AcroTEX.Net

JJ_Game Class

D. P. Story

Prepared: December 15, 2016

www.acrotex.net
Version v3.1, 2016/11/24

Table of Contents

1. Introduction

The "JavaScript Jeopardy Game", a.k.a., jj_game (http://ctan.org/pkg/jj-game), is a LaTeX class file that can be used to construct a Jeopardy-like game board and the accompanying answers and questions. The jj_game is capable of constructing a game board for an arbitrary number of categories and an arbitrary number of questions per category.

There are options for users of dvipsone (from the now defunct Y&Y), dvips, pdflatex, lualatex, or xelatex to build a Jeopardy game.

A Jeopardy game, as constructed by jj_game, consists of three parts: the instructions, the game board and the question pages. See figures 1, 2 and 3 below.

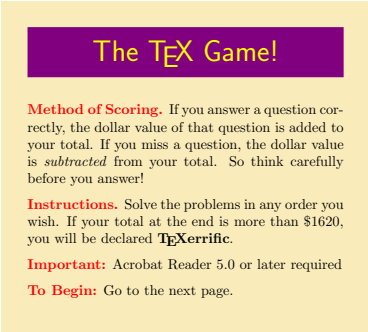

Figure 1: Instructions Page

Figure 2: Game Board Page

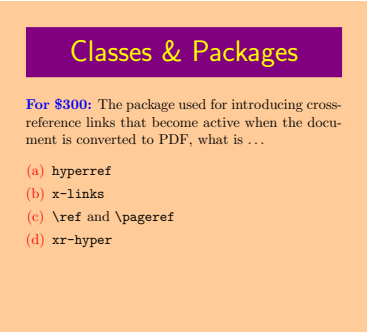

Figure 3: Question Page

Corresponding to each of these pages are environments and commands that allow you create content for your Jeopardy quiz game and to designate the color scheme to be used.

After reading your detailed instructions, the contestant arrives at the game board page. The contestant chooses a category and a question in that category ("I'll take 'Candles and Wicks' for 200"). When the contestant clicks on the cell, the game jumps to the question page ("Which end of the candle do you light? What is..."[1]) corresponding to that choice and a question is posed.

A question can be multiple choice, math fill-in or text fill-in. In the case of the fill-in type questions, the exerquiz package is needed. After the contestant answers the question, the game returns to the game board page, and the score is updated.

Should the contestant reach the goal set by the Goal property of the \GameDesign command, when the contestant has finished answering all questions, a hidden field under that banner of the game board appears with a short message, as defined by the Champion property of \GameDesign command.

There is no reset button, so if you want to play again, just close the game, and reopen it. Do not save the game before it closes, this will save the form data and when you open it again, the initial state of the game will not be restored.

You should be aware of the following features, which are meant to discourage contestants from looking at the questions by paging through the PDF or attempting to edit earlier responses.

[1] The author is making a vague and rather poor play on words with Wiccan. pooh.

- When a question page is opened, the cell in the game board that corresponds to that page is blanked out. When the contestant returns to the game board, that cell is no longer clickable, hence, the contestant cannot legally answer that question. When a contestant is on a question page illegally, a nasty message appears in an alert box if the contestant answers the question on that page.
- If the contestant answers the question legally, then later returns to that same question page and tries to edit the question (perhaps because it was missed earlier), a (less than nasty) message appears saying that you cannot change your answer.

I do hope there will be people out in Cyberland who will use jj_game and construct some interesting games for us to enjoy.

Now, I simply must get back to my retirement. ☺

2. Preliminaries

2.1. Requirements

The new jj_game class now uses many of the standard package files from AeB (AcroTEX eDucation Bundle); in particular, the AeB components used are web, insdljs and eforms. The exerquiz package can optionally be used as well to pose math and text fill-in questions. Therefore, AeB should already be installed (http://ctan.org/pkg/acrotex).

The jj_game class usse the very fine package xkeyval (2006/11/18 v2.5f or later),[2], and the equally nice package xcolor (2005/11/25 v2.08 or later),[3] if available on the system. The comment package is also used.[4] The graphicx package is needed if you wish to insert graphical backgrounds.

Requirements based on driver applications:

- pdflatex, lualatex, and xelatex: jj_game class works with these two applications that that write a document in PDF. Some of the features—the forcredit and pro options—of jj_game are not available to these applications.
- dvips and dvipsone: In these two cases, it is assumed that PDF is creating Acrobat Distiller, not Ghostscript; in this case, the forcredit option (see 'The forcredit Option', page 15) is available. If the document author has Acrobat 7.0 Pro (or later), the pro option (see 'The pro Option', page 15) can be used as well.

Once the game is built, Adobe Reader 5.0 or later is sufficient to enjoy the wonderfulness of the time-consuming pass time that is Jeopardy. If the document author used the pro option, Adobe Reader 7.0 or later is required.

[2] Located at CTAN:/tex-archive/macros/latex/contrib/xkeyval
[3] CTAN:/tex-archive/macros/latex/contrib/xcolor
[4] CTAN: /tex-archive/macros/latex/contrib/comment

2.2. Listing of Sample Files

The sample files are good templates for designing your own game:

- `jjg_test.tex`: The original demo file from the first version of this class.
- `jg_ca.tex`: A Jeopardy quiz game constructed for and distributed to a College Algebra class for extra credit.
- `jjg_custom.tex`: An example of a custom design.
- `pro/jjg_pro_ca.tex`: Same as `jg_ca.tex` but uses the `aeb_pro` package (with layers) with the `forcredit` option of `jj_game`. (dvips/Distiller workflow required)

The first three files can be successfully build with dvips/Distiller, pdflatex, lualatex, and xelatex.

