Package: shinyquiz (via r-universe)

September 12, 2024

Title Create Interactive Quizzes in 'shiny'

Version 0.0.1.9000

Description Simple and flexible quizzes in 'shiny'. Easily create quizzes from various pre-built question and choice types or create your own using 'htmltools' and 'shiny' packages as building blocks. Integrates with larger 'shiny' applications. Ideal for non-web-developers such as educators, data scientists, and anyone who wants to assess responses interactively in a small form factor.

License MIT + file LICENSE

URL https://priism-center.github.io/shinyquiz/, https://github.com/priism-center/shinyquiz

BugReports https://github.com/priism-center/shinyquiz/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports cli (>= 3.6.1), fontawesome (>= 0.5.0), glue (>= 1.6.2), htmltools (>= 0.5.5), methods (>= 4.0.0), purrr (>= 1.0.1), reactable (>= 0.4.4), scales (>= 1.2.1), shiny (>= 1.7.4), shinyjs (>= 2.1.0), stringi (>= 1.7.12), stringr (>= 1.5.0)

Suggests knitr, pkgload, renv, rmarkdown, sortable (>= 0.5.0), testthat (>= 3.1.7)

Config/testthat/edition 3

Depends R (>= 4.1.0)

Repository https://priism-center.r-universe.dev

RemoteUrl https://github.com/priism-center/shinyquiz

RemoteRef HEAD

RemoteSha 123c5d0f497d9e68c27b6e7fe4fa5578f1091bda

Contents

`add_choice` *Add choices to a quiz question*

Description

Add a choice to a quiz question. Used in conjunction with `create_question()` to generate a question.

Usage

```
add_choice(text, correct = FALSE)

add_numeric(correct)

add_slider(min = 0, max = 1, default_position = 0.5, correct)

add_text(correct, exact = FALSE)
```

Arguments

`text`	Text of the choice answer
`correct`	Boolean denoting if this `choice` is correct; numeric for `slider` or `numeric`
`min`	the minimum value of the slider range
`max`	the maximum value of the slider range
`default_position`	the default value the slider should take
`exact`	Boolean denoting if the grader should use exact matching. If FALSE, the user's answer will be compared to the correct answer after trimming whitespace, converting to lower case, and normalizing diacritics. If you wish to use your own normalizing function, please see `create_question_raw()`.

Value

an object of class 'quizChoice'

an object of class 'quizChoiceNumeric'

an object of class 'quizChoiceSlider'

an object of class 'quizChoiceText'

Functions

- `add_choice()`: Create a discrete choice
- `add_numeric()`: Create a numeric choice
- `add_slider()`: Create a slider choice
- `add_text()`: Create a free text choice

Author(s)

Joseph Marlo

George Perrett

See Also

`create_question()`

Examples

```
add_choice('39')
add_choice('39', TRUE)
add_slider(0, 1, 0.5, 0.8)
add_text('Correct answer')

q <- create_question(
 'My question prompt',
 add_choice('39'),
 add_choice('41', TRUE)
)

q1_fuzzy <- create_question('My Label', add_text(correct = ' hEllo'))
q1_fuzzy@grader('Héllo ')
q1_exact <- create_question('My Label', add_text(correct = 'hEllo', exact = TRUE))
q1_exact@grader('Héllo ')
```

`create_question` *Create a quiz question*

Description

Create a single quiz question. Used in conjunction with `create_quiz()` to generate a quiz.

Usage

```
create_question(
  prompt,
  ...,
  type = c("auto", "single", "multiple"),
  input = c("auto", "select", "checkbox", "radio"),
  shuffle = FALSE,
  ns = shiny::NS("quiz")
)

create_question_raw(
  prompt,
  grader,
  correct_answer_pretty,
 user_answer_prettifier = function(user_input) paste0(user_input, collapse = ", ")
)
```

Arguments

`prompt`	Text of the question prompt. Can also be an HTML element such as `htmltools::div()`.
`...`	Objects of class 'quizChoice' generated from `add_choice()`, `add_numeric()`, `add_slider()`, or `add_text()`. Named options to `shiny::selectInput()` or `shiny::checkboxGroupInput()` can be passed as well.
`type`	One of c('auto', 'single', 'multiple'). Can the user select only one answer or multiple?
`input`	One of c('auto', 'select', 'checkbox'). Should the UI for a select input (`shiny::selectInput()`), checkbox (`shiny::checkboxGroupInput()`), or radio buttons (`shiny::radioButtons()`)?
`shuffle`	TRUE or FALSE. If TRUE order of choices will be randomly shuffled.
`ns`	Namespace function generated from `shiny::NS()`
`grader`	A function that takes the user answer and determines if it is correct. Must take one argument and return TRUE or FALSE. Note that this is wrapped with `purrr::possibly()` and `base::isTRUE()` to catch any errors.
`correct_answer_pretty`	A string representing the correct answer that is printed 'pretty'
`user_answer_prettifier`	A function with one argument that takes the user's answers and prints it 'pretty'

Details

`create_question` is the default method of creating quiz questions.

