Package 'learnr'

February 13, 2020

Type Package

Title Interactive Tutorials for R

Version 0.10.1

Description Create interactive tutorials using R Markdown. Use a combination of narrative, figures, videos, exercises, and quizzes to create self-paced tutorials for learning about R and R packages.

License Apache License (>= 2.0)

URL https://rstudio.github.io/learnr/, https://github.com/rstudio/learnr

BugReports https://github.com/rstudio/learnr/issues

Imports utils, parallel, withr, rappdirs, rprojroot, jsonlite, htmltools (>= 0.3.5), htmlwidgets, evaluate, knitr (>= 1.14), markdown, shiny (>= 1.0), rmarkdown (>= 1.12.0), ellipsis (>= 0.2.0.1), checkmate, renv (>= 0.8.0)

Encoding UTF-8

LazyData true

RoxygenNote 7.0.2

Suggests testthat (>= 2.1.0), callr, rlang

VignetteBuilder knitr

NeedsCompilation no

Author Barret Schloerke [aut, cre] (<https://orcid.org/0000-0001-9986-114X>), JJ Allaire [aut], Barbara Borges [aut], Angela Li [ctb] (vignette), RStudio [cph, fnd], Ajax.org B.V. [ctb, cph] (Ace library), Zeno Rocha [ctb, cph] (clipboard.js library), Nick Payne [ctb, cph] (Bootbox library), Jake Archibald [ctb, cph] (idb-keyval library)

Maintainer Barret Schloerke <barret@rstudio.com>

Repository CRAN

Date/Publication 2020-02-13 22:20:02 UTC

R topics documented:

`available_tutorials` *Run a tutorial*

Description

Run a tutorial which is contained within an R package.

Usage

```
available_tutorials(package = NULL)
```

Arguments

`package`	Name of package

Details

Note that when running a tutorial Rmd file with `run_tutorial` the tutorial Rmd should have already been rendered as part of the development of the package (i.e. the corresponding tutorial .html file for the .Rmd file must exist).

Value

available_tutorials will return a data.frame containing "package", "name", "title", "description", "package_dependencies", "private", and "yaml_front_matter".

---

correct *Question is correct value*

Description

Helper method to return

Usage

```
correct(messages = NULL)

incorrect(messages = NULL)

mark_as(correct, messages = NULL)
```

Arguments

messages	a vector of messages to be displayed. The type of message will be determined by the correct value.
correct	boolean that determines if a question answer is correct

Examples

```
# Radio button question implementation of `question_is_correct`
question_is_correct.radio <- function(question, value, ...) {
  for (ans in question$answers) {
    if (as.character(ans$option) == value) {
      return(mark_as(
        ans$correct,
        ans$message
      ))
    }
  }
  mark_as(FALSE, NULL)
}
```

disable_all_tags *Disable all html tags*

Description

Method to disable all html tags to not allow users to interact with the html.

Usage

```
disable_all_tags(ele)
```

Arguments

ele	html tag element

Examples

```
# add an href to all a tags
disable_all_tags(
  htmltools::tagList(
    htmltools::a(),
    htmltools::a()
  )
)
```

duplicate_env *Create a duplicate of an environment*

Description

Copy all items from the environment to a new environment. By default, the new environment will share the same parent environment.

Usage

```
duplicate_env(envir, parent = parent.env(envir))
```

Arguments

envir	environment to duplicate
parent	parent environment to set for the new environment. Defaults to the parent environment of envir.

Examples

```
# Make a new environment with the object 'key'
envir <- new.env()
envir$key <- "value"
"key" %in% ls() # FALSE
"key" %in% ls(envir = envir) # TRUE

# Duplicate the envir and show it contains 'key'
new_envir <- duplicate_env(envir)
"key" %in% ls(envir = new_envir) # TRUE
```

`filesystem_storage` *Filesystem-based storage for tutor state data*

Description

Tutorial state storage handler that uses the filesystem as a backing store. The directory will contain tutorial state data partitioned by user_id, tutorial_id, and tutorial_version (in that order)

Usage

```
filesystem_storage(dir, compress = TRUE)
```

Arguments

`dir`	Directory to store state data within
`compress`	Should `.rds` files be compressed?