3. Options

The options of this class are listed below. In some cases, you are referred to other sections of this manual for further details.

- Driver Options: `jj_game` recognizes four drivers. Choose only one.
  - `dvips`: The dvi-to-ps converter that comes with many TEX systems. The Adobe Acrobat Distiller must be used to convert from Postscript to PDF.
  - `dvipsone`: The dvi-to-ps converter by Y&Y.[5] In this case, the Adobe Acrobat Distiller must be used to convert from Postscript to PDF.
  - `pdftex`: The tex-to-pdf converter. The distiller is not needed in this case.
  - `luatex`: The tex-to-pdf converter. The distiller is not needed in this case.
  - `xetex`: The tex-to-pdf converter. The distiller is not needed in this case.
  - `dvipdfm` and `dvipdfmx`: Less used dvi-to-pdf converters. The distiller is not needed in this case.

  In the cases of `pdftex`, `luatex`, and `xetex`, `jj_game` performs automatic driver detection so none of these need to be actually specified in the option list of `jj_game`. If you use the `dvips` driver, specify it in the `jj_game.cfg` file, as explained on the next page.
- `debug`: In the developmental stage, you can use the `debug` option that will give you more information in the log about what is going on.
- `double`: The face values of all cells are doubled. (Double JavaScript Jeopardy!) Use it if you have time, energy and motivation to write two Jeopardy games.
- `final`: After you have completed your `jj_game`, you can, optionally, build it one more time with the `final` option. This option removes the menu, the toolbar, the GUI interface of the Acrobat Viewer; all that is left is your game.

[5] Now out of business, but this author still uses it.

- `pro`: The `pro` option is available only with the `dvips` or `dvipsone` driver. You are required to use the AeB Pro package. This package requires that you use Acrobat 7.0 Pro (or later) and to create you PDF using Acrobat Distiller. For the pro option, the questions are placed in OCGs (optional content groups, or simply layers). See 'The `pro` Option' on page 15.
- `forcredit`: This option brings in some security measures that helps the instructor set up a Jeopardy game that can be used for (extra) credit in the classroom. See the section titled The `forcredit` Option on page 15.
- `design=`⟨*design_name*⟩: There are a limited number of predefined designs. The values of the design parameter are `jeopardy`, `florida`, `iceland`, `hornet`, `qatar`, `norway`, `bahamas`, and `spain`. These are in addition to the default color design scheme.
- `nodesigngraphics`: Some of the designs specified by the `design` option have graphical backgrounds. To use a specific design without the designed graphical background, use the `nodesigngraphics`. In the preamble, you can optionally specify your own graphical background using the graphical background insert commands, see 'Graphical Backgrounds' on page 18.
- `lang=`⟨`english|german`⟩: The Jeopardy game has some language strings, most notably, the semi-humorous messages that appear after the player responds to one of the questions. The `lang` option allows these common strings to appear in `german`. The default is `english`. When the `german` option is in effect, a crude euro symbol is typeset instead of the dollar, and the euro symbol is used in the score board as well.
- `allowpeeking`: The normal behavior of the game is that if the contestant looks ahead at a question, then the tile on the game board corresponding to the question viewed is hidden. The contestant is not allowed to answer that question. Use this switch to remove this feature of the game.
- `twoplayer`: At the request of a user, I've created a `twoplayer` option; useful for classroom participation. See 'Two Player Jeopardy!' on page 16 for details.

An example of usage of some of these options is

```
\documentclass[dvips,forcredit,lang=german,
    design=florida]{jj_game}
```

You can also set your favorite driver as the default. Edit the file `jj_game.cfg` that comes with the distribution and place in it, for example, the command

```
\ExecuteOptions{dvips}
```

(This sets `dvips` as the default driver.) In this case the above example becomes

```
\documentclass[forcredit,lang=german,design=florida]{jj_game}
```

Specifying a driver as an optional parameter in the `\documentclass` declaration will override whatever default set in the `jj_game.cfg` file.

4. Designing the Game and Posing the Questions

The following sections outline the design of the game. If you don't specify a particular part of the design of the game, then a default value will be substituted.

The sample file `jj_test.tex` represents a prototype game that you can modify.

4.1. Stuff for the Preamble

There is potentially a lot of stuff that goes into the design of the game. Again, all have default values, but there are some required design parameters.

• Banners and Backgrounds

Throughout the document, there are many colors living in LaTeX space that are under the control of the document author. Most all colors can be set through the argument of the `\DeclareColors` command.

- `\DeclareColors`: The argument of `\DeclareColors` consist of key-value pairs. These key-values are in the JavaScript style, key and value are separated by '`: `', the space is required to follow the colon. The key references some color aspect of the game, the value is named color. A color of `transparent` is also recognized. The following is a `\DeclareColors` structure listing all possible keys, with some named color values.

```
\DeclareColors
{
    fillCells: lightgray,       % background for cells
    fillBanner: PineGreen,      % color of banner
    textBanner: Yellow,         % color of text in banner
    textBoard: Yellow,          % color of text in categories on game board
    fillInstructions: cornsilk, % background of the instructions page(s)
    fillGameBoard: cornsilk,    % background of the game board page
    fillQuestions: cornsilk,    % background of the questions pages
    dollarColor: blue,          % color of the heading of questions page
    linkColor: red,             % color of any hypertext links used
}
```

If `\DeclareColors` is not present in the preamble, or if present, if the keys are not fully populated, the default values are used.

