobe Reader.

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

Having entered the information you can now type the standard sort of LaTeX commands of `\maketitle` and `\tableofcontents`:

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

🗎 Use the file `webeqtst.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 30.

[2] The code for handling PDFDocEncoding for hyperref is due to Heiko Oberdiek

8.2. 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 the \optionalPageMatter command does not appear in the preamble, then this part of the title page is empty. Here is an example of usage:

```
\optionalPageMatter{\vfill
    \begin{center}
    \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@ul{\web@copyright\ \web@copyright@symbol\
    \webcopyrightyears}
\def\maketitle@trailer@ll{\web@revision\ \@date}
\def\maketitle@trailer@ur{\thewebemail}
\def\maketitle@trailer@lr{\webversion}
```

The ul corner: The macro \web@copyright expands to 'Copyright', it can be redefined, or made to go away by executing \nocopyright. \webcopyrightyears expands to the argument of \copyrightyears. the final element in the upper left is the email address. To make everything go away except for the email address, executed \nocopyrightNotice.

The ll 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 goes into the lower-right corner. The command expands to \web@versionlabel\ <version_number>. \web@versionlabel expands to "Version". This label can be changed with \versionLabel (for example, versionLabel{Attempt}). The <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 36.

8.3. Redefining \maketitle

The arguments of the Basic Information Commands 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 the \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.

When making the design, it is useful to know that the web package uses \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 15.2, 'The Title Page: \maketitle' on page 31.

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

Table 1: The Basic Information Macros

8.4. 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 `latextoc` option can be used.

8.5. The `nodirectory` Option

The inclusion of `\tableofcontents` is optional. The article you write may be short, or perhaps it may just be a collection of exercises and quizzes. In this case, you may not want a table of contents.

If you do not want a table of contents, you would not include `\tableofcontents` just after `\begin{document}`. Without a table of contents, you may as well turn off the directory listing on the cover page as well. Use the `nodirectory` option to do this:

```
\usepackage[dvips,nodirectory]{web} % dvipsone, pdftex
```

The directory listing does not appear on the title page.

8.6. The `latextoc` 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 `latextoc` option with the web package:

```
\usepackage[latextoc]{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.

8.7. 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.

8.8. The `\makeinlinetitle` Command

For some short documents,[3] 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:

```
                               \webtitle
\webuniversity                 \webauthor                    \webemail
date                                                       \webversion
```

The table above extends the entire `\linewidth`, just as the expansion of `\makeinlinetitle` does. The values of `\webtitle`, `\webuniversity`, `\webauthor`, `\webmail` and `\webversion` are populated by their command counterparts, see Table 1, page 20. 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 15.1, entitled '`\DeclareDocInfo`' on page 30.
- 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. If `talksite` is specified, the value of this key is used instead of `\webversion`.

[3] Short documents such as white papers, homework assignments, for example.

9. Template Options

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

9.1. 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 26.

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

9.2. The `leftpanel` and `rightpanel` Options

When either of the these two options is specified, a vertical panel is created. See the section entitled 'Template Building and Management' on page 26 for a complete discussion of the commands related to these options.

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

10. Navigational Aids

The web package offers a couple of navigational aids to help you move around: the `navibar` Option, and some direction icons.

10.1. A Navigational Bar

Use the `navibar` option of web to add a navigational 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 navigational 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 through the commands `\navibarTextColor` and `\navibarBgColor`. Each take a single argument, a named color. The default colors are `webblue` and `webgray`, respectively.

10.2. \newNaviIcon

The \newNaviIcon can be used to define a navigational 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 navigational 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 through the commands \navibarTextColor and \navibarBgColor.

10.3. 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}}}
```

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.

10.4. `\panelNaviGroup`

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

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

11. 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` and `brazil`.

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.

12. Paper Related Options

12.1. 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 LaTeX document. The `\textwidth` is determined by the `\screensize` and `\margins` parameters or by the design option (see Screen Design Options); consequently, the line breaks are the same for the "web" version and the "print" version.

Using the forpaper option with the latexlayout option will give you the standard 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.

12.2. The forcolorpaper Option

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

12.3. The latexlayout Option

For those who want to go "totally native", use the latexlayout option with the forpaper option. 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 forcoloroption is executed. To remove color, you need to explicitly take the forpaper option.

13. Formatting for screen and paper

As we learned in Section 12, Web can format a document in a for screen or for paper. Web 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.

14. Template Building and Management

The Web Package now 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 (navigational) side panel, define your own navigational 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`.

14.1. 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.

14.2. 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 overlayed with the \AddToTemplate command.

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

The command takes one argument, the *template_name*. Define an overlay using \newcommand,

```
\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 overlayed onto the page, we would type

```
\AddToTemplate{myTemplate}
```

► Example: Insert the "AcroTeX" 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 "AcroTeX" logo at coordinates of (36,36) relative to the lower-left corner. The units are measured in (TeX) 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 28 for details of how to manage your templates.

14.3. Panel Screen Template

When the `leftpanel` or `rightpanel` option is specified, a (navigational/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.

▶ Additional overlays can be added with `\AddToPanelTemplate`. This command, which works the same as `\AddToPanelTemplate`, may not be as useful as `\AddToPanelTemplate` 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.