Value

Storage handler suitable for `options(tutorial.storage = ...)`

`format.tutorial_question_answer`
Formatting and printing quizzes, questions, and answers

Description

Notes:

- If custom question types are created, custom s3 formating methods may be implemented as well.
- Due to the shiny runtime of questions, a text representation of quizzes, questions, and answers will be presented.

Usage

```
## S3 method for class 'tutorial_question_answer'
format(x, ..., spacing = "")

## S3 method for class 'tutorial_question'
format(x, ..., spacing = "")

## S3 method for class 'tutorial_quiz'
format(x, ...)

## S3 method for class 'tutorial_question'
print(x, ...)

## S3 method for class 'tutorial_question_answer'
print(x, ...)

## S3 method for class 'tutorial_quiz'
print(x, ...)
```

Arguments

x	object of interest
...	ignored
spacing	Text to be placed at the beginning of each new line

See Also

quiz, question, answer

Examples

```
ex_question <- question("What number is the letter A in the alphabet?",
  answer("8"),
  answer("14"),
  answer("1", correct = TRUE),
  answer("23"),
  incorrect = "See [here](https://en.wikipedia.org/wiki/English_alphabet) and try again.",
  allow_retry = TRUE
)
cat(format(ex_question), "\n")
```

---

initialize_tutorial *Initialize Tutorial R Markdown Extensions*

Description

One time initialization of R Markdown extensions required by the **tutor** package. This function is typically called automatically as a result of using exercises or questions.

Usage

```
initialize_tutorial()
```

---

knit_print.tutorial_question *Knitr quiz print methods*

Description

knitr::knit_print methods for question and quiz

Usage

```
## S3 method for class 'tutorial_question'
knit_print(x, ...)

## S3 method for class 'tutorial_quiz'
knit_print(x, ...)
```

Arguments

x	An R object to be printed
...	Additional arguments passed to the S3 method. Currently ignored, except two optional arguments options and inline; see the references below.

---

question_checkbox *Checkbox question*

Description

Creates a checkbox group tutorial quiz question. The student may select one or more checkboxes before submitting their answer.

Usage

```
question_checkbox(
  text,
  ...,
  correct = "Correct!",
  incorrect = "Incorrect",
  try_again = incorrect,
  allow_retry = FALSE,
  random_answer_order = FALSE
)
```

Arguments

text	Question or option text
...	answers and extra parameters passed onto question.
correct	For question, text to print for a correct answer (defaults to "Correct!"). For answer, a boolean indicating whether this answer is correct.
incorrect	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is FALSE.
try_again	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is TRUE.
allow_retry	Allow retry for incorrect answers. Defaults to FALSE.
random_answer_order	Display answers in a random order.

See Also

question_radio, question_text

Examples

```
question_checkbox(
  "Select all the toppings that belong on a Margherita Pizza:",
  answer("tomato", correct = TRUE),
  answer("mozzarella", correct = TRUE),
  answer("basil", correct = TRUE),
  answer("extra virgin olive oil", correct = TRUE),
  answer("pepperoni", message = "Great topping! ... just not on a Margherita Pizza"),
  answer("onions"),
  answer("bacon"),
  answer("spinach"),
  random_answer_order = TRUE,
  allow_retry = TRUE,
  try_again = "Be sure to select all four toppings!"
)
```

---

question_radio *Radio question*

Description

Creates a radio button tutorial quiz question. The student can select only one radio button before submitting their answer.