See Section 9, beginning page 19, for a visual representation of these colors.

- `\titleBanner`: On the instructions page, see Figure 1, page 3, there is a banner heading. The text of this banner is set by `\titleBanner`

`\titleBanner{⟨Banner Text⟩}`

This command is required, though if not set, the default text will appear, "You are the Winner!". For example

```
\titleBanner{The \TeX\ Game!}
```

You can format the banner text using `\bannerTextFont` by redefining it. Its default definition is

```
\newcommand{\bannerTextFont}{\sffamily\huge}
```

There are other banners in the Jeopardy game, the ones on the questions pages. The text for these banners are the categories of question you declare in the \GameDesign command, discussed in the next section. The \bannerTextFont effects these banners as well.

Finally, there is \bannerTextControl used for finer positioning of the banner text. The default definition is

```
\newcommand{\bannerTextControl}[1]{#1}
```

This can be redefined for whatever purpose. Normally it is not needed. But, for example

```
\renewcommand{\bannerTextControl}[1]{\raisebox{3pt}{#1}}
```

The above example raises the banner text 3pt.

• Game Design

▶ \GameDesign: In order to design a game, you must have categories and questions in each category. The \GameDesign is the macro that is used for this purpose.

```
\GameDesign
{
  Cat: \TeX,                        % list the categories
  Cat: \LaTeX,                      %       "
  Cat: Classes \&~Packages,         %       "
  NumQuestions: 3,                  % number questions per category
  Goal: {3,500},                    % set a goal, just for fun
  GoalPercentage: 0,                % goal as a percentage of money
  CellWidth: 1in,                   % cell width
  CellHeight: .5in,                 % cell height
  ExtraWidth: 0pt,                  % additional width if needed
  ExtraHeight: .2in,                % additional height if needed
  Champion: You are a Champion!     % this text appears if goal is met
}
```

The key-value pairs can appear in any order.

Required Keys: The Cat and NumQuestions keys are required, though this is not enforced.

Format for key-value paires. The format is '⟨*key*⟩: ⟨*value*⟩'. Multiple key-values are separated by a comma (,). If the value contains a comma, enclose the value in braces, as I did above with the Goal. The last item can have an optional comma (,).

Each key, except Cat, has a default value. The values of these keys may need to be adjusted depending on the number of categories, number of questions, and font size.

☛ The value of Cat has an optional argument, you can say

```
Cat: [\small] Classes \&~Packages,
```

The value of the optional argument is inserted just before the "Classes & Packages" in the category heading of the game board. In this case, its size is reduced to fit into the bounding rectangle.

☛ The parmeter `GoalPercentage` needs comment. You can set the goal by setting the percentage of the total money in the game; i.e., `GoalPercentage: 90`, sets the goal to 90% of the total money. If `GoalPercentage` is present in the `\GameDesign`, and its value differs from the default value of zero (0), then the value of the `Goal` parameter, if present, is discarded, and a new goal is calculated based on the percentage of the total money, which is a function of the number of categories, the number of questions per category, and whether the `double` option has been invoked.

The `jj_game` class takes these entries and computes the values for `\paperheight`, `\paperwidth` and `\textheight`.

I've used some code from Radhakrishnan C V's `pdfscreen`. In that regard, you can, however, use `\marginsize` to set the margins around the page, the default is

```
\marginsize{.25in}{.25in}{.25in}{.25in}
```

We now turn to defining the attributes of the Acrobat form fields that appear in the game.

In the next few paragraphs, covering the appearance attributes of the fields corresponding to `\APHidden`, `\APDollar`, `\APRight`, `\APWrong` and `\APScore`, colors are specified in the RGB color space; for example `0.98 0.92 0.73`. Note that unlike the color package of LaTeX, the numbers are separated by spaces not commas. Also, the keys that begin with the word "Fill" (e.g., `FillColor`) recognize the word `transparent` as a value, in this case, there is no background color for that field.

▶ There is a hidden text field in the banner above the game board that is revealed when the game is over (all questions are answered) and the goal, set in `\GameDesign`, is met. The `\APHidden` controls the appearance of this hidden banner.

```
\APHidden
{
  Champion: You are a Champion!,    % this text appears if goal met
  Font: TiRo,                       % font to be used
  Size: 20,                         % size of the font
  TextColor: 0 0 1,                 % color of the text
  BorderColor: 0 0 0,               % border color
  FillColor: 0.98 0.92 0.73,        % background color
}
```

The Champion key can be placed in the `\GameDesign` structure; this is more convenient for creating pre-packaged designs (such as `design=jeopardy` and `design=florida`).

▶ `\APDollar`: The `\DesignGame` just sets basic parameters, it does nothing towards defining color. For that, I have the `\APDollar` macro. This macro sets the attributes of the numbers that appear as the game cells on the game board.

```
\APDollar
{
```

```
    Font: TiRo,           % font to be use for the numbers
    Size: 20,             % size of font
    TextColor: 0 0 1,     % color for the numbers
    BorderColor: 0 0 0,   % color of border
    FillColor: 1 .35 1,   % the fill color for the cell face
  }
```

The above are all the defaults. The `\APDollar` macro does not have to appear, unless you want to change one of the defaults.