```
\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`.

14.4. 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 which were inserted into the output stream with the command `\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 overlayed 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.

```
\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 `\ClearAllTemplates` command is equivalent to executing both `\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.

15. 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.

New features provided by the `pro` option are

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 one 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.

15.1. `\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 (LaTeXed). This was formerly the last revision date.
14. `preparedLabel`: The label that precedes the date `prepared`. The default 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, 2007}
    ```

    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.

Example 1: Example of usage of `\DeclareDocInfo` can be found in Figure 1, page 32.

☛ 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.

15.2. The Title Page: `\maketitle`

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

```
\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={LaTeX, PDF, AcroTeX},
    copyrightStatus=False,
    copyrightNotice={Copyright D. P. Story, \the\year},
    copyrightInfoURL=http://www.acrotex.net
}
```

Figure 1: The `\DeclareDocInfo` Command

• 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 the `\optionalPageMatter` command 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, listed in `\titlepageTrailer` are the values of the `\DeclareDocInfo` keys `email`, `talkdate`, `talksite`, `copyrightyears`, as described above. The font size of this bottom part is set by `\trailerFontSize`, the default is `\footnotesize`. This command must be re-defined in the usual way: `\renewcommand{\trailerFontSize}{\scriptsize}`, for example.

Figure 2 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 30.

- 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 38.

- Next comes the value of `title`, its color is controlled using the `titleColor` key of `\selectColors`.

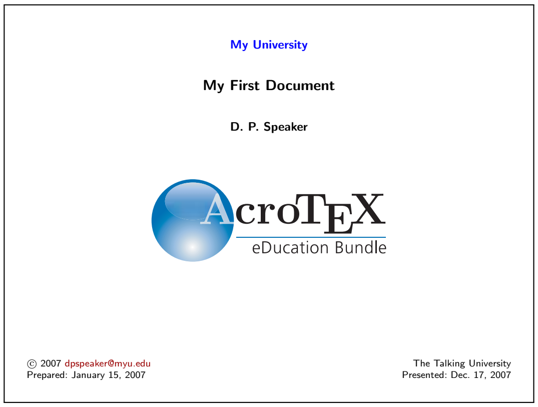
- The author follows, which is the value of `author`. The color is set by the `authorColor` key of `\selectColors`.

- 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 `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 2 is the directory, which has been turned off by the `nodirectory` option of Web.



My University

My First Document

D. P. Speaker

© 2007 dpspeaker@myu.edu
Prepared: January 15, 2007

The Talking University
Presented: Dec. 17, 2007

Figure 2: 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 39. 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, permissible 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, permissible values are `tiny`, `scriptsize`, `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 a 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 paramters, for example, `usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}`. The first four are the arguments of the LaTeX’s `\usefont`, encoding, family, series and shape. The last argument are the arguments of the 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 3 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 3 came about as a result of the following commands in the preamble:

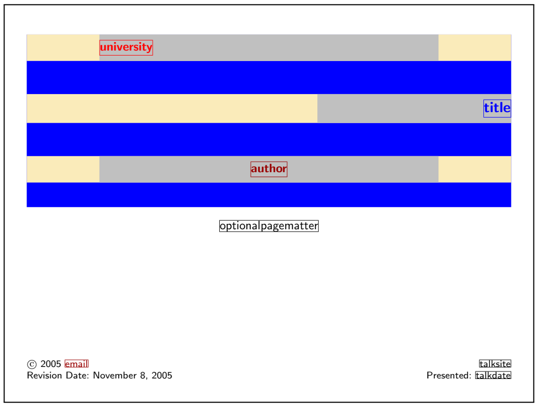


Figure 3: The Title Page Layout Structure

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

In Figure 3, 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 value 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.

11. fontSize: Set the font size of the text, the default is \footnotesize. This key simply redefines \trailerFontSize.

12. formatting: Any additional formatting you wish, such as formatting=\bfseries, or anything that does not cause trouble.

The demo file web_pro illustrates this command.

15.3. 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 39.

Command Location: Place this command in the preamble.

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

When the \noSectionNumbers command is used ('Section Headings' on page 39), 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 for `\selectTocDings`

1. `dDing`: The ding for the first (section) level entry. An example is `dDingToc=\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.

15.4. 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 (see page 30). 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 30). 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 30). 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 15.5, page 39, for a discussion of the commands `\sectionLayout`, `\subsectionLayout` and `\subsubsectionLayout`.

The demo file `web_pro` illustrates this command.

15.5. Section Headings

The Web package uses the section headings \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 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 the section title, permissible 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 a 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 paramters, for example, `usefont={OT1}{cmdh}{m}{n}{{16}{16pt}}`. The first four are the arguments of the LaTeX's `\usefont`, encoding, family, series and shape. The last argument are the arguments of the LaTeX'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,
    halign=c,
    color=red,
    shadowcolor=blue
}
```

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

```
\shadowhoffset{<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 `\shadowhoffset{.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 `halign` 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.

The Exerquiz Package

16. Overview

The exerquiz package provides environments for creating the following interactive elements in a pdf document.

- 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.

16.1. 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.

16.2. 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.

17. 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 26; they are also discussed below.

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

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.

17.1. 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[4], a screen design package written for pdftex by C. V. Radhakrishnan, has been tested and works correctly with exerquiz. For example,

```
\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`.