Usage

```
question_radio(
  text,
  ...,
  correct = "Correct!",
  incorrect = "Incorrect",
  try_again = incorrect,
  allow_retry = FALSE,
  random_answer_order = FALSE
)
```

Arguments

text	Question or option text
...	answers and extra parameters passed onto question.
correct	For question, text to print for a correct answer (defaults to "Correct!"). For answer, a boolean indicating whether this answer is correct.
incorrect	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is FALSE.
try_again	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is TRUE.
allow_retry	Allow retry for incorrect answers. Defaults to FALSE.
random_answer_order	Display answers in a random order.

Details

Note: Multiple correct answers are allowed.

See Also

question_checkbox, question_text

Examples

```
question_radio(
  "Pick the letter B",
  answer("A"),
  answer("B", correct = TRUE),
  answer("C"),
  answer("D"),
  allow_retry = TRUE,
  random_answer_order = TRUE
)
```

question_text *Text box question*

Description

Creates a text box group tutorial quiz question.

Usage

```
question_text(
  text,
  ...,
  correct = "Correct!",
  incorrect = "Incorrect",
  try_again = incorrect,
  allow_retry = FALSE,
  random_answer_order = FALSE,
  placeholder = "Enter answer here...",
  trim = TRUE,
  options = list()
)
```

Arguments

text	Question or option text
...	answers and extra parameters passed onto question.
correct	For question, text to print for a correct answer (defaults to "Correct!"). For answer, a boolean indicating whether this answer is correct.
incorrect	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is FALSE.
try_again	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is TRUE.
allow_retry	Allow retry for incorrect answers. Defaults to FALSE.
random_answer_order	Display answers in a random order.
placeholder	A character string giving the user a hint as to what can be entered into the control. Internet Explorer 8 and 9 do not support this option.
trim	Logical to determine if whitespace before and after the answer should be removed. Defaults to TRUE.
options	Extra options to be stored in the question object.

See Also

question_radio, question_checkbox

Examples

```
question_text(
  "Please enter the word 'C0rrect' below:",
  answer("correct", message = "Don't forget to capitalize"),
  answer("c0rrect", message = "Don't forget to capitalize"),
  answer("Correct", message = "Is it really an 'o'?"),
  answer("C0rrect ", message = "Make sure you do not have a trailing space"),
  answer("C0rrect", correct = TRUE),
  allow_retry = TRUE,
  trim = FALSE
)
```

`question_ui_initialize` *Custom question methods.*

Description

There are five methods used to define a custom question. Each S3 method should correspond to the `type = TYPE` supplied to the question.

- `question_ui_initialize.TYPE(question,value,...)`
  - Determines how the question is initially displayed to the users. This should return a shiny UI object that can be displayed using shiny::renderUI. For example, in the case of `question_ui_initialize.radio`, it returns a shiny::radioButtons object. This method will be re-executed if the question is attempted again.
- `question_ui_completed.TYPE(question,...)`
  - Determines how the question is displayed after a submission. Just like `question_ui_initialize`, this method should return an shiny UI object that can be displayed using shiny::renderUI.
- `question_is_valid.TYPE(question,value,...)`
  - This method should return a boolean that determines if the input answer is valid. Depending on the value, this function enables and disables the submission button.
- `question_is_correct.TYPE(question,value,...)`
  - This function should return the output of correct, incorrect, or mark_as. Each method allows for custom messages in addition to the determination of an answer being correct. See correct, incorrect, or mark_as for more details.
- question_ui_try_again <- function(question, value, ...)
  - Determines how the question is displayed to the users while the "Try again" screen is displayed. Usually this function will disable inputs to the question, i.e. prevent the student from changing the answer options. Similar to `question_ui_initialize`, this should should return a shiny UI object that can be displayed using shiny::renderUI.

