

# Package ‘shinyalert’

April 29, 2020

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

**Version** 1.1

**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** digest, shiny (>= 1.0.4), stats

**Suggests** colourpicker, knitr, rmarkdown

**License** MIT + file LICENSE

**SystemRequirements** pandoc with https support

**LazyData** true

**VignetteBuilder** knitr

**RoxygenNote** 7.1.0

**NeedsCompilation** no

**Author** Dean Attali [aut, cre] (R interface),  
Tristan Edwards [aut] (sweetalert library)

**Maintainer** Dean Attali <daattali@gmail.com>

**Repository** CRAN

**Date/Publication** 2020-04-29 16:50:02 UTC

## R topics documented:

runExample	2
shinyalert	2
useShinyalert	6

<b>Index</b>	<b>8</b>
--------------	----------

---

runExample	<i>Run shinyalert example</i>
------------	-------------------------------

---

### 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	<i>Display a popup message (modal) in Shiny</i>
------------	---

---

### Description

A modal can contain text, images, OK/Cancel buttons, an input to get a response from the user, and many more customizable options. The value of the modal can be retrieved in Shiny using `input$shinyalert` or using the two callback parameters. See the [demo Shiny app](#) online for examples or read the [full README](#).

shinyalert must be initialized with a call to `useShinyalert` in the app's UI.

### 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"
  )

```

## Arguments

<code>title</code>	The title of the modal.
<code>text</code>	The modal's text.
<code>type</code>	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 <code>type="input"</code> to get a prompt in the modal where the user can enter a response. By default, the modal has no type.
<code>closeOnEsc</code>	If TRUE, the user can dismiss the modal by pressing the Escape key.
<code>closeOnClickOutside</code>	If TRUE, the user can dismiss the modal by clicking outside it.
<code>html</code>	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.
<code>showCancelButton</code>	If TRUE, a "Cancel" button will be shown, which the user can click on to dismiss the modal.
<code>showConfirmButton</code>	If TRUE, a "OK" button will be shown. If FALSE, make sure to either use <code>timer</code> , <code>closeOnEsc</code> , or <code>closeOnClickOutside</code> to allow the user a way to close the modal.
<code>inputType</code>	When using <code>type="input"</code> , change the type of the input field. The input type can be "number", "text", "password", or any other valid HTML input type.
<code>inputValue</code>	When using <code>type="input"</code> , specify a default value that you want the input to show initially.
<code>inputPlaceholder</code>	When using <code>type="input"</code> , specify a placeholder text for the input.
<code>confirmButtonText</code>	The text in the "OK" button.
<code>confirmButtonCol</code>	The background colour of the "OK" button (must be a HEX value).
<code>cancelButtonText</code>	The text in the "Cancel" button.
<code>timer</code>	The amount of time (in milliseconds) before the modal should close automatically. Use 0 to not close the modal automatically (default). If the modal closes automatically, no value is returned from the modal. See the 'Modal return value' section below.

<code>animation</code>	If FALSE, the modal's animation will be disabled. Possible values: FALSE, TRUE, "slide-from-top", "slide-from-bottom", "pop" (the default animation when <code>animation=TRUE</code> ).
<code>imageUrl</code>	Add a custom icon to the modal.
<code>imageWidth</code>	Width of the custom image icon, in pixels.
<code>imageHeight</code>	Height of the custom image icon, in pixels.
<code>className</code>	A custom CSS class name for the modal's container.
<code>callbackR</code>	An R function to call when the modal exits. See the 'Modal return value' and 'Callbacks' sections below.
<code>callbackJS</code>	A JavaScript function to call when the modal exits. See the 'Modal return value' and 'Callbacks' sections below.
<code>inputId</code>	The input ID that will be used to retrieve the value of this modal (default: "shinyalert"). You can access the value of the modal with <code>input\$&lt;inputId&gt;</code> .

### 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.

Only a single input can be used inside a modal. By default, the input will be a text input, but you can use other HTML input types by specifying the `inputType` parameter. For example, `inputType = "number"` will provide the user with a numeric input in the modal.

See the 'Modal return value' and 'Callbacks' sections below for information on how to access the value entered by the user.

### Modal return value

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

When there is an input field in the modal (`type="input"`), the value of the modal is the value the user entered. When there is no input field in the modal, the value of the modal is TRUE if the user clicked the "OK" button, and FALSE if the user clicked the "Cancel" button.

When the user exits the modal using the Escape key or by clicking outside of the modal, the return value is FALSE (as if the "Cancel" button was clicked). If the `timer` parameter is used and the modal closes automatically as a result of the timer, no value is returned from the modal.

The return value of the modal can be accessed via `input$shinyalert` (or using a different input ID if you specify the `inputId` parameter) in the Shiny server's code, 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 are functions that get called, either in R or in JavaScript, 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 hello 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(
      useShinyalert(), # 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")
      })
    }
  )
}

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

  shinyApp(
    ui = fluidPage(
      useShinyalert(), # Set up shinyalert
      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)) }
        )
      })
    }
  )
}

```

---

 useShinyalert

*Set up a Shiny app to use shinyalert*


---

### Description

This function must be called from a Shiny app's UI in order for the `shinyalert` function to work.

You can call `useShinyalert()` from anywhere inside the UI.

### Usage

```
useShinyalert(rmd = FALSE)
```

### Arguments

`rmd` Set this to TRUE if using `shinyalert` inside an interactive R markdown document. The YAML of the Rmd file must have `runtime: shiny`.

### 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(), # 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

`runExample`, 2

`shinyalert`, 2, 6, 7

`useShinyalert`, 2, 5, 6