Package: standby (via r-universe)

November 20, 2026

Type Package

Title Alerts, Notifications and Loading Screen in 'Shiny'

Version 0.2.2

Description Easily create alerts, notifications, modals, info tips and loading screens in 'Shiny'. Includes several options to customize alerts and notifications by including text, icons, images and buttons. When wrapped around a 'Shiny' output, loading screen is automatically displayed while the output is being recalculated.

Depends R(>= 3.3)

Imports grDevices, htmltools, shiny

Suggests rmarkdown, kableExtra, knitr, testthat (>= 3.1.5), jsonlite, chromote, covr

License GPL (>= 3)

URL https://standby.rsquaredacademy.com/, https://github.com/rsquaredacademy/standby

BugReports https://github.com/rsquaredacademy/standby/issues

Encoding UTF-8

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Aravind Hebbali [aut, cre], Zong Bin [ctb, cph] (Author of Three Dots), Tobias Ahlin [ctb, cph] (Author of SpinKit), https://github.com/RIDICS [ctb, cph] (CSS loader code), Raphael Fabini [ctb, cph] (Author of included CSS loader code), Luke Hass [ctb, cph] (Author of included CSS loader code), Mohammad Younes [ctb, cph] (Author of Alertify), Nick Payne [ctb, cph] (Author of BootBox), Indrashish Ghosh [ctb, cph] (Author of MicroTip), https://github.com/codrops [ctb, cph] (Author of

Notification Styles), Hunter Perrin [ctb, cph] (Author of PNotify), Robin Parisi [ctb, cph] (Author of Tingle), Marcelo Dolza [ctb, cph] (Author of iziToast)

Maintainer Aravind Hebbali <aravindhebbali@rsquaredacademy.com>

Repository https://cran.r-universe.dev

Date/Publication 2026-11-20 14:30:09 UTC

RemoteUrl https://github.com/cran/standby

RemoteRef HEAD

RemoteSha a6ba1a10bc4a177ead54abb3d45f07aa7b0f2024

Contents

preview . . . 2
useAlertify . . . 3
useBootBox . . . 6
useLoaders . . . 8
useMicroTip . . . 10
useNotify . . . 12
useNS . . . 14
useSpinkit . . . 16
useSpinners . . . 18
useThreeDots . . . 19
useTingle . . . 20
useToast . . . 22
useVizLoad . . . 26

Index **28**

`preview` *Preview Alerts*

Description

Preview different types of alerts/notifications.

Preview different types of spinners/loaders.

Usage

```
previewAlerts(type = "toast")

previewSpinners(spinner = "threedots")
```

Arguments

type	Type of alert/notification. Valid values are: • alertify • bootbox • notice • notify • tingle • toast
spinner	Type of spinner. The following spinners are available: • threedots • spinkit • vizload • spinners • loaders

Value

In an interactive session, a Shiny app object; otherwise NULL invisibly.

In an interactive session, browsable HTML; otherwise NULL invisibly.

---

useAlertify *Alertify*

Description

Pretty browser alerts and notifications.

Usage

```
useAlertify()

alertify_alert(
  title = "Alert Title",
  message = "Alert Message",
  type = "success",
  btn_label = "OK",
  transition = "pulse",
  transition_off = FALSE,
  closable = TRUE,
  auto_reset = FALSE,
  frameless = FALSE,
  maximizable = FALSE,
  modal = FALSE,
  movable = FALSE,
```

```
  move_bounded = TRUE,
  overflow = FALSE,
  padding = TRUE,
  pinnable = FALSE,
  resizeable = FALSE,
  start_maximized = FALSE,
  session = getDefaultReactiveDomain()
)

alertify_notify(
  message = "Alert Message",
  type = "success",
  delay = 5,
  position = "bottom-right",
  session = getDefaultReactiveDomain()
)
```

