

Package: shinysurveys (via r-universe)

September 12, 2024

Title Create and Deploy Surveys in 'Shiny'

Version 0.2.0.9000

Description Easily create and deploy surveys in 'Shiny'. This package includes a minimalistic framework similar to 'Google Forms' that allows for url-based user tracking, customizable submit actions, easy survey-theming, and more.

License MIT + file LICENSE

URL <https://shinysurveys.jdtrat.com>,
<https://github.com/jdtrat/shinysurveys>

BugReports <https://github.com/jdtrat/shinysurveys/issues>

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.2

Imports shiny, sass, htmltools, jsonlite

Depends R (>= 2.10)

Suggests knitr, rmarkdown, tibble

VignetteBuilder knitr

Repository <https://jdtrat.r-universe.dev>

RemoteUrl <https://github.com/jdtrat/shinysurveys>

RemoteRef HEAD

RemoteSha aa4d393b99bc86345f81b5c5b4d10377e60c159f

Contents

demo_survey	2
demo_survey_multipage	3
extendInputType	3
getSurveyData	5

hideSurvey	7
listInputExtensions	8
numberInput	9
radioMatrixInput	10
renderSurvey	12
surveyID	13
surveyLabel	14
surveyOptions	15
surveyOutput	15
teaching_r_questions	17

Index	18
--------------	-----------

demo_survey	<i>Demo Survey</i>
-------------	--------------------

Description

This function runs a Shiny app that shows an example of running a demographic survey in Shiny. It has a sample title and description and its theme color can be customized using a hex color code.

Usage

```
demo_survey(theme = "#63B8FF")
```

Arguments

theme	A valid hex color such as #63B8FF (default)
-------	---

Value

A Shiny App

Examples

```
if (interactive()) demo_survey()
```

demo_survey_multipage *Demo Survey over Multiple Pages*

Description

This function runs a Shiny app that shows an example of running a demographic survey in Shiny, spanning multiple pages. It has a sample title and description and its theme color can be customized using a hex color code.

Usage

```
demo_survey_multipage(theme = "#63B8FF")
```

Arguments

theme A valid hex color such as #63B8FF (default)

Value

A Shiny App

Examples

```
if (interactive()) demo_survey_multipage()
```

extendInputType *Add Custom Input Types for a Survey*

Description

Add Custom Input Types for a Survey

Usage

```
extendInputType(input_type, extension)
```

Arguments

input_type A string of the input type supplied in the data frame of questions.
extension A shiny input type not natively supported by shinysurveys. See the examples section for more information.

Value

NA; used to register custom input types for use with a shiny survey.

See Also

[surveyID](#)
[surveyLabel](#)

Examples

```
# Register a slider input to {shinysurveys} with a custom minimum and maximum value.

extendInputType("slider", {
  shiny::sliderInput(
    inputId = surveyID(),
    label = surveyLabel(),
    min = 1,
    max = 10,
    value = 5
  )
})

# Define a question as normal with the `input_type` set to the custom slider type defined above.
slider_question <- data.frame(question = "On a scale from 1-10,
how much do you love sushi?",
option = NA,
input_type = "slider",
input_id = "sushi_scale",
dependence = NA,
dependence_value = NA,
required = TRUE)

# Watch it in action
if (interactive()) {
  ui <- fluidPage(
    surveyOutput(df = slider_question, "Sushi Scale Example")
  )

  server <- function(input, output, session) {
    renderSurvey()
  }

  shinyApp(ui, server)
}

# Register a date input to {shinysurveys},
# limiting possible dates to a twenty-day period.

extendInputType("date", {
  shiny::dateInput(
    inputId = surveyID(),
    value = Sys.Date(),
    label = surveyLabel(),
```

```
      min = Sys.Date()-10,
      max = Sys.Date()+10
    )
  })

# Define a question as normal with the `input_type` set to
# the custom date type defined above.

date_question <- data.frame(question = "When do you graduate?",
  option = NA,
  input_type = "date",
  input_id = "grad_date",
  dependence = NA,
  dependence_value = NA,
  required = FALSE)

# Watch it in action
if (interactive()) {
  ui <- fluidPage(
    surveyOutput(df = date_question, "Date Input Extension Example")
  )

  server <- function(input, output, session) {
    renderSurvey()
  }

  shinyApp(ui, server)
}

# Combine both custom input types:

if (interactive()) {
  ui <- fluidPage(
    surveyOutput(df = rbind(slider_question, date_question),
      "Date & Slider Input Extension Example")
  )

  server <- function(input, output, session) {
    renderSurvey()
  }

  shinyApp(ui, server)
}
```

Description

Get a participant's responses.

