

Package ‘waiter’

July 22, 2025

Title Loading Screen for 'Shiny'

Version 0.2.5

Date 2022-01-02

Description

Full screen and partial loading screens for 'Shiny' with spinners, progress bars, and notifications.

License MIT + file LICENSE

URL <https://waiter.john-coene.com/>,
<https://github.com/JohnCoene/waiter>

BugReports <https://github.com/JohnCoene/waiter/issues>

Encoding UTF-8

Imports R6, shiny, htmltools

RoxygenNote 7.1.2

Suggests httr, knitr, packer, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author John Coene [aut, cre],
Jinhwan Kim [ctb],
Victor Granda [ctb] (ORCID: <<https://orcid.org/0000-0002-0469-1991>>)

Maintainer John Coene <jcoenep@gmail.com>

Repository CRAN

Date/Publication 2022-01-03 14:30:02 UTC

Contents

Attendant	2
attendantBar	5
autoWaiter	6
garcon	7
hostess	10

hostessLoader	14
htrr_progress	17
preview_spinner	18
spinners	18
steward	22
transparent	22
triggerWaiter	23
useAttendant	24
waiter	24
waiterClass	27
waiterTheme	29
waitress	30
waitressClass	31
withProgressAttendant	35
withProgressWaitress	36
withWaiter	37
Index	38

Attendant	<i>Attendant</i>
-----------	------------------

Description

Manage the attendant loading bar with bootstrap 4.

Active bindings

max Maximum value of the bar.

Methods

Public methods:

- [Attendant\\$new\(\)](#)
- [Attendant\\$inc\(\)](#)
- [Attendant\\$dec\(\)](#)
- [Attendant\\$set\(\)](#)
- [Attendant\\$done\(\)](#)
- [Attendant\\$close\(\)](#)
- [Attendant\\$auto\(\)](#)
- [Attendant\\$getMin\(\)](#)
- [Attendant\\$getMax\(\)](#)
- [Attendant\\$getValue\(\)](#)
- [Attendant\\$clone\(\)](#)

Method new():

Usage:

```
Attendant$new(  
  id,  
  min = NULL,  
  max = NULL,  
  session = shiny::getDefaultReactiveDomain(),  
  hide_on_max = FALSE  
)
```

Arguments:

id Id of progress bar set with [attendantBar](#).

min, max Minimum and maximum value of the progress bar.

session A valid shiny session.

hide_on_max Whether to hide the progress bar when it reaches its maximum value (defined in [attendantBar](#)). The progress bar automatically becomes visible again when it is set to a value below the maximum.

Details: Initialise a progress bar

Method inc():*Usage:*

```
Attendant$inc(value = 1, text = NULL)
```

Arguments:

value Value to increase the progress bar.

text Text to display on the progress bar.

Details: Increase

Method dec():*Usage:*

```
Attendant$dec(value = 1, text = NULL)
```

Arguments:

value Value to decrease the progress bar.

text Text to display on the progress bar.

Details: Decrease

Method set():*Usage:*

```
Attendant$set(value, text = NULL)
```

Arguments:

value Value to set the progress bar.

text Text to display on the progress bar.

Details: Set

Method done():*Usage:*

Attendant\$done(text = NULL)

Arguments:

text Text to display on the progress bar.

Details: Done with progress

Method close():

Usage:

Attendant\$close(text = NULL)

Arguments:

text Text to display on the progress bar.

Details: Done with progress

Method auto():

Usage:

Attendant\$auto(ms = 400, value = 1)

Arguments:

ms Milliseconds between increment of value.

value Value to increment by at every ms.

Details: Automatically increase the progress bar until done

Method getMin():

Usage:

Attendant\$getMin()

Details: Get minimum value

Method getMax():

Usage:

Attendant\$getMax()

Details: Get maximum value

Method getValue():

Usage:

Attendant\$getValue()

Details: Get current value

Method clone(): The objects of this class are cloneable with this method.

Usage:

Attendant\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Description

Create a Bootstrap 4 progress bar.

