Package 'rquiz'

May 9, 2026

Type Package

Title Interactive Quizzes as HTML Widgets

Version 1.0.0

Description Creates interactive JavaScript-based quizzes as 'HTML' widgets. Offers three quiz types: a single question with instant feedback (singleQuestion()), a multi-question quiz with navigation, timer, and results (multiQuestions()), and fill-in-the-blank cloze exercises (fillBlanks()). All quizzes auto-detect single-choice and multiple-choice modes from the input data, support customizable styling, keyboard navigation, and multilingual UI (English, German, French, Spanish). Designed for use in 'R Markdown', 'Quarto', and 'Shiny' applications. The singleQuestion() quiz design was inspired by Ozzie Kirkby <https://codepen.io/ozzie/pen/pvrVLm>. The multiQuestions() quiz design was inspired by Abhilash Narayan <https://codepen.io/abhilashn/pen/BRepQz>.

URL https://github.com/saskiaotto/rquiz, https://saskiaotto.github.io/rquiz/

BugReports https://github.com/saskiaotto/rquiz/issues

VignetteBuilder knitr

License MIT + file LICENSE

Language en-US

Depends R (>= 4.1.0)

Imports htmlwidgets (>= 1.5.0), jsonlite (>= 1.7.0)

Suggests knitr, rmarkdown, shiny, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Saskia Otto [aut, cre] (ORCID: <https://orcid.org/0000-0001-7780-1322>)

Maintainer Saskia Otto <saskia.a.otto@gmail.com>

Repository CRAN

Date/Publication 2026-04-16 10:20:03 UTC

Contents

`checkFillBlanks` *Validate a fill-in-the-blank question object*

Description

Checks that a cloze question list is correctly structured for use in `fillBlanks`. Validates required fields, cloze markers, and optional additional options.

Usage

```
checkFillBlanks(x)
```

Arguments

`x`	A named list with at least the following element:

cloze A character string containing the cloze text with blank markers. Each blank must be enclosed between `$$!` (opening tag) and `!$$` (closing tag), e.g. `"R is a $$!programming!$$ language"`.

addOptions (optional) A character vector of additional distractor options displayed as answer choices alongside the correct answers.

Value

The input `x` invisibly if all checks pass.

See Also

`fillBlanks` which calls this function internally.

Examples

```
# Valid cloze text
txt <- list(
  cloze = "R is a $$!programming!$$ language for $$!statistical computing!$$.",
  addOptions = c("natural", "colloquial")
)
checkFillBlanks(txt)

# Minimal (without addOptions)
txt2 <- list(cloze = "The capital of France is $$!Paris!$$.")
checkFillBlanks(txt2)
```

`checkMultiQuestions` *Validate a list of question objects*

Description

Checks that all questions in a list are correctly structured for use in `multiQuestions`. Each question is validated via `checkSingleQuestion`. The list elements can be named (e.g. `q1`, `q2`) or unnamed — unnamed elements are automatically assigned names `q1`, `q2`, etc.

Usage

```
checkMultiQuestions(x)
```

Arguments

`x`	A list of question lists, each conforming to the structure described in `checkSingleQuestion`.

Value

The input `x` (with names assigned if needed) invisibly if all checks pass.

See Also

`multiQuestions` which calls this function internally, `checkSingleQuestion` for validating individual questions.

Examples

```
questions <- list(
  q1 = list(
    question = "Which planet is closest to the Sun?",
    options  = c("Venus", "Mercury", "Mars"),
    answer   = 2L
  ),
  q2 = list(
    question = "Which are noble gases?",
```

```
    options  = c("Helium", "Oxygen", "Neon", "Iron"),
    answer   = c(1L, 3L)
  )
)
checkMultiQuestions(questions)
```

`checkSingleQuestion` *Validate a single question object*

Description

Checks that a question list is correctly structured for use in `singleQuestion`. Validates required fields, answer index bounds, and detects question type automatically.

Usage

```
checkSingleQuestion(x)
```

Arguments

x	A named list with at least the following elements: **question** A character string with the question text. **options** A character vector of answer options (min. 2). **answer** An integer vector of correct answer index/indices (1-based). Length > 1 is interpreted as multiple-choice. **tip** (optional) A character string shown as a hint.

Value

The input `x` invisibly if all checks pass.

See Also

`singleQuestion` which calls this function internally, `checkMultiQuestions` for validating lists of questions.

Examples

```
# Single-choice question
q_sc <- list(
  question = "Which planet is closest to the Sun?",
  options  = c("Venus", "Mercury", "Mars"),
  answer   = 2L,
  tip      = "It's also the smallest planet."
)
checkSingleQuestion(q_sc)

# Multiple-choice question (auto-detected from answer length)
```

```
q_mc <- list(
  question = "Which are noble gases?",
  options  = c("Helium", "Oxygen", "Neon", "Iron"),
  answer   = c(1L, 3L)
)
checkSingleQuestion(q_mc)
```

`fillBlanks` *Create a fill-in-the-blank cloze quiz*

Description

`fillBlanks` creates an interactive cloze (fill-in-the-blank) quiz where users type missing words into blank fields. Blank fields are defined by placing answer text between `$$!` and `!$$` markers in the cloze string. A submit button checks all answers at once. The description container changes color as feedback: green if all answers are correct, red otherwise (with an alert showing the score). Optional tip and solution buttons provide additional help.