Arguments

title	Dialog title. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
message	Dialog contents. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
type	Alert/notification type. Defaults to `"success"`. Valid values are: • `"success"` • `"error"` • `"warning"` • `"message"` For `alertify_alert()` the type sets a class on the dialog and a type-specific accent border and header colour. For `alertify_notify()` it sets the notification style.
btn_label	The `OK` button label. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
transition	Transition effect to be used when showing/hiding the dialog. Defaults to `"pulse"`. Valid values are: • `"pulse"` • `"slide"` • `"zoom"` • `"fade"` • `"flipx"` • `"flipy"`
transition_off	Logical; if `TRUE`, transition effect is disabled. Defaults to `FALSE`.
closable	Logical; if `TRUE` (the default), a `Close` button is displayed in the header of the dialog.
auto_reset	Logical; if `TRUE`, the dialog will reset size/position on window resize. Defaults to `FALSE`.
frameless	Logical; if `TRUE`, hides both header and footer of the dialog. Defaults to `FALSE`.

`maximizable` Logical; if `TRUE`, the `Maximize` button is displayed in the header of the dialog. Defaults to `FALSE`.

`modal` Logical; if `TRUE`, a screen dimmer will be used and access to the page contents will be prevented. Defaults to `FALSE`.

`movable` Logical; if `TRUE`, the dialog is movable. Defaults to `FALSE`.

`move_bounded` Logical; if `TRUE` (the default), the dialog is not allowed to go off-screen.

`overflow` Logical; if `TRUE`, the content overflow is managed by the dialog. Defaults to `FALSE`.

`padding` Logical; if `TRUE` (the default), the content padding is managed by the dialog.

`pinnable` Logical; if `TRUE`, the `Pin` button is displayed in the header of the dialog. Defaults to `FALSE`.

`resizeable` Logical; if `TRUE`, the dialog is resizable. Defaults to `FALSE`.

`start_maximized` Logical; if `TRUE`, the dialog will start in a maximized state. Defaults to `FALSE`.

`session` Shiny session object.

`delay` The time (in seconds) to wait before the notification is auto-dismissed. `0`, `Inf`, or `NULL` will keep notification open till clicked. The strings `"infinity"` and `"infinite"` also keep it open. For compatibility, `FALSE`, `NA`, `NaN`, `-Inf`, and negative numeric delays also keep the notification open.

`position` Position of the notification. Defaults to `"bottom-right"`. Valid values are:

- `"bottom-right"`
- `"bottom-left"`
- `"bottom-center"`
- `"top-right"`
- `"top-left"`
- `"top-center"`

Value

`useAlertify()` returns an HTML dependency to include in the UI. alertify_alert() and alertify_notify() send a custom message to the Shiny session for its side effect and have no meaningful return value.

Functions

- `useAlertify`: Dependencies to include in your UI.
- `alertify_alert`: Display alerts.
- `alertify_notify`: Display notifications.

Examples

```
# Example 1: Alert
if (interactive()) {
library(shiny)
library(standby)
```

```
ui <- fluidPage(

  useAlertify(), # include dependencies
  actionButton(inputId = "btn",
               label   = "Alert Demo")

)

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

  observeEvent(input$btn, {
    # display alert
    alertify_alert("Hey there!", "Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}

# Example 2: Notification
if (interactive()) {
library(shiny)
library(standby)

ui <- fluidPage(

  useAlertify(), # include dependencies
  actionButton(inputId = "btn",
               label   = "Notification Demo")

)

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

  observeEvent(input$btn, {
    # display notification
    alertify_notify("Hey there! Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

---

useBootBox *BootBox*

Description

Bootstrap modals made easy.

Usage

```
useBootBox()