[4] CTAN:macros/latex/contrib/supported/pdfscreen

17.2. 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 (the material described in Section 18) is supported by these two options. None of the quiz environments can be used with these two options at this time. Y&Y users need to use the dvipsone option if the quiz environment is needed.

17.3. The Language Option

The language option, available in the web package, can be invoked even when the web package is not used.[5] Currently, dutch, french, german, italian, norsk, russian, spanish, polish, finnish, czech, catalan and brazil 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.

17.4. The forpaper and forcolorpaper Options

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

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

See the discussion of the forpaper on page 24 given earlier.

17.5. 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.

[5] Otherwise, the language option is introduced as an option of the web package.

17.6. 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.

17.7. The exercisesonly 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.

17.8. 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. 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.

An exercise with multiple parts can also be defined, with hypertext links to the solutions of the individual parts of the exercise.

The exercise environment has its own counter (eqexno), but there is an option for using another counter—or no counter at all. This may be useful for creating a numbered example environment.

There is an option for placing the solutions immediately after the statement of the problem. This, again, may be useful for an example environment where you want the solution to the example to follow the statement, rather than being hypertext-linked to the solution.

Finally, there is an option for hiding solutions, in the following sense: When the hidden option is used, the solutions are commented out rather then being written to the \jobname.sol file. Additionally, there is a global option, nohiddensolutions; in this case, when you re-LaTeX, the solutions are written to \jobname.sol, and input back into the document.

18.1. Basic Usage

The syntax for the exercise and solution environments is as follows:

```
\begin{exercise}
Your Question.
\begin{solution}
The Solution to Your Question
. . . . . . . . . . . . . . .
. . . . . . . . . . . . . . .
. . . . . . . . . . . . . . .
\end{solution}
\end{exercise}
```

Here is an example of the usage.

Exercise Evaluate the integral $\displaystyle\int x^2 e^{2x}\,dx$.

The code for this is

```
\begin{exercise}\label{ex:int}%
Evaluate the integral \(\displaystyle\int x^2 e^{2x}\,dx\).
\begin{solution}
We evaluate by \texttt{integration by parts}:
\begin{alignat*}{2}
 \int x^2 e^{2x}\,dx &
   = \frac12 x^2 e^{2x} - \int x e^{2x}\,dx &&\quad
           \text{$u=x^2$, $dv=e^{2x}\,dx$}\\&
... lines removed ...
   = \frac14(2x^2-2x+1)e^{2x} &&\quad
           \text{simplify!}
\end{alignat*}
\end{solution}
\end{exercise}
```

See the demo file webeqtst.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 the macro `\includeexersolutions`.

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

```
\renewcommand{\theeqexno}{\thesection.\arabic{eqexno}}
```

can be placed in the preamble of your document. In this case, the above exercise would appear as `Exercise 18.1`.

▶ The usual cross-referencing mechanisms for 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 46.

```
    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]{\textsc{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 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 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 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.[6]

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 the file webeqtst.tex that comes with this distribution.

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 multiparts 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) $1/z$

The syntax is the same as an exercise with multiparts.

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

[6] 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.

```
&
\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 51).

▶ The sample file webeqtst.tex contains this particular example.

18.2. 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. (If solutionsafter and nosolutions both appear in the option list, solutionsafter overrides nosolutions.)

▶ 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).

• 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-LaTeXing, 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 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 from the document under construction.

• 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[dvipsone]{web}  % dvips or pdftex
\usepackage[solutionsafter]{exerquiz}
```

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

Exercise 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

The option solutionsafter is global; all exercises will be 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 typsetting examples. See the example in the section entitled 'Redesigning the exercise Environment' on page 52.

► 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 files webeqtst.pdf and hw02.pdf (and their source files) 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.

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

```
\newcommand\exlabelsol{\exlabel}
```

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

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

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