`create_question_raw()` allows any HTML in the `prompt`. This must contain a shiny input that is accessible via `input$answers`. The namespace also needs care. The default `inputId` is `shiny::NS('quiz')('answers')`.

Value

an object of class `quizQuestion`

an object of class `quizQuestion`

Functions

- `create_question()`: Create a quiz question
- `create_question_raw()`: Create a quiz question using custom inputs. This is a more flexible function that allows any html.

Author(s)

Joseph Marlo, George Perrett

Joseph Marlo

See Also

`add_choice()`, `add_slider()`, `add_numeric()`, `add_text()`, `create_question_raw()`

Examples

```
q <- create_question(
  prompt = 'My prompt explaining what the ATE of this thing should be',
  add_choice("34"),
  add_choice("59", TRUE),
  add_choice("98", TRUE)
)

q2 <- create_question(
  prompt = 'My prompt explaining what the ATC of this thing should be',
  add_slider(0, 30, 15, correct = 10)
 )
quiz <- create_quiz(q, q2)
q3 <- create_question_raw(
  prompt = htmltools::div(
    htmltools::p("my question"),
    shiny::selectInput(
      inputId = shiny::NS('quiz')('answers'),
      label = 'Select 5',
      choices = c(4, 5, 6)
    )
 ),
 grader = \(user_input) user_input == '5',
```

```
 correct_answer_pretty = '5'
)
quiz2 <- create_quiz(q3, q2)
```

`create_question_random` *Create a random question*

Description

Create questions with inherit randomness. Allows one function to generate many different questions.

Usage

```
create_question_random(.f, n = 50, verify_randomness = TRUE)
```

Arguments

`.f`	a function that outputs an object of class `quizQuestion`. This function can not have any arguments and must be able to produce random permutations of questions. The easiest way to ensure this is by including a `create_question()` or `create_question_raw()` call inside your function (see example).
`n`	a numeric value indicating how many draws of function .f to include in the random question bank.
`verify_randomness`	a boolean to denote if `.f` has inherit randomness. Defaults to TRUE.

Details

`create_question_random()` takes any user generated function `.f`. The function passed to the `.f` argument creates a random prompt along with an updated answer, the function passed to the `.f` argument must return an object of class `quizQuestion`. `create_question_random()` will automatically check to ensure the function passed to `.f` is in the appropriate format. The `n` argument controls how many random draws from the function passed to `.f` are included in the question bank for the quiz. Higher values of `n` allow more unique questions but extreme values of `n` may also lead to slower performance. To create a quiz with `n` randomly generated questions, `create_question_random()` can be passed as an argument to `create_quiz()`.

Value

a list of length n that includes objects of class `quizQuestionRandom`

Author(s)

George Perrett, Joseph Marlo

Examples

```
# a function that generates a random question
random_question <- function() {
  number <- round(rnorm(1, 30, 10), 0)
  rand_prompt <- paste('Is', number, 'an even number?')

  # using create_question inside the function helps to ensure correct class
  q <- create_question(
    prompt = rand_prompt,
    add_choice('Yes, it is even', correct = number %% 2 == 0),
    add_choice('No, it is odd', correct = number %% 2 != 0)
  )

  return(q)
}

# create a quiz with a question bank of 20 randomly generated questions
quiz <- create_quiz(
  create_question_random(.f = random_question, n = 20)
)
```

`create_quiz` *Create a quiz*

Description

Create a single quiz comprising of questions generated from `create_question()` and/or `create_question_raw()`.

Usage

```
create_quiz(..., options = set_quiz_options())
```

Arguments

`...`	objects of class 'quizQuestions'. See `create_question()`, `create_question_raw()`
`options`	a list of options generated from `set_quiz_options()`

Value

an object of class `quiz`

Author(s)

Joseph Marlo

See Also

`set_quiz_options()`, `create_question()`, `create_question_raw()`

Examples

```
quiz <- create_quiz(
  create_question(
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Select nulla.',
    add_choice('auctor'),
    add_choice('nulla', correct = TRUE)
  ),
  create_question(
    'Mauris congue aliquet dui, ut dapibus lorem porttitor sed. Select 600.',
    add_choice('600', correct = TRUE),
    add_choice('800')
  )
)
```

`preview_app` *Tools for previewing quizzes*

Description

Launch a viewer to preview the structure of the questions in a quiz.