bootBox(
  title = "Your title",
  message = "Your message here...",
  size = "small",
  close_on_escape = TRUE,
  show = TRUE,
  backdrop = NULL,
  close_button = TRUE,
  animate = TRUE,
  class = NULL,
  session = getDefaultReactiveDomain()
)
```

Arguments

`title`	Adds a header to the dialog. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
`message`	Text displayed in the dialog. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
`size`	Adds the relevant Bootstrap modal size class to the dialog wrapper. Valid values are: • `"small"` • `"large"` • `"extra-large"` With Shiny's default Bootstrap 3, extra-large uses a responsive width up to 1140 pixels on viewports at least 992 pixels wide. The aliases `"sm"`, `"lg"`, and `"xl"` are also accepted; `NULL` or `""` uses the standard modal size.
`close_on_escape`	Logical; if `TRUE` (the default), allows the user to dismiss the dialog by hitting `ESC` key.
`show`	Logical; if `TRUE` (the default), the dialog is shown immediately.
`backdrop`	Logical; if `TRUE`, the backdrop is displayed and clicking on it dismisses the dialog. Defaults to `NULL`. Valid values are: • `NULL`: The backdrop is displayed, but clicking on it has no effect. • `TRUE`: The backdrop is displayed, and clicking on it dismisses the dialog. • `FALSE`: The backdrop is not displayed. The Bootbox value `"static"` is also accepted and behaves like `NULL`.
`close_button`	Logical; if `TRUE` (the default), a close button is displayed.
`animate`	Logical; if `TRUE` (the default), animates the dialog in and out.
`class`	Custom CSS class (using Animate.css).
`session`	Shiny session object.

Value

useBootBox() returns a Shiny tag list of HTML dependencies to include in the UI. bootBox() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- useBootBox: Dependencies to include in your UI.
- bootBox: Display modals.

Examples

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

ui <- fluidPage(

  useBootBox(), # include dependencies
  actionButton(inputId = "btn",
               label   = "BootBox Demo")

)

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

  observeEvent(input$btn, {
    # display modal
    bootBox("Hey there!", "Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

---

useLoaders *CSS Loaders*

Description

Simple CSS loaders

Usage

```
useLoaders()

loaders(uiOutput, type = "default", style = NULL, text = NULL)
```

Arguments

`uiOutput`	An output element to be wrapped within a loader.
`type`	The type of loader to use. Visit https://css-loader.raphaelfabeni.com/ for details. • default • bar • bar-ping-pong • border • double • clock • curtain • pokeball • ball • smartphone • bouncing • music The `music` type uses the bundled animated `hey-oh` coin graphic.
`style`	Custom styling for the loaders: `"half"`, `"inverse"`, `"rounded"`, or `"blink"`. Use `NULL` for no custom style. `"half"` supports only `"default"`; `"inverse"` supports `"default"` and `"bar"`; `"rounded"` supports `"bar"` and `"bar-ping-pong"`; `"blink"` supports `"default"`, `"bar"`, and `"border"`, blinking their text (`"Loading"` when `text = NULL`). Supported styles can be combined with custom text. Unsupported type/style or type/text combinations raise an error.
`text`	Custom text. Available only for the following types: • default • bar • border • curtain • smartphone Must be `NULL` or one non-missing character string.

Details

The supported domain is the listed types and type-specific style/text combinations. These loaders display a viewport overlay and require CSS animation enabled in a modern browser. Allow at least 320 by 320 pixels for the largest graphic at its default size.

Value

`useLoaders()` returns a Shiny tag list of HTML dependencies to include in the UI; `loaders()` returns a Shiny tag wrapping `uiOutput`.

Functions

- `useLoaders`: Dependencies to include in your UI.
- `loaders`: Display loading animation.

Examples

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

  shinyApp(
    ui = fluidPage(
      useLoaders(),
      actionButton("render", "Render"),
      loaders(uiOutput = plotOutput("plot"),
              type = "default",
              style = "half",
              text = "Loading...")
    ),
    server = function(input, output) {
      output$plot <- renderPlot({
        input$render
        Sys.sleep(3)
        hist(mtcars$mpg)
      })
    }
  )
}
```

`useMicroTip` *MicroTip*

Description

Minimal CSS only tooltip.

Usage

```
useMicroTip()

microTip(
  id = NULL,
  tip = "Hey! tooltip!",
  position = "top",
  size = NULL,
  session = getDefaultReactiveDomain()
)
```

Arguments

id	The id of the element to attach the tooltip.
tip	Content of the tooltip.
position	Where the tooltip should appear relative to the target element. Defaults to "top". Valid values are: • "top" • "bottom" • "left" • "right" • "top-left" • "top-right" • "bottom-left" • "bottom-right"
size	Size of the tooltip. Defaults to NULL, which fits the text, as do "fit" and "". Valid values are: • "fit" • "small" • "medium" • "large"
session	Shiny session object.