Usage

```
fillBlanks(
  x,
  title = NULL,
  description = NULL,
  language = "en",
  showTipButton = FALSE,
  showSolutionButton = TRUE,
  width = "100%",
  height = "500px",
  center = TRUE,
  scroll = FALSE,
  clozeWidth = "100%",
  clozeHeight = NULL,
  blanksWidth = "auto",
  blanksAlign = "center",
  fontFamily = "'Helvetica Neue', Helvetica, Arial, sans-serif",
  fontSize = 16,
  titleFontSize = 20,
  descriptFontSize = 16,
  titleAlign = "left",
  titleCol = "#FFFFFF",
  titleBg = "#7E7E7E",
  questionCol = "#FFFFFF",
  questionBg = "#5F5F5F",
  mainCol = "#1C1C1C",
  mainBg = "#F7F7F7",
  navButtonCol = "#FFFFFF",
```

```
  navButtonBg = "#28A745",
  tipButtonCol = "#1C1C1C",
  tipButtonBg = "#F1F1F1",
  solutionButtonCol = "#1C1C1C",
  solutionButtonBg = "#F1F1F1",
  tipAreaCol = "#1C1C1C",
  tipAreaBg = "#E7F9E7",
  tipAreaBorder = "#28A745",
  solutionAreaCol = "#1C1C1C",
  solutionAreaBg = "#D6EAF8",
  solutionAreaBorder = "#3498DB",
  theme = NULL
)
```

Arguments

x	A named list containing: • $cloze (required) A character string with the full text. Words or phrases to become blanks are wrapped in $$!answer!$$ markers. • $addOptions (optional) A character vector of additional (wrong) answer options shown in the tip area. Use checkFillBlanks to validate the input before passing it.
title	character; the title text. If set to NULL (default), no title will be shown in the header box.
description	character; a short description below the title. If set to NULL (default), a language-specific default text is used (e.g. "Fill in the blanks." for English, "Fülle die Lücken." for German). Set to "" (empty string) to hide the description entirely.
language	character; the language of the quiz interface. Sets button labels, result text, tips and description automatically. Currently supported: "en" (English, default), "de" (German), "fr" (French), and "es" (Spanish). To request additional languages, please open an issue on GitHub.
showTipButton	logical; whether the tip button for answer options should be displayed. The default is FALSE.
showSolutionButton	logical; whether the solution button should be displayed. The default is TRUE.
width	character; the width of the quiz frame area in pixels or percentage. The default is "100%".
height	character; the height of the quiz frame area in pixels or percentage. The default is "500px". If scroll = FALSE, the height is automatically determined based on content.
center	logical; if TRUE (default), the quiz frame is centered when the width is less than "100%".
scroll	logical; if TRUE, the height argument sets a fixed frame height with a scroll bar for overflowing content. Useful for HTML presentations. If FALSE (default), the height adjusts automatically.

clozeWidth — character; the width of the cloze section in pixels or percentage. The default is "100%".

clozeHeight — character or NULL; the minimum height of the cloze section. The default is NULL (auto). Set to e.g. "250px" for a fixed minimum height.

blanksWidth — character; the width of blank input fields. The default is "auto", which sizes fields based on the longest correct answer (minimum 50px). If the cloze text uses <pre> tags, a monospace font is used for measuring. Set to a fixed value like "150px" for a specific width, or NULL for browser default.

blanksAlign — character; the alignment of text in filled blanks. One of "center" (default), "left", or "right".

fontFamily — character; the font family for all text elements in CSS style. The default is "'Helvetica Neue', Helvetica, Arial, sans-serif".

fontSize — integer; the font size in pixels for the main text elements. Other properties are rescaled accordingly. The default is 16.

titleFontSize — integer; the font size in pixels for the title. The default is 20.

descriptFontSize — integer; the font size in pixels for the description. The default is 16.

titleAlign — character; the alignment of the title text. One of "left" (default), "center", or "right".

titleCol — character; the title (and description) text color as hex color or R color name. The default is "#FFFFFF".

titleBg — character; the title box background color as hex color or R color name. The default is "#7E7E7E".

questionCol — character; the description text color. The default is "#FFFFFF". Named questionCol for consistency with singleQuestion and multiQuestions, where this parameter controls the question container. In fillBlanks, it controls the description container, which also provides visual feedback after submission (green for all correct, red for errors).

questionBg — character; the background color of the description container. The default is "#5F5F5F".

mainCol — character; the text color of the cloze content area. The default is "#1C1C1C".

mainBg — character; the background color of the main content area. The default is "#F7F7F7".

navButtonCol — character; the submit button text color. The default is "#FFFFFF".

navButtonBg — character; the submit button background color. The default is "#28A745".

tipButtonCol — character; the tip button text color. The default is "#1C1C1C".

tipButtonBg — character; the tip button background color. The default is "#F1F1F1".

solutionButtonCol — character; the solution button text color. The default is "#1C1C1C".

solutionButtonBg — character; the solution button background color. The default is "#F1F1F1".

tipAreaCol — character; the text color of the tip area. The default is "#1C1C1C".

tipAreaBg — character; the background color of the tip answer options area. The default is "#E7F9E7".

`tipAreaBorder` character; the border color of the tip answer options area. The default is `"#28A745"`.

`solutionAreaCol` character; the text color of the solution area. The default is `"#1C1C1C"`.

`solutionAreaBg` character; the background color of the solution area. The default is `"#D6EAF8"`. Note: in fillBlanks, solutions are shown directly in the input fields, so this parameter is reserved for future use.

`solutionAreaBorder` character; the border color of the solution area. The default is `"#3498DB"`. See note for `solutionAreaBg`.

`theme` an optional `rquizTheme` object. Theme values are used as defaults and are overridden by any explicitly passed arguments.

Details

Cloze exercises are widely used in language learning, programming courses, and knowledge assessment. They can be flexibly tailored to meet different learning needs.

Cloze text format

The full cloze text needs to be provided in one single string. Any word, symbol or other text item that should be removed from the cloze and become an answer option has to be placed between an opening 'tag' `$$!` and a closing 'tag' `!$$`. In the following example the words **programming** and **statistical computing** will be replaced with blank fields and converted into blank input fields:

x = list(cloze = "R is a $$!programming!$$ language and free software environment for $$!statistical computing!$$ and graphics.")

To add more (but incorrect) answer options, simply add a character vector named `$addOptions` to the input list.

Formatting of cloze text

The cloze string will be converted straight into html code. If you want to format the text in a specific way, e.g. show some text parts in bold or italic or a specific color, use HTML tags with inline CSS style:

x = list(cloze = "R is a $$!programming!$$ language and free software environment for $$!statistical computing!$$ and graphics.")

Normal text wraps automatically. For monospace code blocks, use **<pre>** tags — inside `<pre>`, line breaks require the **
** tag (normal line breaks are ignored).