Usage

```
preview_app(quiz, launch_browser = TRUE)
```

Arguments

`quiz`	an object of class 'quiz' to preview
`launch_browser`	launch in a web browser?

Value

Called for side effect

Functions

- `preview_app()`: Preview a quiz with full operability

Author(s)

Joseph Marlo

Examples

```
quiz <- create_quiz(
  create_question(
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Select nulla.',
    add_choice('auctor'),
    add_choice('nulla', correct = TRUE)
  ),
  create_question(
    'Mauris congue aliquet dui, ut dapibus lorem porttitor sed. Select 600.',
    add_choice('600', correct = TRUE),
    add_choice('800')
  )
)
preview_app(quiz)
```

---

quiz_ui *Run a quiz in 'shiny'*

Description

A 'shiny' module to implement a quiz. These are the core functions to implement the quiz with a 'shiny' application.

Usage

```
quiz_ui(quiz)

quiz_server(quiz)
```

Arguments

quiz	an object of class quiz. See create_quiz()

Value

a reactive object showing the current results of the quiz

Functions

- quiz_ui(): UI side function
- quiz_server(): Server side function

Author(s)

Joseph Marlo

See Also

create_quiz() preview_app()

Examples

```
quiz <- create_quiz(
  create_question(
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Select nulla.',
    add_choice('auctor'),
    add_choice('nulla', correct = TRUE)
  ),
  create_question(
    'Mauris congue aliquet dui, ut dapibus lorem porttitor sed. Select 600.',
    add_choice('600', correct = TRUE),
    add_choice('800')
  )
)

ui <- shiny::fluidPage(
  htmltools::div(
    style = "max-width: 700px",
    quiz_ui(quiz)
   )
 )
server <- function(input, output, session) {
 quiz_server(quiz)
}
shinyApp(ui, server)
```

---

set_quiz_options *Set the options for the quiz*

Description

These are options to be passed to a quiz.

Usage

```
set_quiz_options(
  ns = shiny::NS("quiz"),
  messages,
  sandbox = FALSE,
  end_on_first_wrong = !sandbox,
  class = NULL,
  progress_bar = !sandbox,
  progress_bar_color = "#609963",
  question_heading = "Practice what you've learned",
  include_question_title = TRUE,
```

```
  ...
)

create_messages(message_correct, message_wrong, message_skipped)
```

Arguments

ns	namespace generated from shiny::NS(). When using custom namespaces, the individual create_question() requires the namespace as well.
messages	an object of class quizMessages generated from create_messages() containing the messages to show at the end. If not provided, defaults are used.
sandbox	boolean. Quiz no longer ends of the first wrong, removes the progress bar, and grading does not include unattempted questions. Note that the presence of a random question automatically triggers sandbox mode. It can be overridden with set_quiz_options(override = TRUE).
end_on_first_wrong	Should the quiz immediately end once the user gets one question wrong?
class	string. A custom CSS class to add to the quiz div
progress_bar	boolean. Show the progress bar UI at the top of the quiz
progress_bar_color	Color code for the progress bar background
question_heading	Default text to show above the question prompt.
include_question_title	boolean. Should the text "Question #" be included above the prompt? If not included, the green check / red X will not be shown either.
...	other named options to pass to quiz
message_correct	a string to be shown at the end of the quiz when the user gets all questions correct
message_wrong	a string to be shown at the end of the quiz when the user gets at least one question wrong
message_skipped	a string to be shown at the end of the quiz when the user skips the quiz or ends it early

Value

a list

an object of class quizMessages

Functions

- set_quiz_options(): Sets the options for a quiz
- create_messages(): Create a messages object

Examples

```
# set the options when creating the quiz
quiz <- create_quiz(
  create_question(
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Select nulla.',
    add_choice('auctor'),
    add_choice('nulla', correct = TRUE)
  ),
  create_question(
    'Mauris congue aliquet dui, ut dapibus lorem porttitor sed. Select 600.',
    add_choice('600', correct = TRUE),
    add_choice('800')
  ),
  options = set_quiz_options(sandbox = TRUE)
)

# or modify the options on a quiz object
quiz@options <- set_quiz_options(sandbox = FALSE)

# adjust the messages shown at the end of the quiz
messages <- create_messages(
  'Congrats!',
  'Ahh, bummer! Got at least one wrong',
  'Looks like you skipped to the end!'
 )
quiz@options <- set_quiz_options(messages = messages)
```

Index