- `\APRight` and `\APWrong`: Under the face of the cells are two hidden text fields, one for a "Right" answer, one for a "Wrong" answer. The two macros `\APRight` and `\APWrong` have the same key-value arguments.

```
\APRight{
  Font: TiRo, Size: 20, TextColor: 0 0 1, Message: Right!}

\APWrong{
  Font: TiRo, Size: 20, TextColor: 1 0 0, Message: Wrong!}
```

The default value for the `Message` parameter is "`Right!`" ("`Wrong!`"), but these can be changed with `\correctText` (`\incorrectText`). For German, for example, you can put in the preamble `\correctText{Richtig!}` (`\incorrectText{Falsch!}`), this command lends itself to customization.

- `\APScore`: There is a text field, set by the command `\ScoreBoard`, to keep track of the score. The following macro designs the appearance attributes for the score board.

```
\APScore{
  Font: TiRo,                   % font to be used
  Size: 20,                     % font size to be used
  TextColor: 0 0 1,             % text color
  BorderColor: 0 0 0,           % border color
  FillColor: 0.98 0.92 0.73,    % background color
  CellHeight: \the\cellHeight,  % default same as cells (usually not specified)
  CellWidth: \the\cellWidth,    % default same as cells (usually not specified)
  AutoPlacement: true,          % auto placement of score, values: true or false
  Score: "",                    % score text
  Currency: "$",                % currency
  align: r,                     % horizonal align: l (left), c (center), r (right)
}
```

☛ To get a currency of a EURO or some other currency symbol, use Unicode; in the case of the EURO, use `Currency: "\string\\u20AC"`. When the `lang` option is set to `german`, the EURO automatically becomes the default currency symbol.

The following is a listing of some of the common currency signs. Keep in mind, that these must be escaped like so `Currency: "\string\\00A5"` (Yen Sign).

Description	Unicode	Symbol	Description	Unicode	Symbol
Dollar Sign	\u0024	$	Lira Sign	\u20A4	
Pound Sign	\u00A3	£	Peseta Sign	\u20A7	
Currency Sign	\u00A4	¤	New Sheqel Sign	\u20AA	
Yen Sign	\u00A5	¥	Dong Sign	\u20AB	
French Franc	\u20A3		Euro Sign	\u20AC	€

The Font key needs to have a value of a font that contains the currency symbol. For example, `TiRo` and `Arial` (the open type font versions) both have these currency symbols.

Some currency symbols, and I don't know which are appended to the end of the number. Use

```
\prependCurrency
\appendCurrency
```

The first, which is the default, prepends the currency sign to the number, whereas, the command `\appendCurrency` appends it to the end of the number. Execute these in the preamble.

☛ The `align` parameter horizontally aligns the contestant's current total in the text field. Permissible values are `l` (for left aligned), `c` (for centered), and `r` (for right aligned, the default).

☛ Then `AutoPlacement` is `true`, the `\ScoreBoard` is centered directly under the game board. If `false`, then you have to provide a placement for the `\ScoreBoard`. In this case, use the macro `\PlaceScoreBoard`:

```
\PlaceScoreBoard{\vfill\hspace{\rulewidth}\hspace{\extraWidth}%
  \medskip\ScoreBoard}
```

This places the score board at the bottom of the page, aligned with the left edge of the game board.

4.2. The Stuff in the Body of the Document

Now that we've finished laying out the design of the game, we are ready to actually pose the questions. This is done in the body of the document, following `\begin{document}`, don't you know. First come the instructions, the contestant can't play the game without instructions!

• The Instructions

The first page of the game consists of the instructions. Place the instructions in the `instructions` environment.

```
\begin{instructions}
⟨instructions go here⟩ % perhaps a graphic as well
...
\end{instructions}
```

The `\end{instructions}` ends the page, and inserts the game board on the next page.

Should you need more than one page of instructions, use `\insertJJTitleBanner`. This command will start a new page and place the title banner at the top of the page.

• The Questions

Following the instructions environment comes the question and multiple choice alternatives. For this, there is a `Category` environment, within which is placed a series of `Question` environments. The number of questions per category must be the same as declared as the value of `NumQuestions` in the `\GameDesign` macro.

The format is as follows:

```
\begin{Category}{⟨category_name⟩}
\begin{Question}[⟨num_of_columns⟩]
⟨Text of the question⟩
\Ans0 ...                % a wrong answer
\Ans1 ...                % the right answer
\Ans0 ...                % another wrong answer
\end{Question}
\end{Category}
..........
..........
```

The number of categories must be the same as was listed in macro `\GameDesign`. The ⟨*category_name*⟩ is required; it is used to send messages to the log about whether you are listing the categories in the correct order. It doesn't have to be exactly the same as was defined in `\GameDesign`.

☛ The macro `\Ans` typesets the answers, it takes one argument. An argument of '0', e.g., `\Ans0`, means that answer is wrong, an argument of '1', e.g., `\Ans1`, means the answer is the correct one.

☛ The default behavior is a column listing of the answers. You can have a tabular format by specifying an optional number in the brackets, ⟨*num_of_columns*⟩.

Beginning with version 3.0 of `jj_game`, you can now pose fill-in questions, in addition to multiple choice questions. Use `\RespBoxMath` for math fill-in questions and `\RespBoxTxt` for text fill-in questions. See the manual `aeb_man.pdf`, the manual of usage for the AcroTEX eDucation Bundle.