```
\newcommand\exrtnlabelformat{\exlabelsol\ \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. Normally, the value of this counter figures into the definitions of `\exlabelformat`, `\exsllabelformat` and `\exrtnlabelformat`. 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\exrtnlabelformat{$\blacktriangleleft$}
\renewcommand\exsecrunhead{\exsectitle}
\begin{exercise}}%
{\end{exercise}}
```

See any standard LaTeX 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 must be handled with special care, they are \exsllabelformat and \exrtnlabelformat; 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\exrtnlabelformat{$\square$}
\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. `\exsolafter` defaults to '`\textit{Solution}:`'. You can redefine it as follows:

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

This redefinition yields:

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 \textsc{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.

▶ Finally, see the sample file `hw01.tex` that illustrates how to change all the labels. The file also demonstrates how web and exerquiz can be used to post problems on the Internet, or on paper, with or without solutions included.

19. 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 postphoned to Section 21, entitled Objective Style Questions. The environment allows redefinition to customize the look of your quizzes. (See 'Redesigning the `shortquiz` Environment' on page 63.)

19.1. 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 $\dfrac{d}{dx}\sin(x^3)$?

(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 $\dfrac{d}{dx}{\sin(x^3)}$?
\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 with `p{<width>}` to set up the columns. 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 files webeqtst.tex and qz01.tex for examples. The later file gives examples of how to redefine some of the standard shortquiz labels.

• 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).[7] 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?

□ 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?

[7] In previous versions of exerquiz, there was a *-option that was use to signal the use of radio buttons. This option is still available, but the shortquiz* environment is preferred.

□True □False

Here 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." 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}
\begin{solution}
The answer is ‘Yes’.  The definition requires that
$$
           F’(x) = f(x) \quad\text{for all $x$,}
```

```
$$
well, let's check it out.
.........................
.........................
Therefore,
$$
       F'(x) = 4x^3 = f(x)\quad\text{for all $x$,}
$$
as required by the definition.
\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.

• The `questions` Environment

The `questions` environment was designed to work with the `quiz` environment—taken up in Section 20 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. $\dfrac{3x}{2y^2z^3} - \dfrac{2}{xy^3z^2}$.

(a) LCD $= 2xy^5z^5$ (b) LCD $= 2y^3z^3$

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

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

(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  $\dfrac{3x}{2y^2z^3}-\dfrac2{xy^3 z^2}$.
\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}
```

19.2. 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 63, 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.

By the way, if you want to create a series of multiple choice questions with solutions, you must make up a lot of named destinations (the optional argument of the `answers` environment). Alternately, you can let LaTeX assign the names for you, which provides for you a uniform naming system. You can use `questionno` to do this:

```
\begin{shortquiz} Answer each, then look at solutions.
  \begin{questions}
    \item  ...
      \begin{answers}[quiz:\thequestionno]{4}
        ...
      \end{answers}
      \begin{solution}
        ...
      \end{solution}
    \item  ...
```

```
   \begin{answers}[quiz:\thequestionno]{4}
    ...
   \end{answers}
   \begin{solution}
   ...
   \end{solution}
 \end{questions}
\end{shortquiz}
```

• 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 locally 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 can be redefined, its definition is

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

This option works for the quiz environment defined below (page 65), 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 an the text or the solution.

Using 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 \aebshowgrayletterstrue at some appropriately chosen point in the document; to turn off the gray letter feature insert \aebshowgraylettersfalse.

In support of the showgrayletters option is a new command \REF. \REF acts like the LaTeX 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 a an answer.

D Another alternative.

Reference both (A) and (B).

The code is

```
\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 a an answer.\eAns
    \Ans0 Another alternative.\eAns
\eChoices
\end{answers}
\end{shortquiz*}
\noindent Reference both (\REF{testsqFirst}) and (\REF*{testsqSecond}).
\aebshowgraylettersfalse
```

Notice that the \ifaebshowgrayletters is not turned to false until after the usage of \REF.

• Moving the Solution Set

The solution set, by default, comes last in the file. You can move its positioning by including the command \includequizsolutions 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 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@sqslrtnlabel{End Exam}
% change solutions label
\renewcommand\eq@sqsllabel{%
   \string\textbf{Solution to Exam:}}
\renewcommand\eq@sqslsectitle{Solutions to Exams}
% change default running header for solutions
\renewcommand\eq@qslsecrunhead{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, \sqslrtnlabel, \sqsllabel and \sqslsectitle. Note that you cannot temporarily change \eq@qslsecrunhead, the running label—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@sqsllabel{%
   \string\textbf{Solution to Question \thequestionno:}}
```

► See the sample files webeqtst.tex and qz01.tex for examples. The later file gives examples of how to redefine some of the standard shortquiz labels.

• Modifying Form Elements

For quizzes that use radio buttons (see page 57 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 21.4, The `shortquiz` Environment. There too, methods of modifying the appearance of the form elements are discussed.

20. 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 21 on page 75

There are two types of quizzes, the link-style and form-style. In Section 20.2, we see that the `quiz` environment can also correct the quizzes.

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

$$\texttt{quiz} \supseteq \texttt{questions} \supseteq \texttt{answers}$$

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 74 for more details on this macro.

20.1. Basic Usage

In this section we discuss the two basic `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

Score:

▶ While you are taking the test, and before you click on 'End Quiz', you can change your answers. A message box comes out, gives you your original choice, and asks you whether you really want to change your answer.

```
\begin{quiz}{qzdiscrl}  % 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:[8]

```
\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

[8] 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.

End Quiz

Score:

▶ 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 files `webeqtst.tex` and `qz02.tex` for examples. The later file gives examples of how to customize `quiz`.

• 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, `\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.

• 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
```

Begin Quiz Answer each of the following. Passing is 100%.