Value

useMicroTip() returns a Shiny tag list of HTML dependencies to include in the UI. microTip() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- useMicroTip: Dependencies to include in your UI.
- microTip: Add tooltip.

Examples

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

ui <- fluidPage(

  useMicroTip(), # include dependencies
  br(), br(), br(), br(),
  actionButton(inputId = "btn",
               label   = "MicroTip Demo")

)
```

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

  # display tooltip
  microTip(id = "btn",
           tip = "Hey there! This is a micro tip!",
           position = "bottom-right")

}

shinyApp(ui, server)
}
```

---

useNotify *PNotify*

Description

Beautiful notifications and prompts.

Usage

```
useNotify()

notify(
  title = "Hey",
  text = NULL,
  type = "notice",
  icon = TRUE,
  delay = 8000,
  hide = TRUE,
  sticker = TRUE,
  closer = TRUE,
  shadow = TRUE,
  mouse_reset = TRUE,
  animation = "fade",
  animate_speed = "normal",
  width = "360px",
  min_height = "16px",
  max_text_height = "200px",
  translucent = FALSE,
  non_blocking = FALSE,
  session = getDefaultReactiveDomain()
)
```

Arguments

title	Title of the notice. Defaults to "Hey". It can be a string, an element or FALSE for no title. Tag objects are rendered to HTML; HTML dependencies attached to them are not installed.
text	Text of the notice. Defaults to NULL. It can be a string, an element or FALSE for no text. Tag objects are rendered to HTML; HTML dependencies attached to them are not installed.
type	Type of notice. Defaults to "notice". Other valid values are: • "info" • "success" • "error"
icon	Logical; if TRUE (the default), default icon is displayed. No icon is displayed if set to FALSE. A string specifying custom icon classes is also accepted.
delay	Delay in milliseconds before the notice is removed. If set to "infinity" (or Inf), the notice will not close.
hide	Logical; if TRUE (the default), notice is closed after delay specified in milliseconds.
sticker	Logical; if TRUE (the default), provides a button for the user to manually stick the notice.
closer	Logical; if TRUE (the default), provides a button for the user to manually close the notice.
shadow	Logical; if TRUE (the default), displays a drop shadow.
mouse_reset	Logical; if TRUE (the default), resets the hide timer if the mouse moves over the notice.
animation	The animation to be used while displaying and hiding the notice. "none" and "fade" (the default) are supported out of the box.
animate_speed	Speed at which the notice animates in and out. Valid values are: • "slow": 400ms • "normal": 250ms • "fast": 100ms
width	Width of the notice. Default is "360px". Bare numbers are interpreted as pixels.
min_height	Minimum height of the notice. Default is "16px". It will expand to fit the content. Bare numbers are interpreted as pixels.
max_text_height	Maximum height of the text container. Default is "200px". If the text goes beyond this height, scrollbars will appear. Use NULL to remove this restriction. Bare numbers are interpreted as pixels.
translucent	Logical; if TRUE, creates see through notice. Defaults to FALSE.
non_blocking	Logical; if TRUE, the notice body passes pointer events to elements underneath and fades while the pointer is over it. Close and sticker controls remain interactive. Defaults to FALSE.
session	Shiny session object.

Value

useNotify() returns an HTML dependency to include in the UI. notify() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- useNotify: Dependencies to include in your UI.
- notify: Display notifications.

