

Package ‘shinyFeedback’

October 14, 2022

Type Package

Title Display User Feedback in Shiny Apps

Version 0.4.0

Date 2021-09-23

Description Easily display user feedback in Shiny apps.

License MIT + file LICENSE

Depends R (>= 3.1.2)

RoxygenNote 7.1.2

Encoding UTF-8

Suggests knitr, rmarkdown

VignetteBuilder knitr

Imports fontawesome, htmltools, jsonlite, shiny

URL <https://github.com/merlinoa/shinyFeedback>

BugReports <https://github.com/merlinoa/shinyFeedback/issues>

NeedsCompilation no

Author Andy Merlino [aut, cre],
Patrick Howard [aut]

Maintainer Andy Merlino <andy.merlino@tychobra.com>

Repository CRAN

Date/Publication 2021-09-23 18:30:08 UTC

R topics documented:

feedback	2
hideFeedback	3
hideToast	4
loadingButton	4
resetLoadingButton	5
showFeedback	5

showToast	8
useShinyFeedback	9
valueBoxModule	9
valueBoxModuleUI	10

Index	12
--------------	-----------

feedback	<i>feedback</i>
----------	-----------------

Description

Show / hide feedback messages.

Usage

```
feedback(
  inputId,
  show,
  text = NULL,
  color = NULL,
  icon = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

```
feedbackWarning(
  inputId,
  show,
  text = "Ye be warned",
  color = "#F89406",
  icon = shiny::icon("warning-sign", lib = "glyphicon"),
  session = shiny::getDefaultReactiveDomain()
)
```

```
feedbackDanger(
  inputId,
  show,
  text = "Danger, turn back!",
  color = "#d9534f",
  icon = shiny::icon("exclamation-sign", lib = "glyphicon"),
  session = shiny::getDefaultReactiveDomain()
)
```

```
feedbackSuccess(
  inputId,
  show,
  text = NULL,
  color = "#5cb85c",
```

```
    icon = shiny::icon("ok", lib = "glyphicon"),
    session = shiny::getDefaultReactiveDomain()
  )
```

Arguments

inputId	the Shiny input's inputId argument
show	Whether or not the feedback should be shown. The 'show' argument uses 'shiny::isTruthy()' to evaluate its value to 'TRUE' or 'FALSE'.
text	text string to display below input
color	the color of the feedback
icon	an html icon tag
session	the shiny session

See Also

showFeedback hideFeedback

hideFeedback	<i>hideFeedback</i>
--------------	---------------------

Description

hide feedback next to Shiny input

Usage

```
hideFeedback(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	the Shiny input's inputId argument
session	the shiny session

hideToast	<i>Hide existing toast messages</i>
-----------	-------------------------------------

Description

Hide existing toast messages

Usage

```
hideToast(animate = TRUE, session = shiny::getDefaultReactiveDomain())
```

Arguments

animate	a logical indicating whether to remove the toast message(s) instantly or use its hideMethod with animations to remove (Default).
session	the Shiny session. Defaults to shiny::getDefaultReactiveDomain().

Value

‘invisible()‘

loadingButton	<i>loadingButton</i>
---------------	----------------------

Description

Button that becomes disabled until reset w/ ‘resetLoadingButton‘

Usage

```
loadingButton(
  inputId,
  label,
  class = "btn btn-primary",
  style = "width: 150px;",
  loadingLabel = "Loading...",
  loadingSpinner = "spinner",
  loadingClass = NULL,
  loadingStyle = NULL
)
```

Arguments

inputId	the input id
label	the button text (label)
class	the class(es) to apply to the button
style	style for button (pre-loading); character string w/ CSS styling format: "color: black; background-color: red;"
loadingLabel	text to show after button is clicked (e.g. during loading)
loadingSpinner	the loading spinner icon. Valid values are NULL, "spinner", "circle-notch", "sync", and "cog"
loadingClass	the loading button css class(es).
loadingStyle	style for button (while loading); character string w/ CSS styling format: "color: black; background-color: red;"

resetLoadingButton	<i>resetLoadingButton</i>
--------------------	---------------------------

Description

Reset the 'loadingButton' to its original style

Usage

```
resetLoadingButton(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	the input id
session	the shiny session

showFeedback	<i>showFeedback</i>
--------------	---------------------

Description

Show feedback next to Shiny inputs.

Usage

```
showFeedback(  
  inputId,  
  text = NULL,  
  color = NULL,  
  icon = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)  
  
showFeedbackWarning(  
  inputId,  
  text = "Ye be warned",  
  color = "#F89406",  
  icon = shiny::icon("warning-sign", lib = "glyphicon"),  
  session = shiny::getDefaultReactiveDomain()  
)  
  
showFeedbackDanger(  
  inputId,  
  text = "Danger, turn back!",  
  color = "#d9534f",  
  icon = shiny::icon("exclamation-sign", lib = "glyphicon"),  
  session = shiny::getDefaultReactiveDomain()  
)  
  
showFeedbackSuccess(  
  inputId,  
  text = NULL,  
  color = "#5cb85c",  
  icon = shiny::icon("ok", lib = "glyphicon")  
)
```

Arguments

<code>inputId</code>	the Shiny input's <code>inputId</code> argument
<code>text</code>	text string to display below input
<code>color</code>	the color of the feedback
<code>icon</code>	an html icon tag
<code>session</code>	the shiny session

Examples

