Package 'shinyalert'

July 23, 2025

Title Easily Create Pretty Popup Messages (Modals) in 'Shiny'

Version 3.1.0

Description Easily create pretty popup messages (modals) in 'Shiny'. A modal can contain text, images, OK/Cancel buttons, an input to get a response from the user, and many more customizable options.

URL https://github.com/daattali/shinyalert, https://daattali.com/shiny/shinyalert-demo/

BugReports https://github.com/daattali/shinyalert/issues

Depends R (>= 3.0.2)

Imports htmltools (>= 0.3.5), shiny (>= 1.0.4), uuid

Suggests colourpicker, shinydisconnect

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Dean Attali [aut, cre] (ORCID: <https://orcid.org/0000-0002-5645-3493>, R interface),
Tristan Edwards [aut] (sweetalert library),
Zhengjia Wang [ctb]

Maintainer Dean Attali <daattali@gmail.com>

Repository CRAN

Date/Publication 2024-04-27 23:10:02 UTC

Contents

closeAlert *Close a shinyalert popup message*

Description

Close a shinyalert popup message

Usage

```
closeAlert(num = 0, id = NULL)
```

Arguments

num	Number of popup messages to close. If set to 0 (default) then all messages are closed. This is only useful if you have multiple popups queued up.
id	To close a specific popup, use the ID returned by shinyalert. Note that if id is specified, then num is ignored.

runExample *Run shinyalert example*

Description

Launch an example Shiny app that shows how easy it is to create modals with shinyalert.

The demo app is also available online to experiment with.

Usage

```
runExample()
```

shinyalert *Display a popup message (modal) in Shiny*

Description

Modals can contain text, images, OK/Cancel buttons, Shiny inputs, and Shiny outputs (such as plots and tables). A modal can also have a timer to close automatically, and you can specify custom code to run when a modal closes. See the demo Shiny app online for examples or read the full README.

Usage

```
shinyalert(
  title = "",
  text = "",
  type = "",
  closeOnEsc = TRUE,
  closeOnClickOutside = FALSE,
  html = FALSE,
  showCancelButton = FALSE,
  showConfirmButton = TRUE,
  inputType = "text",
  inputValue = "",
  inputPlaceholder = "",
  confirmButtonText = "OK",
  confirmButtonCol = "#AEDEF4",
  cancelButtonText = "Cancel",
  timer = 0,
  animation = TRUE,
  imageUrl = NULL,
  imageWidth = 100,
  imageHeight = 100,
  className = "",
  callbackR = NULL,
  callbackJS = NULL,
  inputId = "shinyalert",
  size = "s",
  immediate = FALSE,
  session = getSession()
)
```

Arguments

`title`	The title of the modal.
`text`	The modal's text. Can either be simple text, or Shiny tags (including Shiny inputs and outputs). If using Shiny tags, then you must also set `html=TRUE`.
`type`	The type of the modal. There are 4 built-in types which will show a corresponding icon: `"warning"`, `"error"`, `"success"` and `"info"`. You can also set `type="input"` to get a prompt in the modal where the user can enter a response. By default, the modal has no type.
`closeOnEsc`	If `TRUE`, the user can dismiss the modal by pressing the Escape key.
`closeOnClickOutside`	If `TRUE`, the user can dismiss the modal by clicking outside it.
`html`	If `TRUE`, the content of the title and text will not be escaped. By default, the content in the title and text are escaped, so any HTML tags will not render as HTML.

showCancelButton
: If TRUE, a "Cancel" button will be shown, which the user can click on to dismiss the modal.

showConfirmButton
: If TRUE, a "OK" button will be shown. If FALSE, make sure to either use timer, closeOnEsc, or closeOnClickOutside to allow the user a way to close the modal.

inputType
: When using type="input", change the type of the input field. The input type can be "number", "text", "password", or any other valid HTML input type.

inputValue
: When using type="input", specify a default value that you want the input to show initially.

inputPlaceholder
: When using type="input", specify a placeholder text for the input.

confirmButtonText
: The text in the "OK" button.

confirmButtonCol
: The background colour of the "OK" button (must be a HEX value).

cancelButtonText
: The text in the "Cancel" button.

timer
: The amount of time (in milliseconds) before the modal should close automatically. Use 0 to not close the modal automatically (default).

animation
: If FALSE, the modal's animation will be disabled. Possible values: FALSE, TRUE, "slide-from-top", "slide-from-bottom", "pop" (the default animation when animation=TRUE).

imageUrl
: Add a custom icon to the modal.

imageWidth
: Width of the custom image icon, in pixels.

imageHeight
: Height of the custom image icon, in pixels.

className
: A custom CSS class name for the modal's container.

callbackR
: An R function to call when the modal exits. See the 'Modal return value' and 'Callbacks' sections below.

callbackJS
: A JavaScript function to call when the modal exits. See the 'Modal return value' and 'Callbacks' sections below.

inputId
: The input ID that will be used to retrieve the value of this modal (defualt: "shinyalert"). You can access the value of the modal with input$<inputId>.

size
: The size (width) of the modal. One of "xs" for extra small, "s" for small (default), "m" for medium, or "l" for large.

immediate
: If TRUE, close any previously opened alerts and display the current one immediately.

session
: Shiny session object (only for advanced users).

Value

An ID that can be used by closeAlert to close this specific alert.

Simple input modals

Usually the purpose of a modal is simply informative, to show some information to the user. However, the modal can also be used to retrieve an input from the user by setting the `type = "input"` parameter.