Value

An HTML widget object of class `fillBlanks` that can be rendered in R Markdown/Quarto documents, Shiny applications, or the RStudio viewer.

Author(s)

Saskia A. Otto

See Also

`checkFillBlanks` for input validation, `singleQuestion` for choice quizzes, `multiQuestions` for multi-page quizzes, `rquizTheme` for reusable themes.

Examples

```
# --- BASIC USAGE ---
txt <- list(
  cloze = "R is a $$!programming!$$ language and free
    software environment for $$!statistical computing!$$
    and graphics."
)

### Check if list has the correct format
checkFillBlanks(txt)  # passes

### Default settings (solution button shown)
fillBlanks(x = txt)

### Hide solution button
fillBlanks(x = txt, showSolutionButton = FALSE)

### Tip button with additional wrong options
txtWithOptions <- list(
  cloze = "R is a $$!programming!$$ language and free
    software environment for $$!statistical computing!$$
    and graphics.",
  addOptions = c("natural", "colloquial", "movies", "audio")
)
fillBlanks(x = txtWithOptions, showTipButton = TRUE)

### Title and description
fillBlanks(x = txt, title = "What is R?",
  description = "Can you complete the following sentence?")

### Changing the language
fillBlanks(x = txt, showTipButton = TRUE, language = "de")
fillBlanks(x = txt, showTipButton = TRUE, language = "fr")
fillBlanks(x = txt, showTipButton = TRUE, language = "es")


# --- CHANGING THE STYLE ---

### Window and cloze size
fillBlanks(x = txt, width = "50%")
fillBlanks(x = txt, width = "50%", center = FALSE)
# height only works with scroll = TRUE
fillBlanks(x = txt, width = "50%", height = "200px", scroll = TRUE)
fillBlanks(x = txt, clozeWidth = "50%")
fillBlanks(x = txt, clozeHeight = "300px")

### Blank field size and alignment
fillBlanks(x = txt, blanksWidth = "400px")
fillBlanks(x = txt, blanksAlign = "left")
fillBlanks(x = txt, blanksAlign = "right")

### Customizing the design: dark theme
```

```
fillBlanks(x = txtWithOptions,
  title = "What is R?", showTipButton = TRUE,
  fontFamily = "Georgia, serif",
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  mainCol = "#E0E0E0", mainBg = "#1A1B2E",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560",
  tipButtonCol = "#E0E0E0", tipButtonBg = "#2C2C3E",
  solutionButtonCol = "#E0E0E0", solutionButtonBg = "#2C2C3E",
  tipAreaCol = "#E0E0E0", tipAreaBg = "#1A2A1A",
  tipAreaBorder = "#4CAF50",
  solutionAreaCol = "#E0E0E0", solutionAreaBg = "#1A1A2E",
  solutionAreaBorder = "#5DADE2")

### Using a reusable theme
dark <- rquizTheme(
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560"
)
fillBlanks(x = txtWithOptions, theme = dark,
  title = "What is R?", showTipButton = TRUE)


# --- CODE SYNTAX EXAMPLE ---

### Use <pre> tags for monospace formatting
codequiz <- list(
  cloze = "<pre>myFunc <- $$!function()!$$ { <br>
    print('Hello, world!') <br>
  $$!}!$$ <br>
  $$!myFunc()!$$</pre>",
  addOptions = c("function", ")", "myFunc")
)
fillBlanks(x = codequiz,
  title = "How to define and call a function",
  blanksAlign = "left")
```

`fillBlanks-shiny` *Shiny bindings for fillBlanks*

Description

Output and render functions for using fillBlanks within Shiny applications and interactive Rmd documents.

Usage

```
fillBlanksOutput(outputId, width = "100%", height = "500px")

renderFillBlanks(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a fillBlanks quiz
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

fillBlanksOutput returns an HTML output element for use in a Shiny UI definition. renderFillBlanks returns a server-side rendering function to be assigned to an output slot.

Examples

```
if (interactive()) {
library(shiny)

txt <- list(
  cloze = "R is a $$!programming!$$ language for $$!statistical computing!$$.",
  addOptions = c("natural", "cooking")
)

ui <- fluidPage(
  fillBlanksOutput("quiz1")
)

server <- function(input, output) {
  output$quiz1 <- renderFillBlanks({
    fillBlanks(x = txt, title = "Fill in the Blanks")
  })
}

shinyApp(ui, server)
}
```

multiQuestions *Create a multi-page quiz with navigation, timer, and results*

Description

multiQuestions creates an interactive quiz that presents multiple questions one at a time with Previous/Next navigation. A timer tracks how long the user takes (optional), and a results page shows the total score with an optional 'Show solution' button to reveal correct answers. Supports both single-choice and multiple-choice questions.

Usage

```
multiQuestions(
  x,
  title = "Quiz",
  language = "en",
  shuffle = FALSE,
  showTipButton = FALSE,
  showSolutionButton = TRUE,
  inclTimer = TRUE,
  width = "100%",
  height = "500px",
  center = TRUE,
  scroll = FALSE,
  fontFamily = "'Helvetica Neue', Helvetica, Arial, sans-serif",
  fontSize = 16,
  titleFontSize = 20,
  questionFontSize = 16,
  titleAlign = "left",
  titleCol = "#FFFFFF",
  titleBg = "#5F5F5F",
  questionCol = "#1C1C1C",
  questionBg = "#F7F7F7",
  mainCol = "#1C1C1C",
  mainBg = "#F7F7F7",
  optionBg = "#ECECEC",
  timerCol = "#1C1C1C",
  navButtonCol = "#FFFFFF",
  navButtonBg = "#28A745",
  tipButtonCol = "#1C1C1C",
  tipButtonBg = "#F1F1F1",
  tipAreaCol = "#1C1C1C",
  tipAreaBg = "#E7F9E7",
  tipAreaBorder = "#28A745",
  solutionButtonCol = "#1C1C1C",
  solutionButtonBg = "#F1F1F1",
  solutionAreaCol = "#1C1C1C",
```