```
## Only run examples in interactive R sessions  
if (interactive()) {  
  ui <- fluidPage(  
    useShinyFeedback(),  
  )  
}
```

```
    numericInput(
      "exampleInput",
      "Show Feedback When < 0",
      value = -5
    )
  )
}

server <- function(input, output) {
  observeEvent(input$exampleInput, {

    if (input$exampleInput < 0) {

      showFeedback(
        "exampleInput",
        text = "I am negative",
        color = "#d9534f",
        icon = shiny::icon("exclamation-sign", lib="glyphicon")
      )
    } else {
      hideFeedback("exampleInput")
    }

  })
}

shinyApp(ui, server)
}

## Only run examples in interactive R sessions
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    useShinyFeedback(),

    numericInput(
      "exampleInput",
      "Show Feedback When < 0",
      value = -5
    )
  )

  server <- function(input, output, session) {
    observeEvent(input$exampleInput, {

      if (input$exampleInput < 0) {
        showFeedbackWarning("exampleInput")
      } else {
        hideFeedback("exampleInput")
      }

    })
  }
}
```

```

    }
  shinyApp(ui, server)
}

```

showToast	<i>show toast message</i>
-----------	---------------------------

Description

A wrapper around the ‘toastr’ JavaScript library that uses our preferred default argument values.

Usage

```

showToast(
  type,
  message,
  title = NULL,
  keepVisible = FALSE,
  .options = list(),
  session = shiny::getDefaultReactiveDomain()
)

```

Arguments

type	length 1 character vector. Valid values are "success", "error", "warning", and "info"
message	the toast message
title	the toast title. Defaults to NULL
keepVisible	a logical. If TRUE, the toast notification will remain visible until removed with hideToast . If FALSE, the default, the toast will automatically hide once the "showDuration" option has elapsed.
.options	other options to pass to the toastr JavaScript library. See https://codeseven.github.io/toastr/demo.html for a full demo of options. Valid options are "positionClass", "progressBar", "timeOut", "closeButton", "newestOnTop", "preventDuplicates", "showDuration", "hideDuration", "extendedTimeOut", "showEasing", "hideEasing", "showMethod", & "hideMethod"
session	the Shiny session. Defaults to shiny::getDefaultReactiveDomain().

Value

‘invisible()‘

useShinyFeedback	<i>useShinyFeedback</i>
------------------	-------------------------

Description

function to load js for using shinyFeedback

Usage

```
useShinyFeedback(feedback = TRUE, toastr = TRUE)
```

Arguments

feedback	boolean: source in JS/CSS to use shinyFeedback functions (Default: TRUE)
toastr	boolean: source in JS/CSS to use showToast functions (Default: TRUE)

Example

```
ui <- shinyUI(fluidPage(  
  useShinyFeedback(  
    feedback = TRUE,  
    toastr = TRUE  
  ),  
  pageWithSidebar(  
    headerPanel("Header"),  
    sidebarPanel(  
      ...  
    ),  
    mainPanel(  
      ...  
    )  
  )  
)  
)
```

valueBoxModule	<i>valueBoxModule</i>
----------------	-----------------------

Description

Server function for the 'valueBoxModule'. 'valueBoxModule' is similar to 'shinydashboard::valueBox()' but it moves the UI from the server to the ui (i.e. the entire box is not rendered when the value in the value box updates; only the actual value is rerendered). By moving the box content to the UI the value box does not flash onto the screen when rendered.

Usage

```
valueBoxModule(input, output, session, value, subtitle = function() NULL)
```

Arguments

input	the Shiny server input
output	the Shiny server output
session	the Shiny server session
value	Either a reactive or an R object that can be coerced into a string. The value to be displayed in the value box.
subtitle	reactive to dynamically set the subtitle. Set the "subtitle" argument of <code>valueBoxModuleUI()</code> to <code>"__server__"</code> to display this subtitle.

Details

‘valueBoxModule’ also allows for more custom styling of the box colors than ‘shinydashboard::valueBox()’.

valueBoxModuleUI	<i>valueBoxModuleUI</i>
------------------	-------------------------

Description

valueBoxModuleUI

Usage

```
valueBoxModuleUI(
  id,
  subtitle,
  icon = NULL,
  backgroundColor = "#7cb5ec",
  textColor = "#FFF",
  width = 4,
  href = NULL,
  iconColor = "#00000026"
)
```

Arguments

id	the Shiny module id
subtitle	The subtitle to be displayed in the value box. Set to <code>"__server__"</code> to dynamically render the subtitle from the server.
icon	An icon made by the ‘shiny::icon()’
backgroundColor	A hex color code string

textColor	A hex color code string
width	A number between 1 and 12
href	A url
iconColor	A valid color string

Index

feedback, [2](#)
feedbackDanger (feedback), [2](#)
feedbackSuccess (feedback), [2](#)
feedbackWarning (feedback), [2](#)

hideFeedback, [3](#)
hideToast, [4](#), [8](#)

loadingButton, [4](#)

resetLoadingButton, [5](#)

showFeedback, [5](#)
showFeedbackDanger (showFeedback), [5](#)
showFeedbackSuccess (showFeedback), [5](#)
showFeedbackWarning (showFeedback), [5](#)
showToast, [8](#)

useShinyFeedback, [9](#)

valueBoxModule, [9](#)
valueBoxModuleUI, [10](#)