Enclose the two commands `\RespBoxMath` and `\RespBoxTxt` in a special environment, `oAnswer`. This environment redefines the actions of these two standard `exerquiz` commands.

```
\begin{Question}
    If $f(x) = 2x^2 + x $, what is $f(2x)$?
    \begin{oAnswer}
        \begin{equation*}
            f(2x)=\RespBoxMath{2*(2x)^2 + 2*x}{4}{.0001}{[1,2]}
        \end{equation*}
    \end{oAnswer}
\end{Question}

\begin{Question}
    He was the first president of the United States? Who was\dots
    \begin{oAnswer}
```

```
        \RespBoxTxt{2}{1}{3}{George Washington}
        {G. Washington}{Geo. Washington}
    \end{oAnswer}
\end{Question}
```

Do not use the optional argument for the `Question` environment, it is not needed for fill-in questions.

- The questions page consist of three parts: (1) The banner that states the category for this question; (2) the currency heading which typically says "For $100:"; and (3) the question.

  There are two other commands that can be used to change the appearance of the questions page.

```
\currencyHeading{⟨currencyHeading⟩}
\aboveCurrencySkip{⟨skip⟩}
```

The `\currencyHeading` heading is a command that can be used to change the text that sets up the question for the page, The default value is

```
\currencyHeading{\bfseries For~\$\theCurrencyAmt:}
```

where `\theCurrencyAmt` is a command that expands to the correct amount (of currency) for this question. Use this command in the preamble to redefine the text. For `lang=german`, this text is automatically redefined.

`\aboveCurrencySkip` adjusts the amount of vertical space between the page banner, and the currency heading. The default is

```
\aboveCurrencySkip{0pt}
```

And that's all there is to it!

5. Building the Game

For people using pdftex or dvipdfm, there is no special processing. Just 'pdflatex-it' or 'dvipdfm-it', and you are ready to play!

For dvipsone and dvips, some additional steps are necessary. The jj_game class uses Document Level JavaScripts to control the game as it progresses.

For users of Acrobat 5.0 or later, the document level JavaScript is automatically inserted when the game is open in Acrobat for the first time following distillation, provided the PDF was saved back into the same folder as the source file. (It is in this folder that TEX has written the DLJS in the form of an FDF file.)

Finally, do a 'Save As…' from Acrobat to save the DLJS that have just been inserted. You are now ready to play!

6. Jeopardy for Credit

This section outlines the support implemented to enable the Jeopardy games produced by jj_game to be offered up for credit in a relatively secure environment.

6.1. AeB No Go Pro

Should you wish to offer a Jeopardy game for class credit or extra credit, you need to include `\contestantName` to allow the "contestant" to enter his/her name.

```
\contestantName{⟨length⟩}{⟨width⟩}
```

where ⟨*length*⟩ and ⟨*width*⟩ are the height and width of the text field. See item ③ of Figure 4, page 19.

Command Location: Place this command within the `instructions` environment.

An example of usage:

```
\textcolor{red}{\textbf{Name:}}
    \underbar{\contestantName{1.5in}{11bp}}
```

In the preamble you can use `\afterGameBoardInsertion` to insert material just below the game board and score board. Whereas this is a general purpose command, you can use it to insert a print button

```
\afterGameBoardInsertion{\medskip\gameboardPrintButton}
```

The command `\gameboardPrintButton` creates three form fields, a print-this-page button, a read only text field with the contestant's name pre-filled (remember, contestant entered her/his name in the instructions page), and a read only text field that documents when the contestant first opened the Jeopardy game. There is an optional argument for `\gameboardPrintButton`, this is a standard argument for changing the appearance of the button (see eForms manual).

The three fields have labels that appear in front of the fields, these labels can be changed using the following commands:

```
\printButtonCaption{⟨caption on button⟩}
\printButtonLabel{⟨button label⟩}
\contestantNameLabel{⟨label for contestant name field⟩}
\timestampLabel{⟨label for time stamp field⟩}
\timeStampFormat{⟨util.printd formatting string⟩}
```

Use `\printButtonCaption` to change the text that appears on the button face, called the button caption.[6] The argument of `\timeStampFormat` is the formatting string of the `util.printd()` method, see the documentation on that method from the *JavaScript for Acrobat API Reference*.

The defaults are, respectively,

```
\printButtonCaption{Print}
\printButtonLabel{Print this page:}
\contestantNameLabel{Student:}
\timestampLabel{Time stamp:}
\timeStampFormat{mm-dd-yy, H:MM:ss.}
```

[6] This label will be part of the button, it is not part of LaTeX space, so no formatting is used here.

All the above commands are available for all the drivers.

Should you wish to offer up a Jeopardy game for class credit, the document needs to be secured. If you have Acrobat, version 5.0 or later, you can give the document a no-printing attribute to prevent the students from printing the document and handing it around. Wait, they need to print the game board and hand it in! So setting the document on no-printing does not help.

How can we print the game board page, yet not allow the students to print the questions? The answer to this question is in 'The `pro` Option' on page 15.

6.2. AeB Pro

The techniques described in this section assume the use of the new AeB Pro (Acro TEX eDucation Bundle Professional). The requirement for AeB Pro is the document author, that's you, must use Acrobat Pro, version 7 or later, and must create PDF using Acrobat Distiller. If you are not one of these types, read no further; however, it's okay if you read on, maybe the AeB Pro and its applications will intrigue you enough to convert to the use of Acrobat Pro and its little brother Distiller.

• The `forcredit` Option

When you take the `forcredit` option and are using AeB Pro,[7] JavaScript code is added to the document that prevents the student (contestant :-) from continuing on to the game board page (or any of the questions page) without first identifying themselves in the text field created by the `\contestantName` command.