```
  solutionAreaBg = "#D6EAF8",
  solutionAreaBorder = "#3498DB",
  theme = NULL
)
```

Arguments

x	A list of question lists. Each sub-list should contain: • $question A character string with the question text. • $options A character vector of answer options (min. 2). • $answer An integer index (SC) or numeric vector of indices (MC) of the correct answer option(s). • $tip (optional) A character string shown as a custom tip when showTipButton = TRUE. Sub-lists can be named (e.g. q1, q2) for readability, but this is not required. Use checkMultiQuestions to validate the input before passing it.
title	character; the title text. The default is "Quiz".
language	character; the language of the quiz interface. Sets button labels, navigation text, result page text and other UI elements automatically. Currently supported: "en" (English, default), "de" (German), "fr" (French), and "es" (Spanish). To request additional languages, please open an issue on GitHub.
shuffle	logical; if TRUE, the order of questions is randomized each time the widget is rendered or the quiz is restarted with 'Try again'. The default is FALSE.
showTipButton	logical; whether to show a tip button on each question page. If TRUE and the question list contains a 4th element (tip text), the custom tip is displayed. In multiple-choice mode without a custom tip, the number of correct answers is shown. In single-choice mode, the button only appears for questions with a custom tip. The default is FALSE.
showSolutionButton	logical; if TRUE (default), a 'Show solution' button is displayed on the results page, allowing users to reveal the correct answers. If FALSE, only the score (0 or 1) per question is shown.
inclTimer	logical; whether to include a timer (TRUE, default) or not. The timer starts counting immediately after the page is loaded.
width	character; the width of the quiz frame area in pixels or percentage. The default is "100%".
height	character; the height of the quiz frame area in pixels or percentage. The default is "500px". If scroll = FALSE, the height is automatically determined based on content.
center	logical; if TRUE (default), the quiz frame is centered when the width is less than "100%".
scroll	logical; if TRUE, the height argument sets a fixed frame height with a scroll bar for overflowing content. Useful for HTML presentations. If FALSE (default), the height adjusts automatically.

`fontFamily` character; the font family for all text elements in CSS style. The default is `"'Helvetica Neue', Helvetica, Arial, sans-serif"`.

`fontSize` integer; the font size in pixels for the main text elements. All other properties are rescaled accordingly. The default is `16`.

`titleFontSize` integer; the font size in pixels for the title. The default is `20`.

`questionFontSize` integer; the font size in pixels for the question text. The default is `16`.

`titleAlign` character; the alignment of the title text. One of `"left"` (default), `"center"`, or `"right"`.

`titleCol` character; the title text color as hex color or R color name. The default is `"#FFFFFF"`.

`titleBg` character; the title background color as hex color or R color name. The default is `"#5F5F5F"`.

`questionCol` character; the question text color. The default is `"#1C1C1C"`. The question is displayed in its own container between the title and the answer options.

`questionBg` character; the background color of the question container. The default is `"#F7F7F7"`.

`mainCol` character; the text color of the answer options and main content area. The default is `"#1C1C1C"`.

`mainBg` character; the background color of the main content area. The default is `"#F7F7F7"`.

`optionBg` character; the background color of the answer option blocks. The hover color is automatically derived (30% darker). The default is `"#ECECEC"`.

`timerCol` character; the timer text color as hex color or R color name. The default is `"#1C1C1C"`.

`navButtonCol` character; the text color of the navigation buttons (Previous, Next, Try again). The default is `"#FFFFFF"`.

`navButtonBg` character; the background color of the navigation buttons (Previous, Next, Try again). The default is `"#28A745"`.

`tipButtonCol` character; the tip button text color. The default is `"#1C1C1C"`.

`tipButtonBg` character; the tip button background color. The default is `"#F1F1F1"`.

`tipAreaCol` character; the text color of the tip area. The default is `"#1C1C1C"`.

`tipAreaBg` character; the background color of the tip area. The default is `"#E7F9E7"`.

`tipAreaBorder` character; the border color of the tip area. The default is `"#28A745"`.

`solutionButtonCol` character; the 'Show solution' button text color. The default is `"#1C1C1C"`.

`solutionButtonBg` character; the 'Show solution' button background color. The default is `"#F1F1F1"`.

`solutionAreaCol` character; the text color of the solution answer areas on the results page. The default is `"#1C1C1C"`.

`solutionAreaBg` character; the background color of the solution answer areas on the results page. The default is `"#D6EAF8"`.

`solutionAreaBorder`
: character; the border color of the solution answer areas on the results page. The default is `"#3498DB"`.

`theme`
: an optional `rquizTheme` object. Theme values are used as defaults and are overridden by any explicitly passed arguments.

Details

The quiz mode is auto-detected: if **any** question has multiple correct answers (`length(answer) > 1`), all questions use checkboxes (multiple-choice mode). Otherwise, radio buttons are used (single-choice). An optional tip button can provide hints per question (custom text or, in MC mode, the number of correct answers).

Text formatting: Since quiz content is rendered as HTML, you can use HTML tags in question and option strings: `<em>` for italics (e.g. species names), `<strong>` for bold, `<code>` for inline code, or `<span>` with inline CSS for colored text. Standard Markdown formatting does *not* work inside quiz strings.

Value

An HTML widget object of class `multiQuestions` that can be rendered in R Markdown/Quarto documents, Shiny applications, or the RStudio viewer.

Author(s)

Saskia A. Otto

See Also

`checkMultiQuestions` for input validation, `singleQuestion` for single-question quizzes, `fillBlanks` for cloze quizzes, `rquizTheme` for reusable themes.

Examples

```
# --- SINGLE-CHOICE MODE ---

### Sub-lists can be named (e.g. q1, q2) for readability but don't have to.
sportQuiz <- list(
  q1 = list(
    question = "Which city hosted the 2024 Summer Olympics?",
    options = c("Athens", "London", "Paris", "Tokyo"),
    answer = 3  # i.e. Paris
  ),
  q2 = list(
    question = "Which of the following is NOT an official
      Olympic summer sport?",
    options = c("Beach Volleyball", "Baseball", "Football",
      "American Football", "Basketball", "Wrestling"),
    answer = 4
  ),
  q3 = list(
    question = "Which country has won the most Winter Olympic
```