1. Who created TeX?

(a) Knuth (b) Lamport (c) Carlisle (d) Rahtz

2. Who originally wrote LaTeX?

(a) Knuth (b) Lamport (c) Carlisle (d) Rahtz

End Quiz Score:

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.

```
\useBeginButton[\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 56), 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 62 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 for the `highThreshold()` function for an example of how to do this.

20.2. 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

Score: Correct

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}{qzdiscr1l} Using the discriminant, $b^2-4ac$,
respond to each of the following questions.
\begin{questions}
............................
............................
............................
\end{questions}
\end{quiz}

\ScoreField{qzdiscr1l}\eqButton{qzdiscr1l}
```

► 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 74 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 Score: Correct

▶ In the partial verbatim listing that follows, notice the field name has been changed from `qz:discr1-l` to `qzdiscr1`. 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.

20.3. 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%.

1. Who created TeX?

□Knuth □Lamport □Carlisle □Rahtz

2. Who originally wrote LaTeX?

□Knuth □Lamport □Carlisle □Rahtz

End Quiz Score: Correct

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 also, the `solution` environment follows, and is not nested within the `answers` environment.

20.4. 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 rg}
    \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 Score: Correct

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 rg}
    \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{color.red}
\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 the button \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 Score: TeX

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 rg}      % 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.

▶ See the file `qz02.tex` for details and examples of how to modify the quiz titles. The language files, e.g., `eqfr.def` and `eqde.def`, demonstrate how to redefine all variables, including those listed above.

20.5. Adding Points to a Quiz

The discussion of this topic can be found in 'Assigning Points' on page 91.

21. 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.

21.1. 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 $\dfrac{d}{dx}\sin^2(x) =$ □

See '`\RespBoxMath`: The Math Question' on page 76 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 79 for details.

21.2. 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 $\dfrac{d}{dx}\sin^2(x) =$ □

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.

You can also pose multivariate questions as well, for example

▶ $\dfrac{\partial}{\partial y}4x^2y^3 =$ □

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 preselected amount, an ϵ value, if you will, the user's answer is declared incorrect; otherwise, it is considered correct.[9]

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 variable. JavaScript Note: In JavaScript, functions such as `sin(x)` and `cos(x)` are methods of the `Math` object. It is not necessary, however, to type `Math.sin(x)` or `Math.cos(x)`; this is done by inserting the expression into a `with(Math)` group. For example,

```
with(Math){ 2*sin(x)*cos(x) }.
```

#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

[9] The idea for evaluating user input in this way comes from Drs. Wlodzimierz Bryc and Stephan Pelikan of The University of Cincinnati.

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 81 of the section below and see the demo file `integer_test.tex` to demonstrate the new method for specifying variables.

#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.

#5 : The number of samples points to be used, usually 3 or 4 is sufficient.

#6 : Precision required, the ϵ value, if you will.

#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.
3. The JavaScript of Acrobat 5.0 does have exception handling, but this has not been incorporated into the code yet. Taking advantage of this new capability will be my next project. Exception handling will give the code protection against user's entering spurious answers. For example, based on the correct answer, the author chooses the interval $[0, 1]$, but the user enters a function whose domain does not contain the interval, such as `(x-1)^(1/2)`.

▶ See the file jquiztst.pdf for various examples of the math questions. The source code is available from the main AcroTEX Bundle Web Site

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 74 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.

#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. The source code is available from AcroTEX Bundle Web Site.

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.

21.3. 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

$\dfrac{d}{dt}\sin^2(t) =$ □ Ans

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

$\dfrac{d}{dt}\sin^2(t) =$ □ Ans

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.

• 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

$\dfrac{d}{dx}\sin^2(x) =$ □ Ans □

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

$\dfrac{d}{dx}\sin^2(x) =$ □ Ans □ Clear

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.

21.4. The `shortquiz` Environment

The objective question (with or without the presence of a correction box, `\corrAnsButton` 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. $\dfrac{d}{dx}\sin^2(x) =$ □ Ans □

3. Name *one* of the two people recognized as a founder of Calculus.

□ Ans □

Clear □

▶ 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}\hfill\sqTallyBox

\item $\displaystyle\frac d{dx} \sin^2(x) =$
\RespBoxMath{2*sin(x)*cos(x)}[sinsqx]{4}{.0001}{0}{1}%
\hfill\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 \emph{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}\hfill
\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 `\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}
```

21.5. 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`.

Since the evaluation of the quiz is delayed until the user has finished the quiz, the `\sqTallyBox` macro is not needed.

Begin Quiz Answer each of the following. Passing is 100%.

1. If f is differentiable, then f is continuous.

□ True □ False

2. $\dfrac{d}{dx}\sin^2(x) =$

3. Name *one* of the two people recognized as a founder of Calculus.

End Quiz Score: Correct

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`.

▶ 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}
```

▶ There are some additional grade reporting and statistical fields defined: `\PointsField`, `\PercentField`, and `\GradeField`. See the demo file `quizpts.tex` for details and examples; see also the section below entitled Assigning Points.