This "annoyance factor" guarantees the student has identified himself before playing the game; hence, no excuses when they turn in the game board with the name hand written in.

This is just one more piece of the puzzle to the problem of Jeopardy for credit, but it does not solve the printing problem. See the next section for that.

• The `pro` Option

The `pro` option brings in the use of layers from AeB Pro.[8] Layers are content that are placed in different groups. The groups can be turned on (made visible) or off (made hidden).

The content of each of the question pages is placed in its own layer and made hidden. If you scroll to a question page, the question is not visible, as it is still in a hidden state. When you arrive on the page *legally* via choosing a cell from the game board, the layer for that question is made visible.

The layers are given a "no-print even if visible" attribute,[9] so if the student prints the document, the layers do not print!

Adobe Reader and Acrobat have a Layers Navigation Panel through which layers can be made visible, but `jj_game` through AeB Pro locks the layers. A locked layer means that the document consumer cannot change the state (visible, hidden) of the

[7] If you are not using AeB Pro, this option does nothing.
[8] Adobe refers to layers as optional content groups (OCG).
[9] Yes, layers, and AeB Pro can do that!

layer through the user interface. The state of a locked layer can be changed, however, through JavaScript API, and that is exactly what jj_game does!

When the pro option is used, Adobe Reader 7.0 or later is required to view the layers (some JavaScript methods are used that are first defined in version 7). To prevent people viewing the game who do not have at least version 7, you can use the command \requiresVersion from AeB Pro. In the preamble of your game place

```
\requiresVersion{7}
```

See the documentation of \requiresVersion for further details, and related commands that may be useful. When someone opens the document that does not have at least version 7, either the document is closed immediately and an alert box is issued, or if the document is viewed in a browser, an alert message is displayed, and the user in redirected to another page on the Internet.

This scheme is not air tight, it can be gotten around, but I'm not telling.

7. Two Player Jeopardy!

When the twoplayer option is taken, two score boards are created and the two contestants alternate play. The two contestants are labeled 'Player 1' and 'Player 2', the border of the score board is highlighted to indicate whose turn it is. There is also a check box with a red dot in it to indicated whose turn it is. The game plays the same, except the results of each contestant are tabulated in their respective score board. The rest of this section describes customizations.

In addition to the key-values of \APScore discussed earlier, see Game Design, page 8, the following key-values are specific to the twoplayer option.

```
\APScore {
    tpScaledCellWidth: .5,
    tpHighlightBorderColor: 1 0 0
}
```

When the twoplayer option is used, jj_game multiplies the *value* of CellWidth (as specified in \APScore) by tpScaledCellWidth; if the resultant value is less than the \cellWidth (the width of the cells in the game board), \cellWidth is used for the width of the twoplayer score boards; otherwise, the resultant value is used. The default value is .5.

The key tpHighlightBorderColor specifies the highlight color; the color that is used to paint the boundary of the score board of the player whose turn it is. This value is also used to color the check box, the other indicator of whose turn it is. The value tpHighlightBorderColor is a three component value in RGB color space; the default is 1 0 0 (red). Note that there are *no commas* between the component values.

The jj_game creates a switch \ifjjgtwoplayer to signal the twoplayer option. This switch may be used, for example when specifying the value of CellWidth in \APScore:

```
\APScore {
    Size: 0,
```

```
    tpScaledCellWidth: 1,
    CellWidth: \ifjjgtwoplayer\cellWidth+20pt\else2in\fi,
}
```

In the above, for two player mode, we set the cell width of the score fields to be the `\cellWidth` of the cells on the game board plus `20pt`; otherwise, the width is set to `2in`. We also set `tpScaledCellWidth` to 1 so the calculated width of the two player score fields will be `\cellWidth+20pt` and not half this value, which is what it would be if `tpScaledCellWidth` maintains its default value of .5.[10]

The `\setTwoPlayerOptions` command allows you to specify some other twoplayer key-values.

```
\setTwoPlayerOptions {
    player1: Player 1,
    player2: Player 2,
    playerFmt: \sffamily\small,
    checkbox: true
}
```

When in two player mode, there are two score fields. Beneath these fields are *contestant labels*. The values of `player1` and `player2` are used to specify these labels. (The default values are shown in the example above.) The playerFmt key is a convenient way of specifying the formatting for the contestant labels. The default is `\sffamily\small`. Finally, there are two check boxes that appear, in addition to the two score fields, that indicate the turn. The default is that these check boxes appear, if you say, `checkbox: false`, the check boxes are not created.

8. Additional Customizations

This section includes many commands for customizing the look of a Jeopardy game.

8.1. Customize those Text Strings

In addition to the many strings already documented earlier in this manual, there are a few others. These strings are either AcroForm strings or JavaScript strings, so accents must use octal coding or unicode. For example, use `\string\344` or `\string\u00E4` (version 7.0 or later) for ä.

The following strings appear either as text on a button face or as a message in an alert box. The meaning of each command should be clear.

```
\correctText{Right!}
\incorrectText{False!}
\enterNamePlease{Please, enter your name on the first
    page before you begin the game!}
\illegalAccessMsg{You are peeking at this question. Choose a
    question from the game board first! You may want to start
```

[10] Actually, if `tpScaledCellWidth` were .5, the width of the score fields will be `\cellWidth`, see the description of `tpScaledCellWidth` above.

```
    the game over now!}
\illegalAnswerTwiceMsg{Changing your answer is not allowed!}
```

New values of these commands must be entered into the preamble.