Usage

```
attendantBar(  
  id,  
  value = 0,  
  min = 0,  
  max = 100,  
  text = NULL,  
  color = c("primary", "info", "success", "danger", "warning"),  
  striped = FALSE,  
  animated = FALSE,  
  height = 20,  
  width = "100%",  
  class = "",  
  style = "",  
  bg_color = "#f5f5f5",  
  hidden = FALSE  
)
```

Arguments

id	A unique identifier for the progress bar. Used in Attendant class for handling.
value, min, max	Initial value, minimum, and maximum values the progress bar can take.
text	Optional text to display on the progress bar. This can then be dynamically modified with Attendant.
striped	Whether the progress bar should be striped.
animated	Whether to animate the stripe on the progress bar.
height	Height of the progress bar, numerical values are converted to pixels (px CSS), any other valid CSS size is valid too.
width	Width of the bar, defaults to 100%, numerical values (e.g.: 42) are converted to pixels (px).
class, style	Additional style and class to pass to the parent wrapper of the progress bar.
bg_color, color	Color, and background color of the progress bar.
hidden	Set to TRUE to initialise the attendant as hidden, it will be made visible when set to a value.

autoWaiter

Automatic Waiter

Description

This function allows easily adding waiters to dynamically rendered Shiny content where "dynamic" means render* and *output function pair.

Usage

```
autoWaiter(id = NULL, html = NULL, color = NULL, image = "", fadeout = FALSE)
```

Arguments

id	Vector of ids of elements to overlay the waiter. If NULL then the loading screens are applied to all elements.
html	HTML content of waiter, generally a spinner, see spinners .
color	Background color of loading screen.
image	Path to background image.
fadeout	Use a fade out effect when the screen is removed. Can be a boolean, or a numeric indicating the number of milliseconds the effect should take.

Details

This will display the waiter when the element is being recalculated and hide it when it receives new data.

Examples

```
library(shiny)
library(waiter)

ui <- fluidPage(
  autoWaiter(),
  actionButton(
    "trigger",
    "Render"
  ),
  plotOutput("plot"),
  plotOutput("dom")
)

server <- function(input, output){
  output$plot <- renderPlot({
    input$trigger
    Sys.sleep(3)
    plot(cars)
  })
}
```

```
output$dom <- renderPlot({
  input$trigger
  Sys.sleep(5)
  plot(runif(100))
})
}

if(interactive())
  shinyApp(ui, server)
```

garcon

Garcon

Description

Create a garcon to animate images on the waiter.

Usage

```
useGarcon()
```

```
use_garcon()
```

Methods

Public methods:

- [Garcon\\$new\(\)](#)
- [Garcon\\$set\(\)](#)
- [Garcon\\$inc\(\)](#)
- [Garcon\\$reset\(\)](#)
- [Garcon\\$destroy\(\)](#)
- [Garcon\\$print\(\)](#)
- [Garcon\\$close\(\)](#)
- [Garcon\\$clone\(\)](#)

Method `new()`:

Usage:

```
Garcon$new(
  image,
  bg_color = "#FFFFFF",
  opacity = 0.5,
  direction = c("bt", "tb", "lr", "rl"),
  filter = NULL
)
```

Arguments:

image The CSS id of the image tag.

bg_color Background overlay color in hexadecimal or RGB.

opacity Overlay transparency.

direction Animation direction. Possible values: lr (left to right), rl (right to left), bt (bottom to top), tb (top to bottom).

filter Filter to apply, options are blur, grayscale, sepia, hue-rotate, invert, opacity.

Details: Initialise the garçon.

Examples:

```
\dontrun{Garçon$new("img")$set(30)}
```

Method set():*Usage:*

```
Garçon$set(value)
```

Arguments:

value Percentage to set to.

Details: Value to set the garçon to.

Examples:

```
\dontrun{Garçon$new("img")$set(30)}
```

Method inc():*Usage:*

```
Garçon$inc(value)
```

Arguments:

value Percentage to increase to.

Details: Value to increase the garçon to.

Examples:

```
\dontrun{Garçon$new("img")$inc(30)}
```

Method reset():*Usage:*

```
Garçon$reset(value)
```

Arguments:

value Percentage to set to.

Details: Reset the garçon to.

Examples:

```
\dontrun{Garçon$new("img")$set(30)$reset()}
```

Method destroy():*Usage:*

```
Garçon$destroy()
```

Details: Kill the garçon to.

Examples:

```
\dontrun{Garçon$new("img")$set(30)$destroy()}
```

Method print():

Usage:

```
Garçon$print()
```

Details: print the garçon

Method close():

Usage:

```
Garçon$close()
```

Details: Close the garçon.