```
      gold medals of all time?",
    options = c("USA", "Canada", "Germany",
      "Russia", "Sweden", "Norway"),
    answer = 6
  )
)

### Check if list has the correct format
checkMultiQuestions(sportQuiz)   # passes

### Default settings
multiQuestions(x = sportQuiz)

### Custom title, shuffled questions, no timer, no solution button
multiQuestions(x = sportQuiz, title = "Sports Quiz",
  shuffle = TRUE, inclTimer = FALSE, showSolutionButton = FALSE)

### With tip button (only questions with a tip element show tips in SC mode)
sportQuizWithTips <- list(
  q1 = list(
    question = "Which city hosted the 2024 Summer Olympics?",
    options = c("Athens", "London", "Paris", "Tokyo"),
    answer = 3,
    tip = "They speak French there."
  ),
  q2 = list(
    question = "Which sport was added to the Olympics
      at the 2024 Paris Games?",
    options = c("Skateboarding", "Breaking",
      "Surfing", "Sport Climbing"),
    answer = 2  # no tip for this question
  )
)
multiQuestions(x = sportQuizWithTips, showTipButton = TRUE)


# --- MULTIPLE-CHOICE MODE (auto-detected from answer vector) ---

### If ANY question has multiple correct answers, all use checkboxes.
geoQuiz <- list(
  q1 = list(
    question = "Which of the following cities are in Europe?",
    options = c("Berlin", "Tokyo", "Paris", "Sydney", "Rome"),
    answer = c(1, 3, 5),
    tip = "Three of the five cities are European capitals."
  ),
  q2 = list(
    question = "Which of these rivers flow through Germany?",
    options = c("Rhine", "Thames", "Danube", "Seine", "Elbe"),
    answer = c(1, 3, 5)
    # no custom tip: auto-generated "Number of correct answers: 3"
  )
)
```

```
### Default settings
multiQuestions(x = geoQuiz)

### Adding a title and showing the tip button
multiQuestions(x = geoQuiz, title = "Geography Quiz",
  showTipButton = TRUE)

### Changing the language
multiQuestions(x = sportQuiz, title = "Sport-Quiz",
  language = "de")
multiQuestions(x = sportQuiz, title = "Quiz sportif",
  language = "fr", shuffle = TRUE)
multiQuestions(x = sportQuiz, title = "Cuestionario deportivo",
  language = "es")


# --- CHANGING THE STYLE ---

### Adjusting the window size
multiQuestions(x = sportQuiz, width = "50%")
multiQuestions(x = geoQuiz, width = "50%", center = FALSE)
# height only works with scroll = TRUE
multiQuestions(x = sportQuiz, width = "50%",
  height = "300px", scroll = TRUE)

### Customizing the design: dark theme
multiQuestions(x = sportQuizWithTips,
  title = "Sports Quiz", showTipButton = TRUE,
  fontFamily = "Georgia, serif",
  fontSize = 20, titleFontSize = 14, questionFontSize = 14,
  titleCol = "#E2E8F0", titleBg = "#2D2D44",
  questionCol = "#E2E8F0", questionBg = "#252540",
  mainCol = "#E2E8F0", mainBg = "#1A1B2E",
  optionBg = "#252540", timerCol = "#F687B3",
  navButtonCol = "#FFFFFF", navButtonBg = "#6C63FF",
  tipButtonCol = "#68D391", tipButtonBg = "#2D2D44",
  tipAreaCol = "#68D391", tipAreaBg = "#162616",
  tipAreaBorder = "#68D391",
  solutionButtonCol = "#63B3ED", solutionButtonBg = "#2D2D44",
  solutionAreaCol = "#63B3ED", solutionAreaBg = "#111827",
  solutionAreaBorder = "#4299E1")

### Using a reusable theme
dark <- rquizTheme(
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560"
)
multiQuestions(x = sportQuiz, theme = dark, title = "Themed Sports Quiz")
multiQuestions(x = geoQuiz, theme = dark, title = "Themed Geography Quiz")
```

`multiQuestions-shiny` *Shiny bindings for multiQuestions*

Description

Output and render functions for using multiQuestions within Shiny applications and interactive Rmd documents.

Usage

```
multiQuestionsOutput(outputId, width = "100%", height = "400px")

renderMultiQuestions(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId`	output variable to read from
`width, height`	Must be a valid CSS unit (like `'100%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.
`expr`	An expression that generates a multiQuestions quiz
`env`	The environment in which to evaluate `expr`.
`quoted`	Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Value

`multiQuestionsOutput` returns an HTML output element for use in a Shiny UI definition. `renderMultiQuestions` returns a server-side rendering function to be assigned to an output slot.

Examples

```
if (interactive()) {
library(shiny)

x <- list(
  q1 = list(
    question = "What is 2+2?",
    options = c("3", "4", "5"),
    answer = 2
  ),
  q2 = list(
    question = "What is 3*3?",
    options = c("6", "9", "12"),
    answer = 2
  )
)
```

```
ui <- fluidPage(
  multiQuestionsOutput("quiz1")
)

server <- function(input, output) {
  output$quiz1 <- renderMultiQuestions({
    multiQuestions(x = x, title = "Math Quiz")
  })
}

shinyApp(ui, server)
}
```

`rquizTheme` *Create a reusable quiz theme*

Description

Defines a set of design parameters that can be applied to any rquiz function (`singleQuestion`, `multiQuestions`, `fillBlanks`). Parameters not recognized by a particular quiz type are silently ignored. Explicit values passed directly to a quiz function always override theme values.