There are some strings specific to the `twoplayer` option:

```
\playeriWinnerMsg{\jjg@playeri\space is the winner!}
\playeriiWinnerMsg{\jjg@playerii\space is the winner!}
\gameTiedMsg{It is all tied up! You are equally matched!}
\bothLosersMsg{You two either are not trying,
    or you do not know anything about the subject!}
```

The commands `\jjg@playeri` and `\jjg@playerii` hold the contestant labels, as defined by the values of `player1` and `player2` of `\setTwoPlayerOptions`. New values of these commands must be entered into the preamble.

8.2. Graphical Backgrounds

Beginning with version 3.0 of jj_game, graphical backgrounds can be inserted into the Jeopardy game. The basic command for doing this is `\template`. The `\template` command is defined in the Web package. An example of its usage is

```
\template{myCoolGraphic}
```

where `myCoolGraphic` is a graphic, an EPS file if the driver is dvips or dvipsone, or a PDF graphic for pdftex. Generally, any graphic the driver supports. Place the `\template` command between pages, the graphic will be insert in the next page.

However, to make things even easier, jj_game defines three commands:

```
\defineInstructionPageGraphic{⟨graphic⟩}
\defineGameboardPageGraphic{⟨graphic⟩}
\defineQuestionPagesGraphic{⟨graphic⟩}
```

Command Location: Place in the preamble of the document.

Command Description: The `\defineInstructionPageGraphic` command inserts the graphic for the instruction page(s). The `\defineGameboardPageGraphic` command inserts the graphic for the game board page. Finally, `\defineQuestionPagesGraphic` inserts the graphic for the question pages. See the AeB Manual for more details on the use of templates.

The `web` package manages the templates of the document. Web will resize the graphic to fit the dimensions of the page; as a result, any graphical design may be slightly or greatly distorted. A slight distortion may be acceptable, but a large distortion is not.

▶ The file `jeopardy_demo` demonstrates the work of Simon Singer and Jürgen Gilg, both of Germany. In that game, there are four categories and five questions per category. The game was built and the positioning of the various elements of the game (banners, gameboard, score field, etc.) are then known. Simon brought each of the three types of pages (instruction, game board, and question) into Adobe Photoshop and designed

graphics that fit exactly around the various elements. Using these graphics for the game then yielded the results you see in the Jeopardy game `jeopardy_demo.pdf`. Very swave!

You may be satisfied with a colorful background, a simple graphic as a background, or something based on a high skill level, such as what is demonstrated by Simon Singer and Jürgen Gilg. In the end, it is all up to you.

9. Color Summary

In this section, a visual presentation of the colors that can be set through the various commands.

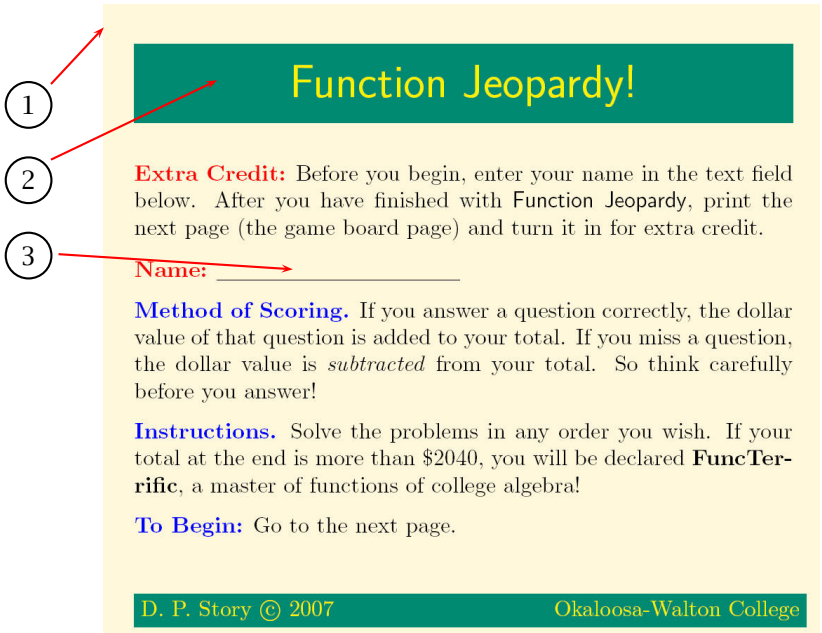

Figure 4: Instructions Page

Explanations

1. This is the background color of the Instruction page.
   Setup: `fillInstructions: myColor` in the `\DeclareColors` command
2. These are the colors of the title banner of the Instruction page.
   Setup: In the `\DeclareColors` command, use `fillBanner: myColor` for the banner background and `textBanner: myColor` for the banner text
3. This is a little special to integrate the user's name to identify who's the one that takes the game.
   Setup: `\underbar{\contestantName{1.5in}{11bp}}`