When using a `type="input"` modal, only a single input can be used. By default, the input will be a text input, but you can use other input types by specifying the `inputType` parameter (for example `inputType = "number"` will expose a numeric input).

Shiny inputs/outputs in modals

While simple input modals are useful for retrieving input from the user, they aren't very flexible - they only allow one input. You can include any Shiny UI code in a modal, including Shiny inputs and outputs (such as plots), by providing Shiny tags in the `text` parameter and setting `html=TRUE`. For example, the following code would produce a modal with two inputs:

```
shinyalert(html = TRUE, text = tagList(
  textInput("name", "What's your name?", "Dean"),
  numericInput("age", "How old are you?", 30),
))
```

Modal return value

Modals created with {shinyalert} have a return value when they exit.

When using a simple input modal (`type="input"`), the value of the modal is the value the user entered. Otherwise, the value of the modal is `TRUE` if the user clicked the "OK" button, and `FALSE` if the user dismissed the modal (either by clicking the "Cancel" button, using the Escape key, clicking outside the modal, or letting the `timer` run out).

The return value of the modal can be accessed via `input$shinyalert` (or using a different input ID if you specify the `inputId` parameter), as if it were a regular Shiny input. The return value can also be accessed using the *modal callbacks* (see below).

Callbacks

The return value of the modal is passed as an argument to the `callbackR` and `callbackJS` functions (if a `callbackR` or `callbackJS` arguments are provided). These functions get called (in R and in JavaScript, respectively) when the modal exits.

For example, using the following {shinyalert} code will result in a modal with an input field. After the user clicks "OK", a hello message will be printed to both the R console and in a native JavaScript alert box. You don't need to provide both callback functions, but in this example both are used for demonstration.

```
shinyalert(
  "Enter your name", type = "input",
  callbackR = function(x) { message("Hello ", x) },
  callbackJS = "function(x) { alert('Hello ' + x); }"
)
```

Notice that the `callbackR` function accepts R code, while the `callbackJS` function uses JavaScript code.

Since closing the modal with the Escape key results in a return value of `FALSE`, the callback functions can be modified to not print anything in that case.

```
shinyalert(
  "Enter your name", type = "input",
  callbackR = function(x) { if(x != FALSE) message("Hello ", x) },
  callbackJS = "function(x) { if (x !== false) { alert('Hello ' + x); } }"
)
```

Chaining modals

It's possible to chain modals (call multiple modals one after another) by making a `shinyalert()` call inside a shinyalert callback or using the return value of a previous modal. For example:

```
shinyalert(
  title = "What is your name?", type = "input",
  callbackR = function(value) { shinyalert(paste("Welcome", value)) }
)
```

See Also

useShinyalert

Examples

```
# Example 1: Simple modal
if (interactive()) {
  library(shiny)
  library(shinyalert)

  shinyApp(
    ui = fluidPage(
      actionButton("btn", "Click me")
    ),
    server = function(input, output) {
      observeEvent(input$btn, {
        # Show a simple modal
        shinyalert(title = "You did it!", type = "success")
      })
    }
  )
}

# Example 2: Simple input modal calling another modal in its callback
if (interactive()) {
  library(shiny)
  library(shinyalert)

  shinyApp(
```

```
    ui = fluidPage(
      actionButton("btn", "Greet")
    ),
    server = function(input, output) {
      observeEvent(input$btn, {
        shinyalert(
          title = "What is your name?", type = "input",
          callbackR = function(value) { shinyalert(paste("Welcome", value)) }
        )
      })
    }
  )
}

# Example 3: Modal with Shiny tags (input and output)
if (interactive()) {
  library(shiny)
  library(shinyalert)

  shinyApp(
    ui = fluidPage(
      actionButton("btn", "Go")
    ),
    server = function(input, output) {
      observeEvent(input$btn, {
        shinyalert(
          html = TRUE,
          text = tagList(
            numericInput("num", "Number", 10),
            "The square of the number is",
            textOutput("square", inline = TRUE)
          )
        )
      })

      output$square <- renderText({ input$num*input$num })
    }
  )
}
```

`useShinyalert` *Set up a Shiny app to use shinyalert*

Description

This function is no longer required.

The first time a {shinyalert} message is shown, the required scripts are *automatically* inserted to the Shiny app. Usually this is not an issue, but in some unique cases this can sometimes cause the modal to appear glitchy (such as inside RStudio's Viewer, on some old browsers, or if the modal

contains certain Shiny inputs).

If you notice issues with the UI of the modal, you may want to try to pre-load the scripts when the Shiny app initializes by calling `useShinyalert(force=TRUE)` anywhere in the UI.

Usage

```
useShinyalert(rmd, force = FALSE)
```

Arguments

`rmd`	Deprecated, do not use this parameter.
`force`	Set to `TRUE` to force pre-loading the {shinyalert} scripts. If `FALSE` (default), you will get a warning saying this function is not required.

Value

Scripts that `shinyalert` requires that are automatically inserted to the app's `<head>` tag.

See Also

shinyalert

Examples

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

  shinyApp(
    ui = fluidPage(
      useShinyalert(force = TRUE),  # Set up shinyalert
      actionButton("btn", "Click me")
    ),
    server = function(input, output) {
      observeEvent(input$btn, {
        # Show a simple modal
        shinyalert(title = "You did it!", type = "success")
      })
    }
  )
}
```

Index