Usage

```
question_ui_initialize(question, value, ...)

question_ui_try_again(question, value, ...)

question_ui_completed(question, value, ...)

question_is_valid(question, value, ...)

question_is_correct(question, value, ...)

## Default S3 method:
question_ui_initialize(question, value, ...)

## Default S3 method:
question_ui_try_again(question, value, ...)

## Default S3 method:
question_ui_completed(question, value, ...)

## Default S3 method:
question_is_valid(question, value, ...)

## Default S3 method:
question_is_correct(question, value, ...)
```

Arguments

question	question object used
value	user input value
...	future parameter expansion and custom arguments to be used in dispatched s3 methods.

See Also

For more information and question type extension examples, please view the question_type tutorial: learnr::run_tutorial("question_type","learnr").

---

quiz *Tutorial quiz questions*

Description

Add interactive quiz questions to a tutorial. Each quiz question is executed within a shiny runtime to provide more flexibility in the types of questions offered. There are three default types of quiz questions:

`learnr_radio` Radio button question. This question type will only allow for a single answer submission by the user. An answer must be marked for the user to submit their answer.

`learnr_checkbox` Check box question. This question type will allow for one or more answers to be submitted by the user. At least one answer must be marked for the user to submit their answer.

`learnr_text` Text box question. This question type will allow for free form text to be submitted by the user. At least one non-whitespace character must be added for the user to submit their answer.

Usage

```
quiz(..., caption = "Quiz")

question(
  text,
  ...,
  type = c("auto", "single", "multiple", "learnr_radio", "learnr_checkbox",
    "learnr_text"),
  correct = "Correct!",
  incorrect = "Incorrect",
  try_again = incorrect,
  message = NULL,
  post_message = NULL,
  loading = c("**Loading:** ", text, "<br/><br/><br/>"),
  submit_button = "Submit Answer",
  try_again_button = "Try Again",
  allow_retry = FALSE,
  random_answer_order = FALSE,
  options = list()
)

answer(text, correct = FALSE, message = NULL)
```

Arguments

`...`	One or more questions or answers
`caption`	Optional quiz caption (defaults to "Quiz")
`text`	Question or option text
`type`	Type of quiz question. Typically this can be automatically determined based on the provided answers. Pass `"radio"` to indicate that even though multiple correct answers are specified that inputs which include only one correct answer are still correct. Pass `"checkbox"` to force the use of checkboxes (as opposed to radio buttons) even though only once correct answer was provided.
`correct`	For `question`, text to print for a correct answer (defaults to "Correct!"). For `answer`, a boolean indicating whether this answer is correct.
`incorrect`	Text to print for an incorrect answer (defaults to "Incorrect") when `allow_retry` is `FALSE`.

try_again	Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is TRUE.
message	Additional message to display along with correct/incorrect feedback. This message is always displayed after a question submission.
post_message	Additional message to display along with correct/incorrect feedback. If allow_retry is TRUE, this message will only be displayed after the correct submission. If allow_retry is FALSE, it will produce a second message alongside the message message value.
loading	Loading text to display as a placeholder while the question is loaded
submit_button	Label for the submit button. Defaults to "Submit Answer"
try_again_button	Label for the try again button. Defaults to "Submit Answer"
allow_retry	Allow retry for incorrect answers. Defaults to FALSE.
random_answer_order	Display answers in a random order.
options	Extra options to be stored in the question object.

Details

Note, the print behavior has changed as the runtime is now Shiny based. If questions and quizes are printed in the console, the S3 structure and information will be displayed.

See Also

For more information and question type extension examples, please see the help documentation for question_methods and view the question_type tutorial: learnr::run_tutorial("question_type","learnr").

random_praise, random_encouragement

Examples

```
quiz(
  question("What number is the letter A in the alphabet?",
    answer("8"),
    answer("14"),
    answer("1", correct = TRUE),
    answer("23"),
   incorrect = "See [here](https://en.wikipedia.org/wiki/English_alphabet) and try again.",
    allow_retry = TRUE
  ),

  question("Where are you right now? (select ALL that apply)",
    answer("Planet Earth", correct = TRUE),
    answer("Pluto"),
    answer("At a computing device", correct = TRUE),
    answer("In the Milky Way", correct = TRUE),
    incorrect = paste0("Incorrect. You're on Earth, ",
                       "in the Milky Way, at a computer.")
  )
```

```
)
```

`random_praise` *Random praise and encouragement*

Description

Random praises and encouragements sayings to compliment your question and quiz experience.