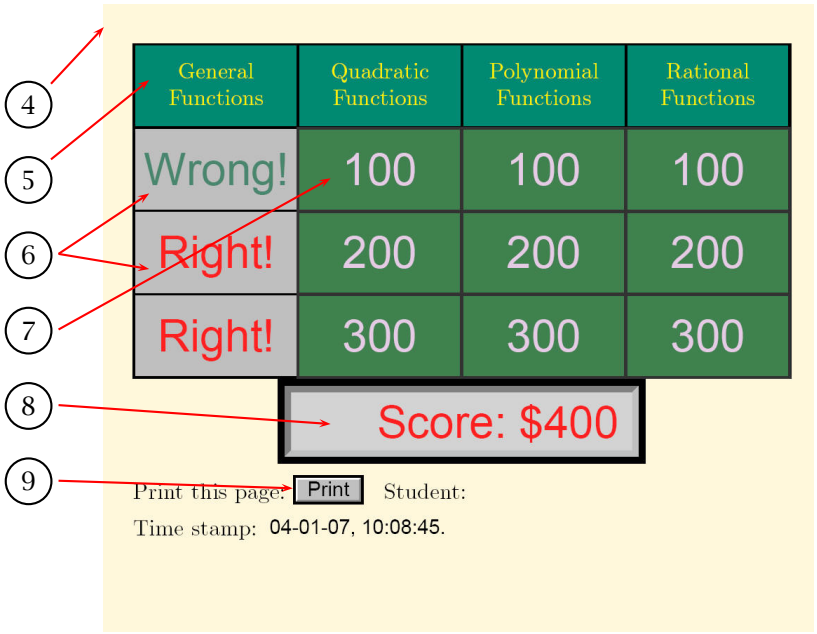

Figure 5: Gameboard Page

Explanations

(4) This is the background color of the Game Board page.

Setup: `fillGameBoard: myColor` in the `\DeclareColors` command.

(5) These are the colors of the game board Categories.

Setup: In the `\DeclareColors` command, use `fillBanner: myColor` for the category background and `textBanner: myColor` for the category text.

Note: This is named equal to the instructions page and possible to be set locally.

(6) These are the colors of the game board Cells **after** the questions are taken.

Setup: `TextColor: 0 0 0, BorderColor:` 0 0 0, for the cell text and border color in the `\APRight` and `\APWrong` commands. The background color is set up using the `fillCells` in the `\DeclareColors` command.

Note: Here the colors are defined in the rgb color scheme.

(7) These are the colors of the game board Cells **before** the questions are taken.

Setup: For the `\APDollar` command, use `TextColor: 0 0 0, BorderColor: 0 0 0, FillColor: 0.92 0.67 0.1` for the cell text, border and background color, respectively.

Note: Here the colors are defined in the rgb color scheme.

(8) These are the colors of the Score field.

Setup: `BorderColor: 0 0 0, FillColor: 0.92 0.67 0.1` for the Score border and background color in the `\APScore` command.

Note: The text color of the score is setup in the colors of the `\APRight` and `\APWrong` commands, so that the user may quickly see, if she/he is in positive or negative score. A minus is additionally setup if the user lost more points than she/he won.

The customer can choose an individual currency as well.

(9) This is a print option, to print out the game board, after the game is finished. This may be useful for teachers, to collect the individual results of her/his students. That's the reason, why in the instructions page the name was forced to be filled in. Now the name is as well on the print out.

This is automatically setup when the `forcredit` option for the `jj_game` class is taken.

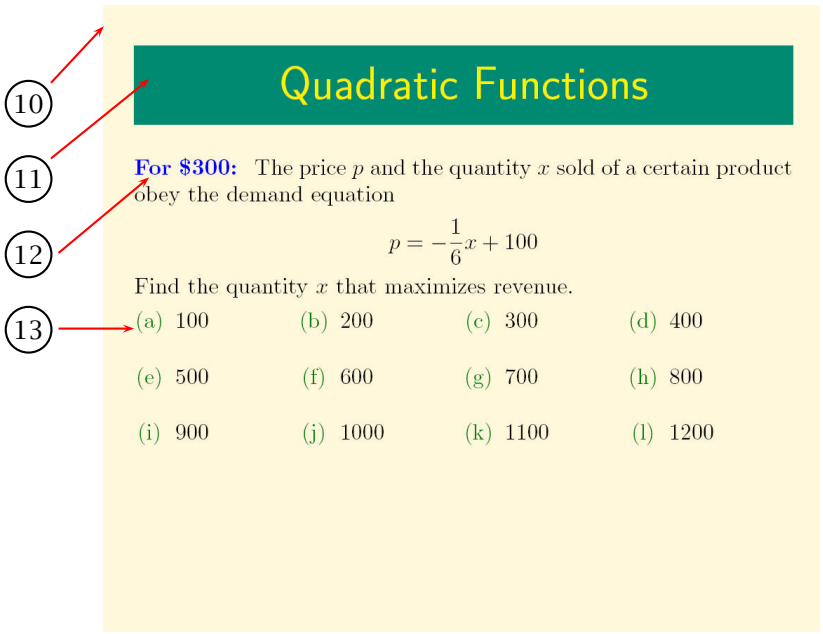

Figure 6: Questions Page

Explanations

(10) This is the background color of the Questions page.

Setup: `fillQuestions: myColor` in the `\DeclareColors` command

(11) These are the colors of the questions Categories.

Setup: In the `\DeclareColors` command, use `fillBanner: myColor` for the questions category background and `textBanner: myColor` for the questions category text.

Note: This is named equal to the banner of the instructions page and possible to be set locally.

(12) This is a referring text to the taken question score.

Setup: `dollarColor: myColor` for the referring text color in the `\DeclareColors` command.

Note: Text changes can be done with the command:

```
\currencyHeading{\bfseries For~\$\theCurrencyAmt:}
```

(13) This is the color of the links (answer possibilities).

Setup: `linkColor: myColor` for the link color in the `\DeclareColors` command