• 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…

Begin (3^{pts}) Compute the following cross product:

$$(3\vec{\imath} - 2\vec{\jmath}) \times (\vec{\imath} + 5\vec{k}) = \square\,\vec{\imath} + \square\,\vec{\jmath} + \square\,\vec{k}$$

End $\underbrace{\square}_{\text{ScoreField}}$ $\underbrace{\square}_{\text{PointsField}}$ Correct 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.

21.6. 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	
`\Ans`	`\everysqRadioButton`
`\sqTallyBox`	`\everysqTallyBox`
`\sqTallyTotal`	`\everysqTallyTotal`
`\sqClearButton`	`\everysqClearButton`

Global Modification Commands	
For the quiz Environment	
\useBeginQuizButton	\everyBeginQuizButton
\useEndQuizButton	\everyEndQuizButton
\Ans	\everyqRadioButton
\ScoreField	\everyScoreField
\eqButton	\everyeqButton
\AnswerField	\everyAnswerField
\PointsField	\everyPointsField
\PercentField	\everyPercentField
\GradeField	\everyGradeField
For Both Environments	
\RespBoxMath	\everyRespBoxMath
\RespBoxTxt	\everyRespBoxTxt
\CorrAnsButton	\everyCorrAnsButton

☛ In additional 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. 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.

22.1. 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.

▶ **Equation handling.** See the sample file jqzequat.tex for examples of posing and evaluating questions that expect an equation as the response. Below is a portion of the preamble of that file; basically, to use one or more of the JavaScripts in the JavaScript library, you specify that option \usepackage command for dljslib pacakge. In this case, we want to process equations so type…

```
\documentclass{article}
\usepackage{amsmath,amscd}
\usepackage[tight,pdftex,designi,nodirectory]{web}
\usepackage{exerquiz}
\usepackage[equations]{dljslib}  % <--choose equations
```

▶ **Vector Handling** There are also JavaScript functions for processing vector answers. See the sample file jqzspec.tex. Actually this file does not use the JavaScript library, but is more of a tutorial on how to use \insdljs to write custom JavaScripts to process exerquiz math fill-in questions.

The preamble of that document could actually be replaced with…

```
\documentclass{article}
\usepackage{amsmath,amscd}
\usepackage[tight,dvipdfm,designi,nodirectory]{web}
\usepackage{exerquiz}
\usepackage[vectors,indefIntegral]{dljslib}
```

The vectors option specifies JavaScripts for processing vector questions. The indefIntegral option is also specified. This is because that in the file jqzspec.tex a comparison function is developed for properly evaluating questions in which an indefinite integral is expected.

▶ **Comma Delimited Answers and Sets** The setSupport option of the dljslib Package provides basic support for math fill-in questions requiring a (unordered) list of comma delimited numerical or symbolic answers, or for a set. The demo and tutorial file is set_test.tex. See that file for more details.

▶ In addition to the two above mentioned sets of JavaScripts there are a couple of comparison functions, one for processing indefinite integrals (see dljs_ex.tex), and the other for using relative absolute error rather than absolute error. Again, see the documentation of dljslib.dtx and the sample file jqzspec.tex.

22.2. 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.

See the documentation that accompanies the package (by latexing `insdljs.dtx`) for details of how to use the library. Also, see the sample file `insdljs_ex.tex` for examples that do not use exerquiz, and the file `jqzspec.tex`, for examples that do use exerquiz.

23. 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 Section 24.

23.1. 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 DLJS 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[10] for this purpose. For example,

```
\newcommand\FirstName[2]{\textField
    [\DV{First Name}\textFont{TiRo}\textSize{10}\textColor{0 0 1 rg}]
    {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 AcroTEX 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:

[10] You can also use hyperref's \TextField command for this purpose as well.

```
\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`, but is known 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.

23.2. Features *apropos* to Submitting

• Assigning Points

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%.

1. (4^{pts}) If $\lim_{x\to a} f(x) = f(a)$, then we say that f is...

   □ differentiable □ continuous □ integrable

2. (6^{pts}) Name *one* of the two people recognized as a founder of Calculus.

End Quiz Score: Correct

Answers:

Points: Percent:

► See the sample file quizpts.tex for a more elaborate version of this question, as well as the source code.

1. `\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.
```

2. `\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{($\eqPTs^{\text{pts}}$)}
```

3. 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`.