Examples:

```
\dontrun{Garçon$new("img")$set(30)$close()}
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Garçon$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## -----
## Method `Garçon$new`
## -----

## Not run: Garçon$new("img")$set(30)

## -----
## Method `Garçon$set`
## -----

## Not run: Garçon$new("img")$set(30)

## -----
## Method `Garçon$inc`
## -----

## Not run: Garçon$new("img")$inc(30)

## -----
## Method `Garçon$reset`
## -----

## Not run: Garçon$new("img")$set(30)$reset()
```

```
## -----
## Method `Garcon$destroy`
## -----

## Not run: Garcon$new("img")$set(30)$destroy()

## -----
## Method `Garcon$close`
## -----

## Not run: Garcon$new("img")$set(30)$close()
```

hostess

Hostess

Description

Add hostess dependencies.

Usage

```
use_hostess()
```

```
useHostess()
```

Methods

Public methods:

- [Hostess\\$new\(\)](#)
- [Hostess\\$start\(\)](#)
- [Hostess\\$print\(\)](#)
- [Hostess\\$set\(\)](#)
- [Hostess\\$inc\(\)](#)
- [Hostess\\$close\(\)](#)
- [Hostess\\$get_loader\(\)](#)
- [Hostess\\$set_loader\(\)](#)
- [Hostess\\$notify\(\)](#)
- [Hostess\\$clone\(\)](#)

Method new():

Usage:

```
Hostess$new(id = NULL, min = 0, max = 100, n = 1, infinite = FALSE)
```

Arguments:

id Id used in hostess_loader if you generate the loader with the loader method you may leave this NULL.

min, max Minimum and maximum representing the starting and ending points of the progress bar.

n Number of loaders to generate.

infinite Set to TRUE to create a never ending loading bar, ideal when you cannot compute increments or assess the time it might take before the loading bar should be removed.

Details: Create a hostess.

Examples:

```
\dontrun{Hostess$new()}
```

Method start():

Usage:

```
Hostess$start()
```

Details: Start the hostess

Method print():

Usage:

```
Hostess$print()
```

Details: Print the hostess

Method set():

Usage:

```
Hostess$set(value)
```

Arguments:

value Value to set, between 0 and 100.

Details: Set the hostess loading bar.

Examples:

```
\dontrun{Hostess$new()$set(20)}
```

Method inc():

Usage:

```
Hostess$inc(value)
```

Arguments:

value Value to set, between 0 and 100.

Details: Increase the hostess loading bar.

Examples:

```
\dontrun{Hostess$new()$inc(10)}
```

Method close():

Usage:

```
Hostess$close()
```

Details: Close the hostess

Examples:

```
\dontrun{Waitress$new("#plot")$close()}
```

Method `get_loader()`:

Usage:

```
Hostess$get_loader(
  preset = NULL,
  text_color = "#FFFFFF",
  center_page = FALSE,
  class = "",
  min = NULL,
  max = NULL,
  svg = NULL,
  progress_type = c("stroke", "fill"),
  fill_direction = c("btt", "ttb", "ltr", "rtl"),
  stroke_direction = c("normal", "reverse"),
  fill_color = NULL,
  stroke_color = NULL,
  ...
)
```

Arguments:

`preset` A loading bar preset, see section below.

`text_color` The color of the loading text.

`center_page` By default the hostess is centered in the middle of the screen, ideal when using it with waiter full screen, set to FALSE to prevent that.

`class` CSS class.

`min`, `max` Minimum and maximum representing the starting and ending points of the progress bar.

`svg` Either an svg path e.g.: M10 10L90 10 or the path to a .svg file. Note that if passing the latter it must be made available to Shiny by placing it either in the www folder or using `shiny::addResourcePath()`.

`progress_type` The progress type, either stroke or fill. The former traces the path of the svg while the latter fills it progressively.

`fill_direction`, `stroke_direction` The direction which the progress bar should take. Whether `fill_direction` or `stroke_direction` is used depends on `progress_type`.

`fill_color`, `stroke_color` The color to use for the progress bar. Whether `fill_color` or `stroke_color` is used depends on `progress_type`.

... Any other other advanced options to pass to the loaded see the [official documentation](#).

Details: Create a hostess loading bar.

Examples:

```
\dontrun{Hostess$new()$get_loader()}
```

Method `set_loader()`:

Usage:

```
Hostess$set_loader(loader)
```

Arguments:

loader Loader as defined by `hostess_loader()`.