Examples

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

ui <- fluidPage(

  useNotify(), # include dependencies
  actionButton(inputId = "btn",
               label   = "PNotify Demo")

)

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

  observeEvent(input$btn, {
    # display notification
    notify("Hey there!", "Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

---

useNS *Notification Styles*

Description

Simple website notifications with effects

Usage

```
useNS()

notice(
  message = "Hello",
```

```
  type = "notice",
  layout = "growl",
  effect = NULL,
  session = getDefaultReactiveDomain()
)
```

Arguments

message	Notification message. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
type	Notification type. Defaults to `"notice"`. Other valid values are: • `"success"` • `"warning"` • `"error"` The notice, success, warning and error types have distinct background and foreground colours in every layout.
layout	Notification layout. Defaults to `"growl"`. Other valid values are: • `"attached"` • `"bar"`
effect	Notification effect type. The default `NULL` selects `"jelly"` for growl, `"flip"` for attached, or `"slidetop"` for bar. An explicit effect must be valid for the selected layout: • For `"growl"` layout – `"scale"` – `"jelly"` – `"slide"` – `"genie"` • For`"attached"` layout – `"flip"` – `"bouncyflip"` • For `"bar"` layout – `"slidetop"` – `"exploader"`
session	Shiny session object.

Value

`useNS()` returns an HTML dependency to include in the UI. notice() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- `useNS`: Dependencies to include in your UI.
- `notice`: Display notifications.

Examples

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

ui <- fluidPage(

  useNS(), # include dependencies
  actionButton(inputId = "btn",
               label   = "Notice Demo")

)

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

  observeEvent(input$btn, {
    # display notification
    notice("Hey there! Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

`useSpinkit` *SpinKit*

Description

Simple CSS spinners.

Usage

```
useSpinkit()

spinkit(uiOutput, type = "plane", color = "#333", size = "40px")
```

Arguments

`uiOutput`	An output element to be wrapped within a spinner.
`type`	Type of spinner to use. Valid values are: • plane • chase • bounce • wave • pulse

- flow
- swing
- circle
- circle-fade
- grid
- fold
- wander

`color` Color of the spinner. Defaults to `"#333"`. Choose between a color name, hexadecimal, or `rgb()`/`rgba()`/`hsl()`/`hsla()` values. Names use R's `colors()` values; CSS colour names not in R are also accepted.

`size` Size of the spinner as a positive CSS length. Bare numbers and unitless numeric strings are interpreted as pixels. Defaults to `"40px"`.

Details

The supported domain is the listed types, colour formats, and finite positive CSS sizes. The output must have enough room for the requested size. CSS animation must be enabled in a modern browser. Transparent colours intentionally hide the spinner.

Value

`useSpinkit()` returns a Shiny tag list of HTML dependencies to include in the UI; `spinkit()` returns a Shiny tag list wrapping `uiOutput`.

Functions

- `useSpinkit`: Dependencies to include in your UI.
- `spinkit`: Display loading animation.

Examples

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

  shinyApp(
    ui = fluidPage(
      useSpinkit(),
      actionButton("render", "Render"),
      spinkit(plotOutput("plot"), type = "circle-fade")
    ),
    server = function(input, output) {
      output$plot <- renderPlot({
        input$render
        Sys.sleep(3)
        hist(mtcars$mpg)
      })
    }
  )
}
```

`useSpinners` *Single Element CSS Spinners*

Description

A collection of loading spinners animated with CSS

Usage

```
useSpinners()

spinners(uiOutput, type = 1, color = "#0275d8")
```

Arguments

`uiOutput`	An output element to be wrapped within a spinner.
`type`	Type of spinner to use. Any number or string between `1` and `8` (or `"1"` and `"8"`) is valid.
`color`	Color of the spinner. Choose between a color name, hexadecimal, or `rgb()`/`rgba()`/`hsl()`/`hsla()` values. Names use R's `colors()` values (so `"green"` is `#00FF00`); CSS colour names not in R (such as `"rebeccapurple"`) are also accepted.

Details

The supported domain is types 1 through 8 and the documented colour formats, with CSS animation enabled in a modern browser and sufficient room for the graphic. Transparent colours intentionally hide the spinner.

Value

`useSpinners()` returns a Shiny tag list of HTML dependencies to include in the UI; `spinners()` returns a Shiny tag list wrapping `uiOutput`.

Functions

- `useSpinners`: Dependencies to include in your UI.
- `spinners`: Display loading animation.

Examples

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

  shinyApp(
    ui = fluidPage(
      useSpinners(),
      actionButton("render", "Render"),
```

```
      spinners(plotOutput("plot"))
    ),
    server = function(input, output) {
      output$plot <- renderPlot({
        input$render
        Sys.sleep(3)
        hist(mtcars$mpg)
      })
    }
  )
}
```

---

useThreeDots *Three Dots*

Description

Single element CSS loading animation.