Usage

```
rquizTheme(
  language = NULL,
  shuffle = NULL,
  showTipButton = NULL,
  showSolutionButton = NULL,
  width = NULL,
  height = NULL,
  center = NULL,
  scroll = NULL,
  fontFamily = NULL,
  fontSize = NULL,
  titleFontSize = NULL,
  titleAlign = NULL,
  titleCol = NULL,
  titleBg = NULL,
  questionCol = NULL,
  questionBg = NULL,
  questionFontSize = NULL,
  mainCol = NULL,
  mainBg = NULL,
  optionBg = NULL,
  optionLabelBg = NULL,
  timerCol = NULL,
```

```
  navButtonCol = NULL,
  navButtonBg = NULL,
  tipButtonCol = NULL,
  tipButtonBg = NULL,
  solutionButtonCol = NULL,
  solutionButtonBg = NULL,
  tipAreaCol = NULL,
  tipAreaBg = NULL,
  tipAreaBorder = NULL,
  solutionAreaCol = NULL,
  solutionAreaBg = NULL,
  solutionAreaBorder = NULL,
  descriptFontSize = NULL
)
```

Arguments

language	character; language for UI text ("en", "de", "fr", "es").
shuffle	logical; whether to randomize options/question order.
showTipButton	logical; whether to show a tip button.
showSolutionButton	logical; whether to show a solution button.
width	character; widget width (CSS value).
height	character; widget height (CSS value).
center	logical; whether to center the widget.
scroll	logical; whether to use a fixed height with scrollbar.
fontFamily	character; font stack for all text.
fontSize	numeric; base font size in pixels.
titleFontSize	numeric; title font size in pixels.
titleAlign	character; title text alignment.
titleCol	character; title text color.
titleBg	character; title background color.
questionCol	character; text color of the question container (singleQuestion, multiQuestions) or description container (fillBlanks). This container also provides visual feedback (green/red) in singleQuestion and fillBlanks.
questionBg	character; background color of the question/description container (all quiz types).
questionFontSize	numeric; question font size (singleQuestion, multiQuestions).
mainCol	character; text color of the main content area and answer options (all quiz types).
mainBg	character; background color of the main content area (all quiz types).
optionBg	character; background color of answer option rows/blocks (singleQuestion, multiQuestions). Hover is auto-derived (30% darker).
optionLabelBg	character; A/B/C label background (singleQuestion only).

`timerCol` character; timer text color (multiQuestions only).

`navButtonCol` character; navigation/submit button text color (all quiz types).

`navButtonBg` character; navigation/submit button background color (all quiz types).

`tipButtonCol` character; tip button text color.

`tipButtonBg` character; tip button background color.

`solutionButtonCol` character; solution button text color.

`solutionButtonBg` character; solution button background color.

`tipAreaCol` character; tip area text color.

`tipAreaBg` character; tip area background color.

`tipAreaBorder` character; tip area border color.

`solutionAreaCol` character; solution area text color.

`solutionAreaBg` character; solution area background color.

`solutionAreaBorder` character; solution area border color.

`descriptFontSize` numeric; description font size (fillBlanks only).

Value

A named list of class `"rquizTheme"` containing all specified parameters. Only non-NULL values are included.

See Also

`singleQuestion`, `multiQuestions`, `fillBlanks` — all accept a `theme` argument.

Examples

```
# Create a dark theme with shared + quiz-specific settings
dark <- rquizTheme(
  fontFamily = "Georgia, serif",
  fontSize = 20,
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  mainCol = "#E0E0E0", mainBg = "#1A1B2E",
  optionBg = "#252540", optionLabelBg = "#0F3460",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560",
  tipButtonCol = "#E0E0E0", tipButtonBg = "#2C2C3E",
  solutionButtonCol = "#E0E0E0", solutionButtonBg = "#2C2C3E",
  tipAreaCol = "#E0E0E0", tipAreaBg = "#1A2A1A",
  tipAreaBorder = "#4CAF50",
  solutionAreaCol = "#E0E0E0", solutionAreaBg = "#1A1A2E",
  solutionAreaBorder = "#5DADE2"
)
```

```
# Apply to singleQuestion:
singleQuestion(
  x = list(
    question = "Which ocean is the largest?",
    options = c("Atlantic", "Indian", "Pacific", "Arctic"),
    answer = 3
  ),
  theme = dark, title = "Geography Quiz"
)

# Apply to multiQuestions (uses mainCol, mainBg, navButtonBg, etc.):
multiQuestions(
  x = list(
    q1 = list(
      question = "What is the capital of Japan?",
      options = c("Seoul", "Tokyo", "Beijing"),
      answer = 2
    ),
    q2 = list(
      question = "Which river is the longest?",
      options = c("Amazon", "Nile", "Yangtze"),
      answer = 2
    )
  ),
  theme = dark, title = "Geography Quiz"
)

# Apply to fillBlanks:
fillBlanks(
  x = list(
    cloze = "The $$!Nile!$$ is the longest river in $$!Africa!$$."
  ),
  theme = dark, title = "Geography Quiz"
)

# Explicit values override the theme:
singleQuestion(
  x = list(
    question = "Which ocean is the largest?",
    options = c("Atlantic", "Indian", "Pacific", "Arctic"),
    answer = 3
  ),
  theme = dark, titleBg = "#FF0000"
)
```

`singleQuestion` *Create a single-question quiz (single-choice or multiple-choice)*

Description

singleQuestion creates an interactive quiz with one question. It supports both single-choice and multiple-choice modes, which are auto-detected from the answer field (single integer = SC, vector of integers = MC).

Usage

```
singleQuestion(
  x,
  title = NULL,
  language = "en",
  shuffle = FALSE,
  showTipButton = FALSE,
  showSolutionButton = TRUE,
  width = "100%",
  height = "500px",
  center = TRUE,
  scroll = FALSE,
  fontFamily = "'Helvetica Neue', Helvetica, Arial, sans-serif",
  fontSize = 16,
  titleFontSize = 20,
  questionFontSize = 16,
  titleAlign = "center",
  titleCol = "#FFFFFF",
  titleBg = "#7E7E7E",
  questionCol = "white",
  questionBg = "#5F5F5F",
  mainCol = "#1C1C1C",
  mainBg = "#F7F7F7",
  optionBg = "#ECECEC",
  optionLabelBg = "#3498DB",
  navButtonCol = "#FFFFFF",
  navButtonBg = "#28A745",
  tipButtonCol = "#1C1C1C",
  tipButtonBg = "#F1F1F1",
  solutionButtonCol = "#1C1C1C",
  solutionButtonBg = "#F1F1F1",
  tipAreaCol = "#1C1C1C",
  tipAreaBg = "#E7F9E7",
  tipAreaBorder = "#28A745",
  solutionAreaCol = "#1C1C1C",
  solutionAreaBg = "#D6EAF8",
  solutionAreaBorder = "#3498DB",
  theme = NULL
)
```