Details: Set a hostess loader as defined by `hostess_loader()`.

Examples:

```
\dontrun{
loader <- hostess_loader()
Hostess$new()$set_loader(loader)
}
```

Method `notify()`:*Usage:*

```
Hostess$notify(
  html = NULL,
  background_color = "transparent",
  text_color = "black",
  position = c("br", "tr", "bl", "tl")
)
```

Arguments:

html Additional HTML content of the tag or a character string.

background_color Background color of the notification.

text_color Color of text of html.

position Position of the notification on the screen. Where br is the bottom-right, tr is the top-right, bl is bottom-left, and tl is the top-left.

Details: Use the hostess as a notification. It is hidden when set tpo 100.

Examples:

```
\dontrun{Hostess$new()$notify()}
```

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Hostess$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## -----
## Method `Hostess$new`
## -----

## Not run: Hostess$new()

## -----
## Method `Hostess$set`
## -----
```

```
## Not run: Hostess$new()$set(20)

## -----
## Method `Hostess$inc`
## -----

## Not run: Hostess$new()$inc(10)

## -----
## Method `Hostess$close`
## -----

## Not run: Waitress$new("#plot")$close()

## -----
## Method `Hostess$get_loader`
## -----

## Not run: Hostess$new()$get_loader()

## -----
## Method `Hostess$set_loader`
## -----

## Not run:
loader <- hostess_loader()
Hostess$new()$set_loader(loader)

## End(Not run)

## -----
## Method `Hostess$notify`
## -----

## Not run: Hostess$new()$notify()
```

hostessLoader

Loader

Description

Customise the Hostess loading bar.

Usage

```
hostess_loader(
  id = "hostess",
  preset = NULL,
  text_color = "#FFFFFF",
  center_page = FALSE,
```

```

    class = "",
    min = 0,
    max = 100,
    svg = NULL,
    progress_type = c("stroke", "fill"),
    fill_direction = c("btt", "ttb", "ltr", "rtl"),
    stroke_direction = c("normal", "reverse"),
    fill_color = NULL,
    stroke_color = NULL,
    ...
)

hostess_gradient(angle = 0, duration = 1, colors = c("red", "white", "blue"))

hostess_bubble(
  color_background = "#697682",
  color_bubble = "#f7fff7",
  count = 25,
  duration = 1
)

hostess_stripe(color1 = "#697682", color2 = "#f7fff7", duration = 1)

```

Arguments

id	Id of hostess (valid CSS).
preset	A loading bar preset, see section below.
text_color	The color of the loading text.
center_page	By default the hostess is <i>not</i> centered in the middle of the screen, centering in the middle of the page is however ideal when using it with waiter full screen, for the latter set to TRUE.
class	CSS class.
min, max	Minimum and maximum representing the starting and ending points of the progress bar.
svg	Either an svg path e.g.: M10 10L90 10 or the path to a .svg file. Note that if passing the latter it must be made available to Shiny by placing it either in the www folder or using <code>shiny::addResourcePath()</code> .
progress_type	The progress type, either stroke or fill. Ther former traces the path of the svg while the latter fills it progressively.
fill_direction, stroke_direction	The direction which the progress bar should take. Wether fill_direction or stroke_direction is used depends on progress_type.
fill_color, stroke_color	The color to use for the progress bar. Wether fill_color or stroke_color is used depends on progress_type.

...	Any other other advanced options to pass to the loaded see the official documentation .
angle	Angle of gradient.
duration	Duration of the loop.
colors	Color vectors composing the gradient.
color_background	The background of the color.
color_bubble	The color of the bubbles contour.
count	The number of bubbles.
color1, color2	Colors of stripes.

Presets

- line
- fan
- circle
- bubble
- rainbow
- energy
- stripe
- text

Examples

```
library(shiny)
library(waiter)

# diagonal line
path <- "M10 10L90 30"

ui <- fluidPage(
  useWaiter(),
  useHostess(),
  actionButton("draw", "redraw"),
  plotOutput("plot")
)

server <- function(input, output) {

  dataset <- reactive({
    input$draw

    hostess <- Hostess$new(min = 0, max = 10)
    hostess$set_loader <- hostess_loader(
      progress_type = "stroke",
      stroke_color = hostess_stripe()
    )
  })
}
```