4. `\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]
}
```

- \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".

24. 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 (the absolute URL to this document is eq2dbman.pdf), the documentation for the distribution. eq2db is available for download at the AcroTeX home page:

http://www.math.uakron.edu/~dpstory/webeq.html

25. 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.

26. List of Options

Options of the Web/Exerquiz Packages	
Options of the Web Package	
`dvipsone`	dvi-to-ps driver by Y&Y, Inc.
`dvips`	dvi-to-ps driver
`pdftex`	tex-to-pdf application
`dvipdfm`	dvi-to-pdf application
`textures`	the Textures System for Mac
`designi`, `designii`, `designiii`, `designiv`, `designv`	these set screen design parameters
`usetemplates`	this option activates mechanism for creating colored backgrounds and overlays
`leftpanel`	creates a left navigational panel
`rightpanel`	creates a right navigational panel
`navibar`	inserts a menu bar at the bottom or each page
`latextoc`	displays the standard toc
`nodirectory`	eliminates the directory listing on the title page
`forpaper`	this turns off color, and does not put solutions on separate pages.
`forcolorpaper`	Same as `forpaper`, but does not turn off color, useful for color printers.
`latexlayout`	`web` uses page layout for `article` class. For use with `forpaper`.
`tight`	redefines list environment parameters so lists don't take up so much space
`pro`	causes web to input enhanced control over various web elements
`nobullets`	forces the use of numbers for subsubsections, also executes the `latextoc` option
`usesf`	switches to sans serif as the default font, good for presentations
`dutch`	Dutch for web, passed to exerquiz. (Thanks to Henny Wilbrink)
`french`	French for web, passed to exerquiz. (Thanks to Jean-Michel Sarlat)
`german`	German for web, passed to exerquiz. (Thanks to Michael Wiedmann)

Options of the Web/Exerquiz Packages (cont.)	
`italian`	Italian for web, passed to exerquiz. (Thanks to PierLuigi Zezza)
`norsk`	Norwegian for web, passed to exerquiz. (Thanks to Hans Fredrik Nordhaug)
`russian`	Russian for web, passed to exerquiz. (Thanks to Sergei V. Znamenskii)
`spanish`	Spanish for web, passed to exerquiz. (Thanks to Pedro Luis Luque)
`polish`	Polish for web, passed to exerquiz. (Thanks to Jerzy Mycielski)
`finnish`	Finnish for web, passed to exerquiz. (Thanks to Päivi Porras)
`catalan`	Catalan for web, passed to exerquiz. (Thanks to Ramon Ballester)
`czech`	Czech for web, passed to exerquiz. (Thanks to Robert Marik)
`brazil`	Brazilian Portuguese for web, passed to exerquiz. (Thanks to Koichi Sameshima)
Options of the Exerquiz Package	
`pdftex`	tex-to-pdf application
`dvipdfm`	dvi-to-pdf application
`nosolutions`	removes the solutions to the exercises
`noquizsolutions`	removes the solutions to the quizzes
`nohiddensolutions`	overrides the '`h`' (hidden) option for the exercises.
`noHiddensolutions`	overrides the '`h`' and '`H`' (hidden) options for the exercises.
`nocorrections`	removes the ability to correct the quizzes
`solutionsafter`	solutions to exercises are typeset just after the question
`forpaper`	same function as in web. Needed when exerquiz is not used with web
`forcolorpaper`	same function as in web. Needed when exerquiz is not used with web
`preview`	shows the outline of all form fields in the dvi previewer
`nodljs`	turns off the insertion of DLJS
`exercisesonly`	if document has only exercises, no doc level JS needed

Options of the Web/Exerquiz Packages (cont.)	
`debug`	this option is passed on to the `insdljs` package
`proofing`	mark the correct answers for `shortquiz` & `quiz` for proof reading.
`dutch`	JavaScript messages in Dutch (Thanks to Henny Wilbrink)
`french`	JavaScript messages in French (Thanks to Jean-Michel Sarlat)
`german`	JavaScript messages in German (Thanks to Michael Wiedmann)
`italian`	JavaScript messages in Italian (Thanks to PierLuigi Zezza)
`norsk`	JavaScript messages in Noregian (Thanks to Hans Fredrik Nordhaug)
`russian`	JavaScript messages in Russian (Thanks to Sergei V. Znamenskii)
`spanish`	JavaScript messages in Spanish (Thanks to Pedro Luis Luque)
`polish`	JavaScript messages in Polish (Thanks to Jerzy Mycielski)
`finnish`	JavaScript messages in Finnish. (Thanks to Päivi Porras)
`catalan`	JavaScript messages in Catalan. (Thanks to Ramon Ballester)
`czech`	JavaScript messages in Czech. (Thanks to Robert Marik)
`brazil`	JavaScript messages in Brazilian Portuguese. (Thanks to Koichi Sameshima)

What's New

The following is a brief enumeration of some of the major new features of the web and exerquiz packages.

- **Web**

1. (Version 5.0) The pro option with many user-convenient features, see 'The pro Option' on page 29.
2. (Version 5.0) New Web options of `usesf`, page 13 and `centertitlepage`, page 21.

- **Exerquiz**

No major changes to exerquiz, some minor bug fixes. See the file `eqchange.txt` for more details on the change history.

- **insdljs**

The `insdljs` Package is a general purpose LaTeX package for inserting Acrobat JavaScript into the document level section of a PDF document. The package features the `insDLJS` environment. This environment typically goes in the preamble of a LaTeX source file, or in the style files of a LaTeX package. See the documentation contained within the `insdljs.dtx` file for additional details. There is a sample file, `insdljs_ex.tex` that can be used as a startup test file.