Arguments

x	A list containing 3 or 4 elements (which can be named for better readability): • $question A string providing the actual question. • $options A character vector containing all answer options. • $answer An integer indicating the correct answer option (single-choice) or a numeric vector of indices indicating all correct answer options (multiple-choice). The quiz mode is auto-detected from the length of this field. • $tip (optional) A string providing a custom tip text, shown when showTipButton = TRUE. In multiple-choice mode, if omitted, the number of correct answers is shown as the default tip. In single-choice mode, the tip button only appears if a custom tip is provided. Use checkSingleQuestion to validate the input before passing it.
title	character; the title text. If set to NULL (default) the header box will not be shown.
language	character; the language of the quiz interface. Sets button labels, feedback messages and other UI text automatically. Currently supported: "en" (English, default), "de" (German), "fr" (French), and "es" (Spanish). To request additional languages, please open an issue on GitHub.
shuffle	logical; if TRUE, the answer options are displayed in a random order each time the widget is rendered. The correct answer mapping is preserved. The default is FALSE.
showTipButton	logical; whether to show a tip button. If TRUE and x contains a 4th element (e.g. $tip), the tip text is displayed. In multiple-choice mode without a custom tip, the number of correct answers is shown as a default tip. In single-choice mode, the button is only shown if a custom tip text is provided in x[[4]]. The default is FALSE.
showSolutionButton	logical; whether to show a solution button that reveals the correct answer(s). The default is TRUE.
width	character; the width of the quiz frame area in pixels or percentage. The default is "100%".
height	character; the height of the quiz frame area in pixels or percentage. The default is "500px". If scroll = FALSE, the height is automatically determined based on content.
center	logical; if TRUE (default), the quiz frame is centered when the width is less than "100%".
scroll	logical; if TRUE, the height argument sets a fixed frame height with a scroll bar for overflowing content. Useful for HTML presentations. If FALSE (default), the height adjusts automatically.
fontFamily	character; the font family for all text elements in CSS style. The default is "'Helvetica Neue', Helvetica, Arial, sans-serif".
fontSize	integer; the font size in pixels for the answer option text. All other properties (margins, letter buttons, etc.) are rescaled accordingly. The default is 16.
titleFontSize	integer; the font size in pixels for the title. The default is 20.

`questionFontSize`
: integer; the font size in pixels for the question text. The default is `16`.

`titleAlign`
: character; the alignment of the title text. One of `"left"`, `"center"` (default), or `"right"`.

`titleCol`
: character; the title text color as hex color or R color name. The default is `"#FFFFFF"`.

`titleBg`
: character; the title background color as hex color or R color name. The default is `"#7E7E7E"`.

`questionCol`
: character; the question text color as hex color or R color name. The default is `"white"`. This container also changes color to provide visual feedback (green for correct, red for wrong).

`questionBg`
: character; the question background color as hex color or R color name. The default is `"#5F5F5F"`.

`mainCol`
: character; the text color in the main content area (answer options). The default is `"#1C1C1C"`.

`mainBg`
: character; the background color of the main content area. The default is `"#F7F7F7"`.

`optionBg`
: character; the background color of the answer option rows. The hover color is automatically derived (30% darker). The default is `"#ECECEC"`.

`optionLabelBg`
: character; the background color of the A/B/C/D option labels. The hover color is automatically derived (20% darker). The default is `"#3498DB"`.

`navButtonCol`
: character; the navigation/submit button text color. The default is `"#FFFFFF"`.

`navButtonBg`
: character; the navigation/submit button background color. The default is `"#28A745"`.

`tipButtonCol`
: character; the tip button text color. The default is `"#1C1C1C"`.

`tipButtonBg`
: character; the tip button background color. The default is `"#F1F1F1"`.

`solutionButtonCol`
: character; the solution button text color. The default is `"#1C1C1C"`.

`solutionButtonBg`
: character; the solution button background color. The default is `"#F1F1F1"`.

`tipAreaCol`
: character; the text color of the tip text area. The default is `"#1C1C1C"`.

`tipAreaBg`
: character; the background color of the tip text area. The default is `"#E7F9E7"`.

`tipAreaBorder`
: character; the border color of the tip text area. The default is `"#28A745"`.

`solutionAreaCol`
: character; the text color of the solution text area. The default is `"#1C1C1C"`.

`solutionAreaBg`
: character; the background color of the solution text area. The default is `"#D6EAF8"`.

`solutionAreaBorder`
: character; the border color of the solution text area. The default is `"#3498DB"`.

`theme`
: an optional `rquizTheme` object. Theme values are used as defaults and are overridden by any explicitly passed arguments.

Details

In **single-choice mode**, clicking an answer option gives immediate visual feedback: green (correct) or red (wrong).

In **multiple-choice mode**, the user selects one or more options and clicks Submit. If the answer is wrong, a feedback message indicates how many correct and wrong selections were made. The Submit button is disabled after each attempt and re-enabled when the selection changes, allowing the user to adjust and retry. Optional Tip and Solution buttons provide additional help.

Text formatting: Since quiz content is rendered as HTML, you can use HTML tags in question and option strings: `<em>` for italics (e.g. species names), `<strong>` for bold, `<code>` for inline code, or `<span>` with inline CSS for colored text. Standard Markdown formatting does *not* work inside quiz strings.

Value

An HTML widget object of class `singleQuestion` that can be rendered in R Markdown/Quarto documents, Shiny applications, or the RStudio viewer.

Author(s)

Saskia A. Otto

See Also

`checkSingleQuestion` for input validation, `multiQuestions` for multi-page quizzes, `fillBlanks` for cloze quizzes, `rquizTheme` for reusable themes.