```

    waiter <- Waiter$new(
      "plot",
      hostess$loader()
    )

    waiter$show()

    for(i in 1:10){
      Sys.sleep(.2)
      hostess$inc(1)
    }

    runif(100)

  })

output$plot <- renderPlot(plot(dataset()))

}

if(interactive()) shinyApp(ui, server)

```

httr_progress

Waitress with httr

Description

Use a waitress progress bar with httr requests. Simply use `httr_progress` where you would use [httr::progress](#).

Usage

```
httr_progress(object, type = c("down", "up"), pre = NULL, post = NULL)
```

Arguments

<code>object</code>	The waitress or attendant object.
<code>type</code>	Type of progress to display: either number of bytes uploaded or downloaded. Passed to httr::progress .
<code>pre, post</code>	Pre and callback functions to run before the progress starts or once it is done.

Examples

```

## Not run:
cap_speed <- httr::config(max_recv_speed_large = 10000)

httr::GET(
  "http://httpbin.org/bytes/102400",

```

```
      htr_progress(w),
      cap_speed
    )

## End(Not run)
```

preview_spinner	<i>Preview spinner</i>
-----------------	------------------------

Description

Allows previewing spinners in web browser or RStudio Viewer.

Usage

```
preview_spinner(spinner, bg_color = "black")
```

Arguments

- spinner A waiter link{spinner}.
- bg_color Background color.

Examples

```
if(interactive()) preview_spinner(spin_1())
```

spinners	<i>Spinners</i>
----------	-----------------

Description

Spinkit spinners to use with [waiter_show](#).

Usage

```
spin_rotating_plane()

spin_fading_circles()

spin_folding_cube()

spin_double_bounce()

spin_wave()
```

`spin_wandering_cubes()`
`spin_pulse()`
`spin_chasing_dots()`
`spin_three_bounce()`
`spin_circle()`
`spin_rotate()`
`spin_solar()`
`spin_orbit()`
`spin_squares()`
`spin_cube_grid()`
`spin_circles()`
`spin_orbiter()`
`spin_pixel()`
`spin_flower()`
`spin_dual_ring()`
`spin_heart()`
`spin_ellipsis()`
`spin_facebook()`
`spin_hourglass()`
`spin_ring()`
`spin_ripple()`
`spin_terminal()`
`spin_loader()`
`spin_throbber()`

`spin_refresh()`
`spin_heartbeat()`
`spin_gauge()`
`spin_3k()`
`spin_wobblebar()`
`spin_atebits()`
`spin_whirly()`
`spin_flowers()`
`spin_dots()`
`spin_3circles()`
`spin_plus()`
`spin_pulsar()`
`spin_hexdots()`
`spin_inner_circles()`
`spin_pong()`
`spin_timer()`
`spin_ball()`
`spin_dual_circle()`
`spin_seven_circle()`
`spin_clock()`
`spin_pushing_shapes()`
`spin_fill()`
`spin_rhombus()`
`spin_balance()`

```
spin_square_circle()

spin_circle_square()

spin_puzzle()

spin_half()

spin_loaders(id = 1, color = "white", style = NULL)

spin_1()

spin_2()

spin_3()

spin_4()

spin_5()

spin_6()

bs4_spinner(
  style = c("spin", "grow"),
  color = c("primary", "secondary", "success", "danger", "warning", "info", "light",
    "dark")
)

bs5_spinner(
  style = c("spin", "grow"),
  color = c("primary", "secondary", "success", "danger", "warning", "info", "light",
    "dark")
)

spin_google()
```

Arguments

id	The spinner identifier, an integer between 1, and 42.
color	Desired color of spinner.
style	CSS style to apply to spinner.

Details

Much of the CSS used is to provide those spinners. One can greatly reduce the load on the browser by only sourcing the CSS for the spinners required. You can find out which CSS kits are required to load by using the spinner in the R console as shown in the example. This prints the kit and instructions to only source the required file.

Value

An object of class spinner.

Examples

```
spin_rotating_plane()
```

steward	<i>Steward</i>
---------	----------------

Description

A colorful steward to work with the [waiter](#).

Usage