1. (Version 2.0f) Added a command `\previewMiKTeX` which redefines `\jsR` and `\jsT` that enables, I hope, the yap previewer to preview an AeB document.

- **dljslib**

The `dljslib` Package acts as a library of JavaScript functions. Due to the increased programmability of `exerquiz` and its new-found flexibility, it is possible to write a number of different routines to handle various kinds of math fill-in questions. These JavaScript functions can be stored in the library and retrieved when needed. This package requires the `insdljs` package.

1. There are some major user-contributed JavaScript routines. The following functions, which are `ProcResp`-types, were written for an online grading system being developed by Drs. Bruce Wagner and David Arnold, and Mr. Jacob Miles-Prystowsky. The descriptions given below were provided by the authors. I want to extend my deep thanks for their contribution! See the documentation of dljslib.dtx for extended details.
   (a) The JS function `ProcRespSetFormula` grades an unordered list of formulas, such as `x`, `x^2`, `x^3`.
   (b) `ProcRespFactors` is for grading polynomial factoring questions. If a polynomial factors as `-5x^2(x-4)(x+2)`, the answer is only unique up to the order of factors and a change of sign on an even number of factors. For example, it could also be written as `-5x^2(x+2)(x-4)` or `5x^2(-x+4)(x+2)` or `-5(x-4)(x+2)x^2`, etc. This script should grade all of these correctly.
   (c) The JS function `ProcPoint` is almost an exact copy of the JS function `ProcVec`, but uses parentheses instead of angle brackets as delimiters. This is for questions that have a point (x, y) as the answer.

   (d) ProcRespIntervals is for grading questions that have an interval or union of intervals as the answer.

   The sample file for this contribution is dljslib_apw.tex.

2. Added a feature whereby you can create a file libcusopt.opt to declare your own options for this package. These declarations should be a combination of existing options. For example,

   ```
   \DeclareOption{procrespgrp}{
       \dljsRegister[y]{vectors}\dljsRegister[y]{setSupport}
       \dljsRegister[y]{equations}\dljsRegister[y]{intervals}
       \dljsRegister[y]{unordered}\dljsRegister[y]{point}
       \dljsRegister[y]{factors}\dljsRegister[y]{ImplMulti}
   }
   ```

   creates a new option for dljslib called procrespgrp which bundles together all frequently used JS functions frequently used. 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.

   A sample libcusopt.opt is distribued with the AeB, this sample file consists of the example given above, and is used by the sample file dljslib_apw.tex.

   ——————— Older New Stuff ———————

3. Added an option limitArith to the dljslib, which limits the use of arithmetic operations and built-in functions. The demo file for this is limarith.tex.
4. Added setSupport option to dljslib. This gives support to math fill-in questions for processing (1) a set of numerical answers or a comma delimited list of answers; (2) a symbolic set of answers. The demo file is set_test.tex.
5. Added combinatorics, nodec and noBinFac to the JavaScript library. These additions are due to Ross Moore and Frances Griffin and were developed for their MacQTEX online testing system. Some of the changes listed in the exerquiz section above were to accommodate these new functions (for example, having integer variables). Many, many thanks for generously sharing their JavaScript with the TEX/PDF community. The test file for these features is integer_tst.tex.

Now, I really must get back to retirement. DPS

Solutions to Exercises

Exercise 1. We evaluate by `integration by parts`:

$$\begin{aligned}
\int x^2 e^{2x}\,dx &= \frac{1}{2}x^2e^{2x} - \int xe^{2x}\,dx && u = x^2,\, dv = e^{2x}\,dx\\
&= \frac{1}{2}x^2e^{2x} - \left[\frac{1}{2}xe^{2x} - \int \frac{1}{2}e^{2x}\,dx\right] && \text{integration by parts}\\
&= \frac{1}{2}x^2e^{2x} - \frac{1}{2}xe^{2x} + \frac{1}{2}\int e^{2x}\,dx && u = x^2,\, dv = e^{2x}\,dx\\
&= \frac{1}{2}x^2e^{2x} - \frac{1}{2}xe^{2x} + \frac{1}{4}e^{2x} && \text{integration by parts}\\
&= \frac{1}{4}(2x^2 - 2x + 1)e^{2x} && \text{simplify!}
\end{aligned}$$

Exercise 1

Exercise 2.

$$\boxed{x + y = 1}$$

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 $\boxed{v = 2t - 5}$. □

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: $\boxed{a = 2}$. □

Exercise 4(a) $i^2 = -1$ □

Exercise 4(b) $i^3 = ii^2 = -i$ □

Exercise 4(c) $z + \bar{z} = \operatorname{Re} z$ □

Exercise 4(d) $\dfrac{1}{z} = \dfrac{1}{z}\dfrac{\bar{z}}{\bar{z}} = \dfrac{z}{z\bar{z}} = \dfrac{z}{|z|^2}$ □

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: Yes, Donald Knuth was the creator of TeX. ■

Solution to Quiz: Yes, Leslie Lamport was the creator of LaTeX. ■

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 Refernce, Technical Note #5431, Version 6.0., Adobe Systems, Inc., 200
http://partners.adobe.com/asn/acrobat/docs.jsp 89

[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. 53

[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. 16

[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

[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 12