Usage

```
getSurveyData(
  custom_id = NULL,
  include_dependencies = TRUE,
  dependency_string = "HIDDEN-QUESTION"
)
```

Arguments

`custom_id` A unique identifier for the survey's respondents. `NULL` by default, and the built-in `shinysurveys` `userID` will be used.

`include_dependencies` LOGICAL: `TRUE` (default) and all dependency questions will be returned, regardless of if the individual respondent saw it. For respondents who did not see a specific question, the 'response' will take on the value from the `dependency_string` argument. If `FALSE`, the output will have variable rows depending on which questions a given participant answered.

`dependency_string` A character string to be imputed for dependency questions that a respondent did not see. Default is "HIDDEN-QUESTION".

Value

A data frame with four columns containing information about the participant's survey responses: The 'subject_id' column can be used for identifying respondents. By default, it utilizes `shinysurveys` URL-based user tracking feature. The 'question_id' and 'question_type' columns correspond to 'input_id' and 'input_type' from the original data frame of questions. The 'response' column is the participant's answer.

The number of rows, corresponding to the questions an individual saw, depends on the `include_dependencies` argument. If `TRUE`, by default, then the resulting data frame will have one row per unique input ID. If `FALSE`, the data frame may have variable length depending on which questions a given individual answers.

Examples

```
if (interactive()) {

  library(shiny)

  ui <- fluidPage(
    surveyOutput(teaching_r_questions)
  )

  server <- function(input, output, session) {
    renderSurvey()
  }
}
```

```
# Upon submission, print a data frame with participant responses
observeEvent(input$submit, {
  print(getSurveyData())
})
}

shinyApp(ui, server)

}
```

hideSurvey

Hide shinysurvey

Description

This function allows you to easily hide the survey, something you may wish to do upon submission.

Usage

```
hideSurvey()
```

Value

NA; used to hide the survey.

Examples

```
if (interactive()) {

  library(shiny)
  library(shinysurveys)

  ui <- fluidPage(
    surveyOutput(teaching_r_questions,
      survey_title = "Now you see me...",
      survey_description = "A demo showing how to hide the survey body upon submission.")
  )

  server <- function(input, output, session) {
    renderSurvey()
    observeEvent(input$submit, hideSurvey())
  }

  shinyApp(ui, server)

}
```

listInputExtensions *List all registered survey extensions*

Description

List all registered survey extensions

Usage

```
listInputExtensions()
```

Value

A named list containing the registered input type and their associated functions.

Examples

```
if (interactive()) {  
  
  # Register a date input to {shinysurveys},  
  # limiting possible dates to a twenty-day period.  
  
  extendInputType("slider", {  
    shiny::sliderInput(  
      inputId = surveyID(),  
      label = surveyLabel(),  
      min = 1,  
      max = 10,  
      value = 5  
    )  
  })  
  
  # Register a slider input to {shinysurveys}  
  # with a custom minimum and maximum value.  
  
  extendInputType("date", {  
    shiny::dateInput(  
      inputId = surveyID(),  
      value = Sys.Date(),  
      label = surveyLabel(),  
      min = Sys.Date()-10,  
      max = Sys.Date()+10  
    )  
  })  
  
  listInputExtensions()  
  
}
```

numberInput	<i>Create a numeric input</i>
-------------	-------------------------------

Description

Create an input control for entry of numeric values. This is identical to `shiny::numericInput()` but is more flexible in **not** requiring an initial value and in allowing placeholders.

Usage

```
numberInput(  
  inputId,  
  label,  
  value = NULL,  
  min = NA,  
  max = NA,  
  step = NA,  
  placeholder = NULL,  
  width = NULL  
)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control, or NULL for no label.
value	Initial value. NULL by default.
min	Minimum allowed value
max	Maximum allowed value
step	Interval to use when stepping between min and max
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.
width	The width of the input, e.g. '400px', or '100%'; see <code>validateCssUnit()</code> .

Value

A numeric input control that can be added to a UI definition.

Server value

A numeric vector of length 1.

See Also

[shiny::updateNumericInput\(\)](#)

Examples

```

if (interactive()) {
  library(shiny)
  library(shinysurveys)

  ui <- fluidPage(
    numberInput("obs", "Observations:", placeholder = "How many do you see?", min = 1, max = 100),
    verbatimTextOutput("value")
  )
  server <- function(input, output) {
    output$value <- renderText({ input$obs })
  }
  shinyApp(ui, server)
}

```

radioMatrixInput	<i>Create a matrix of radio buttons.</i>
------------------	--

Description

Create a matrix of radio buttons.

Usage