Usage

```
random_praise()

random_encouragement()
```

Value

Character string with a random saying

`run_tutorial` *Run a tutorial*

Description

Run a tutorial which is contained within an R package.

Usage

```
run_tutorial(name = NULL, package = NULL, shiny_args = NULL)
```

Arguments

`name`	Tutorial name (subdirectory within `tutorials/` directory of installed package).
`package`	Name of package
`shiny_args`	Additional arguments to forward to `shiny::runApp`.

Details

Note that when running a tutorial Rmd file with `run_tutorial` the tutorial Rmd should have already been rendered as part of the development of the package (i.e. the corresponding tutorial .html file for the .Rmd file must exist).

See Also

`safe` and `available_tutorials`

Examples

```
# display all "learnr" tutorials
available_tutorials("learnr")

# run basic example within learnr
## Not run: run_tutorial("hello", "learnr")
```

`safe` *Execute R code in a safe R environment*

Description

When rendering (or running) a document with R markdown, it inherits the current R Global environment. This will produce unexpected behaviors, such as poisoning the R Global environment with existing variables. By rendering the document in a new, safe R environment, a *vanilla*, rendered document is produced.

Usage

```
safe(expr, ..., show = TRUE, env = safe_env())
```

Arguments

`expr`	expression that contains all the necessary library calls to execute. Expressions within callr do not inherit the existing, loaded libraries.
`...`	parameters passed to `callr::r`
`show`	Logical that determines if output should be displayed
`env`	Environment to evaluate the document in

Details

The environment variable `LEARNR_INTERACTIVE` will be set to `"1"` or `"0"` depending on if the calling session is interactive or not.

Using `safe` should only be necessary when locally deployed.

Examples

```
## Not run:
# Direct usage
safe(run_tutorial("hello", package = "learnr"))

# Programmatic usage
library(rlang)

expr <- quote(run_tutorial("hello", package = "learnr"))
safe(!!expr)
```

```
tutorial <- "hello"
safe(run_tutorial(!!tutorial, package = "learnr"))

## End(Not run)
```

`safe_env` *Safe R CMD environment*

Description

By default, `callr::rcmd_safe_env` suppresses the ability to open a browser window. This is the default execution environment within `callr::r`. However, opening a browser is expected behavior within the learnr package and should not be suppressed.

Usage

```
safe_env()
```

`tutorial` *Tutorial document format*

Description

Long-form tutorial which includes narrative, figures, videos, exercises, and questions.

Usage

```
tutorial(
  fig_width = 6.5,
  fig_height = 4,
  fig_retina = 2,
  fig_caption = TRUE,
  progressive = FALSE,
  allow_skip = FALSE,
  dev = "png",
  df_print = "paged",
  smart = TRUE,
  theme = "rstudio",
  highlight = "textmate",
  ace_theme = "textmate",
  mathjax = "default",
  extra_dependencies = NULL,
  css = NULL,
  includes = NULL,
  md_extensions = NULL,
  pandoc_args = NULL,
  ...
)
```