```
useSteward(  
  colors = c("#ee7752", "#e73c7e", "#23a6d5", "#23d5ab"),  
  speed = 30,  
  angle = -45  
)  
  
use_steward(  
  colors = c("#ee7752", "#e73c7e", "#23a6d5", "#23d5ab"),  
  speed = 30,  
  angle = -45  
)
```

Arguments

- colors Color palette forming gradient.
- speed Seconds it takes to loop over colors.
- angle Degrees at which colors slide.

transparent	<i>Transparency</i>
-------------	---------------------

Description

A convenience function to create a waiter with transparent background.

Usage

```
transparent(alpha = 0)
```

Arguments

alpha Alpha channel where 0 is completely transparent and 1 is opaque.

Examples

```
transparent()
```

triggerWaiter

Trigger Waiter

Description

A a trigger to a waiting screen from the UI.

Usage

```
triggerWaiter(
  el,
  id = NULL,
  html = NULL,
  color = NULL,
  image = "",
  fadeout = FALSE,
  on = "click",
  hide_on_render = !is.null(id),
  hide_on_error = !is.null(id),
  hide_on_silent_error = !is.null(id)
)
```

Arguments

el	Element that triggers the waiter.
id	Id of element to hide or element on which to show waiter over.
html	HTML content of waiter, generally a spinner, see spinners .
color	Background color of loading screen.
image	Path to background image.
fadeout	Use a fade out effect when the screen is removed. Can be a boolean, or a numeric indicating the number of milliseconds the effect should take.
on	The event that triggers the waiter.
hide_on_render	Set to TRUE to automatically hide the waiter when the plot in id is drawn. Note the latter will only work with shiny plots, tables, htmlwidgets, etc. but will not work with arbitrary elements.
hide_on_error, hide_on_silent_error	Whether to hide the waiter when the underlying element throws an error. Silent error are thrown by req and validate .

Examples

```
library(shiny)
library(waiter)

ui <- fluidPage(
  useWaiter(),
  triggerWaiter(
    actionButton(
      "generate",
      "Generate Plot"
    )
  ),
  plotOutput("plot")
)

server <- function(input, output){
  output$plot <- renderPlot({
    input$generate
    Sys.sleep(3)
    plot(runif(50))
  })
}

if(interactive())
  shinyApp(ui, server)
```

useAttendant	<i>Attendant Progress Dependencies</i>
--------------	--

Description

Include in anywhere your shiny UI to import the dependencies required to run attendant progress.

Usage

```
useAttendant()
```

waiter	<i>Waiter</i>
--------	---------------

Description

Programatically show and hide loading screens.

Usage

```
use_waiter(spinners = NULL, include_js = TRUE)

useWaiter(spinners = NULL, include_js = TRUE)

waiter_use(spinners = 1:7, include_js = TRUE)

waiter_show(
  id = NULL,
  html = spin_1(),
  color = "#333e48",
  logo = "",
  image = "",
  hide_on_render = !is.null(id)
)

waiter_show_on_load(html = spin_1(), color = "#333e48", image = "", logo = "")

waiterShowOnLoad(html = spin_1(), color = "#333e48", image = "", logo = "")

waiter_preloader(
  html = spin_1(),
  color = "#333e48",
  image = "",
  fadeout = FALSE,
  logo = ""
)

waiterPreloader(
  html = spin_1(),
  color = "#333e48",
  image = "",
  fadeout = FALSE,
  logo = ""
)

waiter_hide_on_render(id)

waiterHideOnRender(id)

waiter_on_busy(
  html = spin_1(),
  color = "#333e48",
  logo = "",
  image = "",
  fadeout = FALSE
)
```

```
waiterOnBusy(
  html = spin_1(),
  color = "#333e48",
  logo = "",
  image = "",
  fadeout = FALSE
)

waiter_hide(id = NULL)