```
radioMatrixInput(inputId, responseItems, choices, selected = NULL, ...)
```

Arguments

inputId	The input id
responseItems	The questions to be asked (row labels)
choices	Possible choices (column labels)
selected	Initial selected value
...	Additional arguments specific to shinysurveys required questions.

Value

A matrix of radio buttons that can be added to a UI definition. When run in a Shiny application, this will return NULL until all possible response items have been answered, at which time a data frame with the question_id, question_type, and response, the format used in [getSurveyData](#).

Examples

```

# For use as a normal Shiny input:

if (interactive()) {

  library(shiny)

  ui <- fluidPage(
    radioMatrixInput("matInput",
                     responseItems = c("Love sushi?", "Love chocolate?"),
                     choices = c("Disagree", "Neutral", "Agree"))
  )

  server <- function(input, output, session) {
    observe({
      print(input$matInput)
    })
  }

  shinyApp(ui, server)
}

# For use in {shinysurveys}

if (interactive()) {

  df <- data.frame(
    question = c(rep("I love sushi.", 3), rep("I love chocolate.",3),
                 "What's your favorite food?", rep("Goat cheese is the GOAT.", 5),
                 rep("Yogurt and berries are a great snack.",5),
                 rep("SunButter® is a fantastic alternative to peanut butter.", 5)),
    option = c(rep(c("Disagree", "Neutral", "Agree"), 2), "text",
               rep(c("Strongly Disagree", "Disagree", "Neutral", "Agree", "Strongly Agree"), 3)),
    input_type = c(rep("matrix", 6), "text", rep("matrix", 15)),
    # For matrix questions, the IDs should be the same for each question
    # but different for each matrix input unit
    input_id = c(rep("matId", 6), "favorite_food", rep("matId2", 15)),
    dependence = NA,
    dependence_value = NA,
    required = FALSE
  )

  library(shiny)

  ui <- fluidPage(
    surveyOutput(df)
  )

  server <- function(input, output, session) {
    renderSurvey()
    observe({

```

```
    print(input$matId)
    print(input$favorite_food)
    print(input$matId2)
  })
}

shinyApp(ui, server)

}
```

renderSurvey

Server code for adding survey questions

Description

Include server-side logic for shinysurveys.

Usage

```
renderSurvey(df, theme = "#63B8FF")
```

Arguments

df	Deprecated <i>please only place argument in surveyOutput</i> . A user supplied data frame in the format of <code>teaching_r_questions</code> .
theme	Deprecated <i>please place the theme argument in surveyOutput</i> . A valid R color: predefined such as "red" or "blue"; hex colors such as #63B8FF (default). To customize the survey's appearance entirely, supply NULL.

Value

NA; used for server-side logic in Shiny apps.

Examples

```
if (interactive()) {

  library(shiny)
  library(shinysurveys)

  df <- data.frame(question = "What is your favorite food?",
                  option = "Your Answer",
                  input_type = "text",
                  input_id = "favorite_food",
                  dependence = NA,
                  dependence_value = NA,
                  required = F)
```

```
ui <- fluidPage(
  surveyOutput(df = df,
               survey_title = "Hello, World!",
               theme = "#63B8FF")
)

server <- function(input, output, session) {
  renderSurvey()

  observeEvent(input$submit, {
    showModal(modalDialog(
      title = "Congrats, you completed your first shinysurvey!",
      "You can customize what actions happen when a user finishes a survey using input$submit."
    ))
  })
}

shinyApp(ui, server)
}
```

surveyID

Add correct ID for custom input types

Description

`surveyID()` is a helper function for [extendInputType](#). When defining custom input types, the `inputId` argument for shiny UI components should equal `surveyID()`. See examples for more details.

Usage

```
surveyID()
```

Value

NA; used for side effects with [extendInputType](#).

See Also

[extendInputType](#)

[surveyLabel](#)

[surveyOptions](#)

Examples

```
extendInputType("slider", {
  shiny::sliderInput(
    inputId = surveyID(),
    label = surveyLabel(),
    min = 1,
    max = 10,
    value = 5
  )
})
```

surveyLabel

Add correct label for custom input types

Description

`surveyLabel()` is a helper function for [extendInputType](#). When defining custom input types, the `label` argument for shiny UI components should equal `surveyLabel()`. It essentially takes on the value in the "question" column in the data supplied to [surveyOutput](#). See examples for more details.

Usage