Arguments

fig_width	Default width (in inches) for figures
fig_height	Default height (in inches) for figures
fig_retina	Scaling to perform for retina displays (defaults to 2, which currently works for all widely used retina displays). Set to NULL to prevent retina scaling. Note that this will always be NULL when keep_md is specified (this is because fig_retina relies on outputting HTML directly into the markdown document).
fig_caption	TRUE to render figures with captions
progressive	Display sub-topics progressively (i.e. wait until previous topics are either completed or skipped before displaying subsequent topics).
allow_skip	Allow users to skip sub-topics (especially useful when progressive is TRUE).
dev	Graphics device to use for figure output (defaults to png)
df_print	Method to be used for printing data frames. Valid values include "default", "kable", "tibble", and "paged". The "default" method uses a corresponding S3 method of print, typically print.data.frame. The "kable" method uses the knitr::kable function. The "tibble" method uses the **tibble** package to print a summary of the data frame. The "paged" method creates a paginated HTML table (note that this method is only valid for formats that produce HTML). In addition to the named methods you can also pass an arbitrary function to be used for printing data frames. You can disable the df_print behavior entirely by setting the option rmarkdown.df_print to FALSE.
smart	Produce typographically correct output, converting straight quotes to curly quotes, --- to em-dashes, -- to en-dashes, and ... to ellipses.
theme	Visual theme ("rstudio", default", "cerulean", "journal", "flatly", "readable", "spacelab", "united", "cosmo", "lumen", "paper", "sandstone", "simplex", or "yeti").
highlight	Syntax highlighting style. Supported styles include "default", "tango", "pygments", "kate", "monochrome", "espresso", "zenburn", "haddock", and "textmate". Pass 'NULL' to prevent syntax highlighting. Note, this value only pertains to standard rmarkdown code, not the Ace editor highlighting.
ace_theme	Ace theme supplied to the ace code editor for all exercises. See learnr:::ACE_THEMES for a list of possible values. Defaults to "textmate".
mathjax	Include mathjax. The "default" option uses an https URL from a MathJax CDN. The "local" option uses a local version of MathJax (which is copied into the output directory). You can pass an alternate URL or pass NULL to exclude MathJax entirely.
extra_dependencies	Additional function arguments to pass to the base R Markdown HTML output formatter html_document_base
css	One or more css files to include
includes	Named list of additional content to include within the document (typically created using the includes function).
md_extensions	Markdown extensions to be added or removed from the default definition or R Markdown. See the rmarkdown_format for additional details.

pandoc_args	Additional command line options to pass to pandoc
...	Forward parameters to html_document

---

tutorial_html_dependency *Tutorial HTML dependency*

Description

Tutorial HTML dependency

Usage

```
tutorial_html_dependency()
```

Details

HTML dependency for core tutorial JS and CSS. This should be included as a dependency for custom tutorial formats that wish to ensure that that tutorial.js and tutorial.css are loaded prior their own scripts and stylesheets.

---

tutorial_options *Set tutorial options*

Description

Set various tutorial options that control the display and evaluation of exercises.

Usage

```
tutorial_options(
  exercise.cap = "Code",
  exercise.eval = FALSE,
  exercise.timelimit = 30,
  exercise.lines = NULL,
  exercise.checker = NULL,
  exercise.completion = TRUE,
  exercise.diagnostics = TRUE,
  exercise.startover = TRUE
)
```

Arguments

exercise.cap	Caption for exercise chunk (defaults to "Code").
exercise.eval	Whether to pre-evaluate the exercise so the reader can see some default output (defaults to FALSE).
exercise.timelimit	Number of seconds to limit execution time to (defaults to 30).
exercise.lines	Lines of code for exercise editor (defaults to the number of lines in the code chunk).
exercise.checker	Function used to check exercise answers.
exercise.completion	Use code completion in exercise editors.
exercise.diagnostics	Show diagnostics in exercise editors.
exercise.startover	Show "Start Over" button on exercise.

---

tutorial_package_dependencies *List Tutorial Dependencies*

Description

List the R packages required to run a particular tutorial.

Usage

```
tutorial_package_dependencies(name = NULL, package = NULL)
```

Arguments

name	The tutorial name. If name is NULL, then all tutorials within package will be searched.
package	The R package providing the tutorial. If package is NULL, then all tutorials will be searched.

Value

A character vector of package names that are required for execution.

Examples

```
tutorial_package_dependencies(package = "learnr")
```

Index