waiter_update(id = NULL, html = NULL)
```

Arguments

<code>spinners</code>	Deprecated argument. Spinners to include. By default all the CSS files for all spinners are included you can customise this only that which you need in order to reduce the amount of CSS that needs to be loaded and improve page loading speed. There are 7 spinner kits. The spinner kit required for the spinner you use is printed in the R console when using the spinner. You can specify a single spinner kit e.g.: 1 or multiple spinner kits as a vector e.g.: <code>c(1, 3, 6)</code> .
<code>include_js</code>	Deprecated argument, no longer needed.
<code>id</code>	Id of element to hide or element on which to show waiter over.
<code>html</code>	HTML content of waiter, generally a spinner, see spinners .
<code>color</code>	Background color of loading screen.
<code>logo</code>	Path to logo to display. Deprecated.
<code>image</code>	Path to background image.
<code>hide_on_render</code>	Set to TRUE to automatically hide the waiter when the plot in <code>id</code> is drawn. Note the latter will only work with shiny plots, tables, <code>htmlwidgets</code> , etc. but will not work with arbitrary elements.
<code>fadeout</code>	Use a fade out effect when the screen is removed. Can be a boolean, or a numeric indicating the number of milliseconds the effect should take.

Functions

- `use_waiter` and `waiter_use`: waiter dependencies to include anywhere in your UI but ideally at the top.
- `waiter_show_on_load`: Show a waiter on page load, before the session is even loaded, include in UI *after* `use_waiter`.
- `waiter_show`: Show waiting screen.
- `waiter_hide`: Hide any waiting screen.
- `waiter_on_busy`: Automatically shows the waiting screen when the server is busy, and hides it when it goes back to idle.
- `waiter_update`: Update the content `html` of the waiting screen.
- `waiter_hide_on_render`: Hide any waiting screen when the output is drawn, useful for outputs that take a long time to draw, *use in ui*.

- `waiter_preloader`: Shows the waiter on load and automatically removes it once all the UI is rendered, only runs on the first load of the app.

Examples

```
library(shiny)

ui <- fluidPage(
  useWaiter(), # dependencies
  waiterShowOnLoad(spin_fading_circles()), # shows before anything else
  actionButton("show", "Show loading for 5 seconds")
)

server <- function(input, output, session){
  waiter_hide() # will hide *on_load waiter

  observeEvent(input$show, {
    waiter_show(
      html = tagList(
        spin_fading_circles(),
        "Loading ..."
      )
    )
    Sys.sleep(3)
    waiter_hide()
  })
}

if(interactive()) shinyApp(ui, server)
```

waiterClass

Waiter R6 Class

Description

Create a waiter to then show, hide or update its content.

Details

Create an object to show a waiting screen on either the entire application or just a portion of the app by specifying the id. Then show, then hide or meanwhile update the content of the waiter.

Active bindings

`fadeout` Set or get the fade out
`color` Set or get the background color
`image` Set of get the background image
`session` Set or get the shiny session
`html` Set or get the html content

Methods

Public methods:

- `Waiter$new()`
- `Waiter$show()`
- `Waiter$hide()`
- `Waiter$update()`
- `Waiter$print()`
- `Waiter$clone()`

Method `new()`:

Usage:

```
Waiter$new(
  id = NULL,
  html = NULL,
  color = NULL,
  logo = NULL,
  image = "",
  fadeout = FALSE,
  hide_on_render = !is.null(id),
  hide_on_error = !is.null(id),
  hide_on_silent_error = !is.null(id)
)
```

Arguments:

`id` Id, or vector of ids, of element on which to overlay the waiter, if NULL the waiter is applied to the entire body.

`html` HTML content of waiter, generally a spinner, see [spinners](#) or a list of the latter.

`color` Background color of loading screen.

`logo` Logo to display. Deprecated.

`image` Path to background image of loading screen.

`fadeout` Use a fade out effect when the screen is removed. Can be a boolean, or a numeric indicating the number of milliseconds the effect should take.

`hide_on_render` Set to TRUE to automatically hide the waiter when the element in `id` is drawn. Note the latter will work with shiny plots, tables, `htmlwidgets`, etc. but will not work with arbitrary elements.

`hide_on_error`, `hide_on_silent_error` Whether to hide the waiter when the underlying element throws an error. Silent error are thrown by [req](#) and [validate](#).

Details: Create a waiter.

Examples:

```
\dontrun{Waiter$new()}
```

Method `show()`:

Usage:

```
Waiter$show()
```

Details: Show the waiter.

Method hide():

Usage:

Waiter\$hide()

Details: Hide the waiter.

Method update():

Usage:

Waiter\$update(html = NULL)

Arguments:

html HTML content of waiter, generally a spinner, see [spinners](#).

Details: Update the waiter's html content.

Method print():

Usage:

Waiter\$print()

Details: print the waiter

Method clone(): The objects of this class are cloneable with this method.

Usage:

Waiter\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Examples

```
## -----
## Method `Waiter$new`
## -----

## Not run: Waiter$new()
```

waiterTheme

Define a Theme

Description

Define a theme to be used by all waiter loading screens. These can be overridden in individual loading screens.

Usage