```
surveyLabel()
```

Value

NA; used for side effects with [extendInputType](#).

See Also

[extendInputType](#)

[surveyID](#)

[surveyOptions](#)

Examples

```
extendInputType("slider", {
  shiny::sliderInput(
    inputId = surveyID(),
    label = surveyLabel(),
    min = 1,
    max = 10,
    value = 5
  )
})
```

surveyOptions	<i>Add options for custom input types</i>
---------------	---

Description

surveyOptions() is a helper function for [extendInputType](#). When defining custom input types, the choices arguments for shiny UI components should equal surveyOption(). See examples for more details.

Usage

```
surveyOptions()
```

Value

NA; used for side effects with [extendInputType](#).

See Also

[extendInputType](#)

[surveyID](#)

[surveyOptions](#)

Examples

```
extendInputType("inlineRadioButtons", {
  shiny::radioButtons(
    inputId = surveyID(),
    label = surveyLabel(),
    selected = character(0),
    choices = surveyOptions(),
    inline = TRUE
  )
})
```

surveyOutput	<i>Generate the UI Code for demographic questions</i>
--------------	---

Description

Create the UI code for a Shiny app based on user-supplied questions.

Usage

```
surveyOutput(df, survey_title, survey_description, theme = "#63B8FF", ...)
```

Arguments

<code>df</code>	A user supplied data frame in the format of <code>teaching_r_questions</code> .
<code>survey_title</code>	(Optional) user supplied title for the survey
<code>survey_description</code>	(Optional) user supplied description for the survey
<code>theme</code>	A valid R color: predefined such as "red" or "blue"; hex colors such as #63B8FF (default). To customize the survey's appearance entirely, supply NULL.
<code>...</code>	Additional arguments to pass into <code>actionButton</code> used to submit survey responses.

Value

UI Code for a Shiny App.

Examples

```

if (interactive()) {

  library(shiny)
  library(shinysurveys)

  df <- data.frame(question = "What is your favorite food?",
                  option = "Your Answer",
                  input_type = "text",
                  input_id = "favorite_food",
                  dependence = NA,
                  dependence_value = NA,
                  required = F)

  ui <- fluidPage(
    surveyOutput(df = df,
                survey_title = "Hello, World!",
                theme = "#63B8FF")
  )

  server <- function(input, output, session) {
    renderSurvey()

    observeEvent(input$submit, {
      showModal(modalDialog(
        title = "Congrats, you completed your first shinysurvey!",
        "You can customize what actions happen when a user finishes a survey using input$submit."
      ))
    })
  }

  shinyApp(ui, server)
}

```

teaching_r_questions *A sample CSV file for demographic questions*

Description

A dataset containing the prices and other attributes of almost 54,000 diamonds.

Usage

```
teaching_r_questions
```

Format

A data frame with 54 rows and 6 columns:

question: The question to be asked.

option: A possible response to the question. In multiple choice questions, for example, this would be the possible answers. For questions without discrete answers, such as a numeric input, this would be the default option shown on the input. For text inputs, it is the placeholder value.

input_type: What type of response is expected? Numeric, multiple choice, text, etc...

input_id: The input id for Shiny inputs.

dependence: Does this question (row) depend on another? That is, should it only appear if a different question has a specific value? This column contains the input_id of whatever question this one depends upon.

dependence_value: This column contains the specific value that the dependence question must take for this question (row) to be shown.

required: logical TRUE/FALSE signifying if a question is required.

Source

D'Agostino McGowan Data Science Lab at Wake Forest University.

Index

- * **datasets**
 - teaching_r_questions, [17](#)
- actionButton, [16](#)
- demo_survey, [2](#)
- demo_survey_multipage, [3](#)
- extendInputType, [3](#), [13–15](#)
- getSurveyData, [5](#), [10](#)
- hideSurvey, [7](#)
- listInputExtensions, [8](#)
- numberInput, [9](#)
- radioMatrixInput, [10](#)
- renderSurvey, [12](#)
- shiny::numericInput(), [9](#)
- shiny::updateNumericInput(), [9](#)
- surveyID, [4](#), [13](#), [14](#), [15](#)
- surveyLabel, [4](#), [13](#), [14](#)
- surveyOptions, [13–15](#), [15](#)
- surveyOutput, [12](#), [14](#), [15](#)
- teaching_r_questions, [17](#)
- validateCssUnit(), [9](#)