Package ‘standby’

October 28, 2024

Type Package

Title Alerts, Notifications and Loading Screen in 'Shiny'

Version 0.2.0

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 (>= 3.0.0), covr

License GPL (>= 3)

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

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

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

Config/testthat/edition 3

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 <hebbali.aravind@gmail.com>

Repository CRAN

Date/Publication 2024-10-28 11:50:32 UTC

Contents

preview 2
useAlertify 3
useBootBox 6
useLoaders 8
useMicroTip 9
useNotify 11
useNS 13
useSpinkit 15
useSpinners 17
useThreeDots 18
useTingle 19
useToast 20
useVizLoad 24

Index **26**

`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

None

None

`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.
`message`	Dialog contents.
`type`	Dialog type. Defaults to `"success"`. Valid values are: • `"success"` • `"error"` • `"warning"` • `"message"`
`btn_label`	The `OK` button label.
`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 default), the dialog will reset size/position on window resize.
`frameless`	Logical; if `TRUE`, hides both header and footer of the dialog. Defaults to `FALSE`.
`maximizable`	Logical; if `TRUE` (the default), the `Maximize` button is displayed in the header of the dialog.
`modal`	Logical; if `TRUE` (the default), a screen dimmer will be used and access to the page contents will be prevented.
`movable`	Logical; if `TRUE` (the default), the dialog is movable.
`move_bounded`	Logical; if `TRUE`, the dialog is not allowed to go off-screen. Defaults to `FALSE`.
`overflow`	Logical; if `TRUE` (the default), the content overflow is managed by the dialog
`padding`	Logical; if `TRUE` (the default), the content padding is managed by the dialog.

`pinnable`	Logical; if `TRUE` (the default), the `Pin` button is displayed in the header of the dialog.
`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` will keep notification open till clicked.
`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

None

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.
`message`	Text displayed in the dialog.
`size`	Adds the relevant Bootstrap modal size class to the dialog wrapper. Valid values are: • `"small"` • `"large"` • `"extra-large"`
`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.
`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

None

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

`style`	Custom styling for the loaders.
`text`	Custom text. Available only for the following types: • default • bar • border • curtain • smartphone

Value

None

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 `"fit"` as the tooltip will takeup only the size it requires to show the text. Valid values are: • `"fit"` • `"small"` • `"medium"` • `"large"`
`session`	Shiny session object.

Value

None

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. It can be a string, an element or `FALSE` (the default) for no title.
`text`	Text of the notice. It can be a string, an element or `FALSE` (the default) for no text.
`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`.
`delay`	Delay in milliseconds before the notice is removed. If set to `"infinity"`, 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"`.
`min_height`	Minimum height of the notice. Default is `"16px"`. It will expand to fit the content.

`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.
`translucent`	Logical; if `TRUE`, creates see through notice. Defaults to `FALSE`.
`non_blocking`	Logical; if `TRUE`, notice fades to show elements underneath. Defaults to `FALSE`.
`session`	Shiny session object.

Value

None

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 = "jelly",
  session = getDefaultReactiveDomain()
)
```

Arguments

`message`	Notification message.
`type`	Notification type. Defaults to `"notice"`. Other valid values are: • `"success"` • `"warning"` • `"error"`
`layout`	Notification layout. Defaults to `"growl"`. Other valid values are: • `"attached"` • `"bar"`

`effect`	Notification effect type. Valid values include:
	• For `"growl"` layout
	  – `"scale"`
	  – `"jelly"`
	  – `"slide"`
	  – `"genie"`
	• For`"attached"` layout
	  – `"flip"`
	  – `"bouncyflip"`
	• For `"bar"` layout
	  – `"slidetop"`
	  – `"exploader"`

`session`	Shiny session object.

Value

None

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 hexadecimal, RGB or keyword values.
`size`	Size of the spinner. Defaults to `"40px"`.

Value

None

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 integer between 1 and 8 is valid.
`color`	Color of the spinner. Choose between hexadecimal or keyword values.

Value

None

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.
`color`	The color of the loader. Choose between hexadecimal, RGB or keyword values.

Value

None

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.
`close_button`	Logical; if `TRUE`, displays a button to close the modal. Defaults to `FALSE`.
`button_label`	Label of `close_button`.
`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

None

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 = NULL,
  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.
`message`	Message of toast.
`type`	Type of notification. Defaults to `NULL`. Valid values are: • `"info"` • `"success"` • `"warning"` • `"error"`
`theme`	Theme of toast. Choose between `"light"` or `"dark"`.
`position`	Where toast will be shown. Defaults to `"bottomRight"`. Valid values are: • `"bottomRight"` • `"bottomLeft"` • `"topRight"` • `"topLeft"` • `"topCenter"` • `"bottomCenter"` • `"center"`

`duration`	Time in milliseconds to close the toast. Defaults to `5000`. Use `FALSE` to disable.
`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.
`title_line_height`	Title line height.
`message_color`	Message color. Choose between hexadecimal, RGB or keyword values.
`message_size`	Message font size.
`message_line_height`	Message line height.
`image`	Cover image.
`image_width`	Width of cover image. Defaults to `"50px"`.
`zindex`	The z-index CSS attribute of the toast. Defaults to `99999`.
`layout`	Size of the toast. Choose between `1` or `2`.
`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 current toast with the new toast toast.
`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"`
`transition_out`	Toast close animation. Defaults to `"fadeOut"`. Valid values are: • `"fadeOut"` • `"fadeOutDown"` • `"fadeOutUp"` • `"fadeOutLeft"` • `"fadeOutRight"` • `"flipOutX"`
`session`	Shiny session object.

Value

None

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. Valid values are: • large • medium • small • tiny

	• fluid
`color`	The color of the bar/spinner. Choose between hexadecimal, RGB or keyword values.
`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.

Value

None

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), 9

notice (useNS), 13
notify (useNotify), 11

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

spinkit (useSpinkit), 15
spinners (useSpinners), 17

threeDots (useThreeDots), 18
tingle (useTingle), 19
toast (useToast), 20

useAlertify, 3
useBootBox, 6
useLoaders, 8
useMicroTip, 9
useNotify, 11
useNS, 13
useSpinkit, 15
useSpinners, 17
useThreeDots, 18
useTingle, 19
useToast, 20
useVizLoad, 24

vizLoad (useVizLoad), 24