Usage

```
useThreeDots()

threeDots(uiOutput, type = "elastic", color = "#9880ff")
```

Arguments

uiOutput	An output element to be wrapped within a loader.
type	The type of animation to use. Visit https://nzbin.github.io/three-dots/ for details. Supported types are elastic, pulse, flashing, collision, revolution, carousel, typing, windmill, bricks, floating, fire, spin, falling, stretching, gathering, hourglass, overtaking, shuttle, bouncing, and rolling.
color	The color of the loader. Choose between a color name, hexadecimal, or rgb()/rgba()/hsl()/hsla() values. Names use R's colors() values; CSS colour names not in R are also accepted. The emoji types bouncing and rolling retain their native colours.

Details

The supported domain is the listed types and colour formats, rendered with CSS animation enabled in a modern browser and sufficient room in the output for the animation. Transparent colours intentionally hide the dots.

Value

useThreeDots() returns a Shiny tag list of HTML dependencies to include in the UI; threeDots() returns a Shiny tag list wrapping uiOutput.

Functions

- `useThreeDots`: Dependencies to include in your UI.
- `threeDots`: Display loading animation.

Examples

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

  shinyApp(
    ui = fluidPage(
      useThreeDots(),
      actionButton("render", "Render"),
      threeDots(plotOutput("plot"))
    ),
    server = function(input, output) {
      output$plot <- renderPlot({
        input$render
        Sys.sleep(3)
        hist(mtcars$mpg)
      })
    }
  )
}
```

`useTingle` *Tingle*

Description

Minimalist and easy to use modals.

Usage

```
useTingle()

tingle(
  content = "Hello",
  close_button = FALSE,
  button_label = "Close",
  button_type = "default",
  button_position = "right",
  session = getDefaultReactiveDomain()
)
```

Arguments

content	Content of the modal. Interpreted as HTML. Escape untrusted input with htmltools::htmlEscape().
close_button	Logical; if TRUE, displays a button to close the modal. Defaults to FALSE.
button_label	Label of close_button. Interpreted as HTML. Escape untrusted input with htmltools::htmlEscape().
button_type	Type of button. Defaults to "default". Other valid values are: • "primary" • "danger"
button_position	Position of the button inside the modal. Defaults to "right". Valid values are: • "right" • "left"
session	Shiny session object.

Value

useTingle() returns an HTML dependency to include in the UI. tingle() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- useTingle: Dependencies to include in your UI.
- tingle: Display modals.

Examples

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

ui <- fluidPage(

  useTingle(), # include dependencies
  actionButton(inputId = "btn",
               label   = "Tingle Demo")

)

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

  observeEvent(input$btn, {
    # display modal
    tingle("Hey there!, Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

useToast *iziToast*

Description

Lightweight toast notifications

Usage

```
useToast()

toast(
  title = "Hey",
  message = NULL,
  type = NULL,
  theme = NULL,
  position = "center",
  duration = 5000,
  progress_bar_color = NULL,
  background_color = NULL,
  max_width = NULL,
  title_color = NULL,
  title_size = NULL,
  title_line_height = NULL,
  message_color = NULL,
  message_size = NULL,
  message_line_height = NULL,
  image = NULL,
  image_width = "50px",
  zindex = 99999,
  layout = 1,
  balloon = FALSE,
  close = TRUE,
  close_on_escape = FALSE,
  close_on_click = FALSE,
  rtl = FALSE,
  display_mode = 0,
  drag_to_close = TRUE,
  pause_on_hover = TRUE,
  reset_on_hover = FALSE,
  progress_bar_easing = "linear",
  overlay = FALSE,
  overlay_close = FALSE,
  overlay_color = "rgba(0, 0, 0, 0.6)",
  animate_inside = TRUE,
  transition_in = "fadeInUp",
  transition_out = "fadeOut",
```