Examples

```
# --- SINGLE-CHOICE MODE ---

myQuestionSC <- list(
  question = "Which city hosted the 2024 Summer Olympics?",
  options = c("Athens", "London", "Paris", "Tokyo"),
  answer = 3  # (i.e. Paris)
)

### Check if list has the correct format
checkSingleQuestion(myQuestionSC)   # passes

### Default (includes solution button)
singleQuestion(x = myQuestionSC)

### With title and shuffled options
singleQuestion(x = myQuestionSC, title = "Sports Quiz", shuffle = TRUE)

### Without the solution button
singleQuestion(x = myQuestionSC, showSolutionButton = FALSE)

### With tip button (requires tip text in the question list)
myQuestionSCwithTip <- list(
```

```
  question = "Which city hosted the 2024 Summer Olympics?",
  options = c("Athens", "London", "Paris", "Tokyo"),
  answer = 3,
  tip = "Think about the Eiffel Tower."
)
singleQuestion(x = myQuestionSCwithTip, showTipButton = TRUE)

### Changing the window size
singleQuestion(x = myQuestionSC, width = "50%")
singleQuestion(x = myQuestionSC, width = "50%", center = FALSE)
# height works only with scroll = TRUE (useful for slide presentations):
singleQuestion(x = myQuestionSC, width = "50%",
  height = "300px", scroll = TRUE)

### Changing the language (UI text changes automatically)
singleQuestion(x = myQuestionSCwithTip, language = "de",
  showTipButton = TRUE)

### Customizing the design (dark theme):
singleQuestion(x = myQuestionSCwithTip,
  showTipButton = TRUE,
  title = "Sports Quiz",
  fontFamily = "Georgia, serif",
  fontSize = 20, titleFontSize = 14, titleAlign = "left",
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  mainCol = "#E0E0E0", mainBg = "#1A1B2E",
  optionBg = "#252540", optionLabelBg = "#0F3460",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560",
  tipButtonCol = "#E0E0E0", tipButtonBg = "#2C2C3E",
  solutionButtonCol = "#E0E0E0", solutionButtonBg = "#2C2C3E",
  tipAreaCol = "#E0E0E0", tipAreaBg = "#1A2A1A", tipAreaBorder = "#4CAF50",
  solutionAreaCol = "#E0E0E0", solutionAreaBg = "#1A1A2E",
  solutionAreaBorder = "#5DADE2")


# --- MULTIPLE-CHOICE MODE (auto-detected from answer vector) ---

myQuestionMC <- list(
  question = "Which of the following are European capital cities?",
  options = c("Berlin", "Sydney", "Paris", "Tokyo", "Rome"),
  answer = c(1, 3, 5)  # Berlin, Paris, Rome
)

### Default (auto-detected as MC because answer has 3 elements)
singleQuestion(x = myQuestionMC)

### With title and shuffled options
singleQuestion(x = myQuestionMC,
  title = "Geography Quiz", shuffle = TRUE)

### Without the solution button
singleQuestion(x = myQuestionMC, showSolutionButton = FALSE)
```

```
### With tip button (auto-generated: shows number of correct answers)
singleQuestion(x = myQuestionMC, showTipButton = TRUE)

### With custom tip text
myQuestionMCwithTip <- list(
  question = "Which of the following are European capital cities?",
  options = c("Berlin", "Sydney", "Paris", "Tokyo", "Rome"),
  answer = c(1, 3, 5),
  tip = "Three of the five cities are in Europe."
)
singleQuestion(x = myQuestionMCwithTip, showTipButton = TRUE)

### Changing the window size
singleQuestion(x = myQuestionMC, width = "50%")
singleQuestion(x = myQuestionMC, width = "50%",
  height = "450px", scroll = TRUE)

### Changing the language (button labels change automatically)
singleQuestion(x = myQuestionMC, showTipButton = TRUE, language = "de")
singleQuestion(x = myQuestionMC, showTipButton = TRUE, language = "fr")
singleQuestion(x = myQuestionMC, showTipButton = TRUE, language = "es")

### Customizing the design (dark theme)
singleQuestion(x = myQuestionMCwithTip,
  showTipButton = TRUE,
  title = "Geography Quiz",
  fontFamily = "Georgia, serif",
  fontSize = 20, titleFontSize = 14, titleAlign = "left",
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  mainCol = "#E0E0E0", mainBg = "#1A1B2E",
  optionBg = "#252540", optionLabelBg = "#0F3460",
  navButtonCol = "#FFFFFF", navButtonBg = "#E94560",
  tipButtonCol = "#E0E0E0", tipButtonBg = "#2C2C3E",
  solutionButtonCol = "#E0E0E0", solutionButtonBg = "#2C2C3E",
  tipAreaCol = "#E0E0E0", tipAreaBg = "#1A2A1A", tipAreaBorder = "#4CAF50",
  solutionAreaCol = "#E0E0E0", solutionAreaBg = "#1A1A2E",
  solutionAreaBorder = "#5DADE2")

### Using a reusable theme
dark <- rquizTheme(
  titleCol = "#E0E0E0", titleBg = "#1A1A2E",
  questionCol = "#FFFFFF", questionBg = "#16213E",
  navButtonBg = "#E94560"
)
singleQuestion(x = myQuestionMC, theme = dark, title = "Themed Quiz")
```

---

singleQuestion-shiny *Shiny bindings for singleQuestion*

Description

Output and render functions for using singleQuestion within Shiny applications and interactive Rmd documents.

Usage

```
singleQuestionOutput(outputId, width = "100%", height = "400px")

renderSingleQuestion(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a singleQuestion.
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

singleQuestionOutput returns an HTML output element for use in a Shiny UI definition. renderSingleQuestion returns a server-side rendering function to be assigned to an output slot.

Examples

```
if (interactive()) {
library(shiny)

ui <- fluidPage(
  singleQuestionOutput("quiz1")
)

server <- function(input, output) {
  output$quiz1 <- renderSingleQuestion({
    singleQuestion(
      x = list(
        question = "What is 2+2?",
        options = c("3", "4", "5"),
        answer = 2
      ),
      title = "Quick Quiz"
    )
  })
}

shinyApp(ui, server)
}
```

Index