```
  session = getDefaultReactiveDomain()
)
```

Arguments

title	Title of the toast. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
message	Message of toast. Interpreted as HTML. Escape untrusted input with `htmltools::htmlEscape()`.
type	Type of notification. Defaults to `NULL`. Valid values are: • `"info"` • `"success"` • `"warning"` • `"error"` • `"question"`
theme	Theme of toast: `"light"`, `"dark"`, or a custom theme name used as the suffix of the `iziToast-theme-` class. Must be one non-NA string without whitespace or quotes; `NULL` or `""` uses the default theme.
position	Where toast will be shown. Defaults to `"center"`. Valid values are: • `"bottomRight"` • `"bottomLeft"` • `"topRight"` • `"topLeft"` • `"topCenter"` • `"bottomCenter"` • `"center"`
duration	Time in milliseconds to close the toast. Defaults to `5000`. Use `FALSE`, `Inf`, or `NULL` to disable. The strings `"infinity"` and `"infinite"` also disable closing. For compatibility, `0`, `NA`, `NaN`, and `-Inf` also keep the toast open.
progress_bar_color	Progress bar color. Choose between hexadecimal, RGB or keyword values.
background_color	Background color of the toast. Choose between hexadecimal, RGB or keyword values.
max_width	Maximum width of the toast.
title_color	Title color. Choose between hexadecimal, RGB or keyword values.
title_size	Title font size. A bare number is interpreted as pixels.
title_line_height	Title line height. A bare number is interpreted as pixels.
message_color	Message color. Choose between hexadecimal, RGB or keyword values.
message_size	Message font size. A bare number is interpreted as pixels.
message_line_height	Message line height. A bare number is interpreted as pixels.
image	Cover image.

`image_width` Width of cover image. Defaults to `"50px"`. A bare number is interpreted as pixels.

`zindex` The z-index CSS attribute of the toast. Defaults to `99999`.

`layout` Size of the toast. Choose between `1`, `2`, or `3`.

`balloon` Logical; if `TRUE`, applies a balloon like toast. Defaults to `FALSE`.

`close` Logical; if `TRUE` (the default), shows a `x` close button.

`close_on_escape` Logical; if `TRUE`, allows to close toast using ESC key. Defaults to `FALSE`.

`close_on_click` Logical; if `TRUE`, allows to close toast by clicking on it. Defaults to `FALSE`.

`rtl` Logical; if `TRUE`, applies Right to Left style. Defaults to `FALSE`.

`display_mode` Rules to show multiple toasts. Default is `0`. Valid values are:

- `0`: Waits until the current toast is closed before displaying a new one.
- `1`: Replaces the toast(s) currently displayed with the new toast.
- `2`: Replaces only a displayed toast with the same title, message and text colour. The identity is derived by iziToast.

`drag_to_close` Logical; if `TRUE` (the default), toast can be closed by dragging it.

`pause_on_hover` Logical; if `TRUE` (the default), pauses the toast timeout while the cursor is on it.

`reset_on_hover` Logical; if `TRUE`, resets the toast timeout while the cursor is on it. Defaults to `FALSE`.

`progress_bar_easing` Animation easing of progress bar. Defaults to `"linear"`.

`overlay` Logical; if `TRUE`, displays the overlay layer on the page. Defaults to `FALSE`.

`overlay_close` Logical; if `TRUE`, allows to close the toast by clicking on the overlay. Defaults to `FALSE`.

`overlay_color` Overlay background color. Defaults to `"rgba(0, 0, 0, 0.6)"`. Choose between hexadecimal, RGB or keyword values.

`animate_inside` Logical; if `TRUE` (the default), enables animation of elements in the toast.

`transition_in` Toast open animation. Defaults to `"fadeInUp"`. Valid values are:

- `"bounceInLeft"`
- `"bounceInRight"`
- `"bounceInUp"`
- `"bounceInDown"`
- `"fadeIn"`
- `"fadeInDown"`
- `"fadeInUp"`
- `"fadeInLeft"`
- `"fadeInRight"`
- `"flipInX"`
- `"revealIn"`
- `"slideIn"`

Use `NULL` or `""` for no open animation.

`transition_out` Toast close animation. Defaults to `"fadeOut"`. Valid values are:

- `"fadeOut"`
- `"fadeOutDown"`
- `"fadeOutUp"`
- `"fadeOutLeft"`
- `"fadeOutRight"`
- `"flipOutX"`

Use `NULL` or `""` for no close animation.

`session` Shiny session object.

Value

`useToast()` returns an HTML dependency to include in the UI. toast() sends a custom message to the Shiny session for its side effect and has no meaningful return value.

Functions

- `useToast`: Dependencies to include in your UI.
- `toast`: Display toast notifications.

Examples

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

ui <- fluidPage(

  useToast(), # include dependencies
  actionButton(inputId = "btn",
               label   = "iziToast Demo")

)

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

  observeEvent(input$btn, {
    # display toast notification
    toast("Hey there!", "Thank you for exploring standby!")
  })
}

shinyApp(ui, server)
}
```

useVizLoad *Loading Visualization*

Description

Loading bars and spinners.

Usage

```
useVizLoad()

vizLoad(
  uiOutput,
  type = "bars",
  size = "large",
  color = NULL,
  add_label = FALSE,
  label = "Loading..."
)
```

Arguments

uiOutput	An output element to be wrapped within a spinner.
type	The type of bar/spinner to use. Valid values are: • bars • squares • circles • dots • spinner • dashed • line • bordered_line
size	The size of the bar/spinner. For "line" and "bordered_line", valid values are "large", "medium", "small", and "fluid". For all other types, valid values are "large", "medium", "small", and "tiny". Supply one non-missing character string. Line widths are capped at the output width when the nominal preset is larger than its container.
color	The color of the bar/spinner. Choose between a color name, hexadecimal, or rgb()/rgba()/hsl()/hsla() values. Names use R's colors() values; CSS colour names not in R are also accepted.
add_label	Logical; if TRUE, displays a label below the bar/spinner. Defaults to FALSE.
label	The label to be displayed below the bar/spinner. add_label must be set to TRUE to display the label.

Details

The supported domain is the listed type/size combinations and colour formats, with CSS animation enabled in a modern browser and sufficient room for the chosen graphic. Transparent colours intentionally hide the graphic.

Value

`useVizLoad()` returns a Shiny tag list of HTML dependencies to include in the UI; `vizLoad()` returns a Shiny tag list wrapping `uiOutput`.

Functions

- `useVizLoad`: Dependencies to include in your UI.
- `vizLoad`: Display loading animation.

Examples

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

  shinyApp(
    ui = fluidPage(
      useVizLoad(),
      actionButton("render", "Render"),
      vizLoad(plotOutput("plot"))
    ),
    server = function(input, output) {
      output$plot <- renderPlot({
        input$render
        Sys.sleep(3)
        hist(mtcars$mpg)
      })
    }
  )
}
```

Index

alertify *(useAlertify)*, 3
alertify_alert *(useAlertify)*, 3
alertify_notify *(useAlertify)*, 3

bootBox *(useBootBox)*, 6

loaders *(useLoaders)*, 8

microTip *(useMicroTip)*, 10

notice *(useNS)*, 14
notify *(useNotify)*, 12

preview, 2
previewAlerts *(preview)*, 2
previewSpinners *(preview)*, 2

spinkit *(useSpinkit)*, 16
spinners *(useSpinners)*, 18

threeDots *(useThreeDots)*, 19
tingle *(useTingle)*, 20
toast *(useToast)*, 22

useAlertify, 3
useBootBox, 6
useLoaders, 8
useMicroTip, 10
useNotify, 12
useNS, 14
useSpinkit, 16
useSpinners, 18
useThreeDots, 19
useTingle, 20
useToast, 22
useVizLoad, 26

vizLoad *(useVizLoad)*, 26