```
waiter_set_theme(html = spin_1(), color = "#333e48", logo = "", image = "")

waiter_get_theme()

waiter_unset_theme()
```

Arguments

html	HTML content of waiter, generally a spinner, see spinners .
color	Background color of loading screen.
logo	Path to logo to display. Deprecated.
image	Path to background image.

waitress*Waitress*

Description

Programatically show and hide loading bars.

Usage

```
useWaitress(color = "#697682", percent_color = "#333333")

use_waitress(color = "#697682", percent_color = "#333333")
```

Arguments

color, percent_color	Color of waitress and color of percent text shown when theme is set to overlay-percent.
----------------------	---

Details

You can pipe the methods with `$. Waitress$new()` and `call_waitress()` are equivalent.

Examples

```
library(shiny)

ui <- fluidPage(
  useWaitress("red"), # dependencies
  sliderInput("set", "percentage", 1, 100, step = 5, value = 1)
)

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

```

w <- Waitress$
  new()$ # call a waitress
  start() # start waitress

observeEvent(input$set, {
  w$set(input$set) # set at percentage
})
}

if(interactive()) shinyApp(ui, server)

```

waitressClass

Waitress R6 Class

Description

Create a waitress (progress bar) and programmatically set or increase its percentage, then hide it when done.

Active bindings

max Maximum value of the bar.

min Minimum value of the bar.

Methods

Public methods:

- [Waitress\\$new\(\)](#)
- [Waitress\\$start\(\)](#)
- [Waitress\\$notify\(\)](#)
- [Waitress\\$set\(\)](#)
- [Waitress\\$auto\(\)](#)
- [Waitress\\$inc\(\)](#)
- [Waitress\\$close\(\)](#)
- [Waitress\\$getMin\(\)](#)
- [Waitress\\$getMax\(\)](#)
- [Waitress\\$getValue\(\)](#)
- [Waitress\\$print\(\)](#)
- [Waitress\\$clone\(\)](#)

Method `new()`:

Usage:

```
Waitress$new(
  selector = NULL,
  theme = c("line", "overlay", "overlay-radius", "overlay-opacity", "overlay-percent"),
  min = 0,
  max = 100,
  infinite = FALSE,
  hide_on_render = FALSE
)
```

Arguments:

selector Element selector to apply the waitress to, if NULL then the waitress is applied to the whole screen.

theme A valid theme, see function usage.

min, max Minimum and maximum representing the starting and ending points of the progress bar.

infinite Set to TRUE to create a never ending loading bar, ideal when you cannot compute increments or assess the time it might take before the loading bar should be removed.

hide_on_render Set to TRUE to automatically hide the waitress when the element in id is rendered. Note the latter will work with shiny plots, tables, htmlwidgets, etc. but will not work with arbitrary elements.

color, percent_color Color of waitress and color of percent text shown when theme is set to overlay-percent.

Details: Create a waitress.

Examples:

```
\dontrun{Waitress$new("#plot")}
```

Method start():

Usage:

```
Waitress$start(
  html = NULL,
  background_color = "transparent",
  text_color = "black"
)
```

Arguments:

html HTML content to show over the waitress, accepts htmltools and shiny tags.

background_color The background color of the html.

text_color The color of the html content.

Details: Start the waitress.

Examples:

```
\dontrun{Waitress$new("#plot")$start()}
```

Method notify():

Usage:

```
Waitress$notify(
  html = NULL,
  background_color = "white",
  text_color = "black",
  position = c("br", "tr", "bl", "tl")
)
```

Arguments:

html HTML content to show over the waitress, accepts htmltools and shiny tags.

background_color The background color of the html.

text_color The color of the html content.

position Position of the notification on the screen. Where br is the bottom-right, tr is the top-right, bl is bottom-left, and tl is the top-left.

Details: Show the waitress as a notification.

Examples:

```
\dontrun{Waitress$new()$notify()}
```

Method set():

Usage:

```
Waitress$set(value)
```

Arguments:

value Value to set waitress to.

Details: Set the waitress to a specific percentage.

Examples:

```
\dontrun{Waitress$new("#plot")$set(20)}
```

Method auto():

Usage:

```
Waitress$auto(value, ms)
```

Arguments:

value Value to set waitress to.

ms Number of Milliseconds

Details: Automatically start and end the waitress.

Examples:

```
\dontrun{Waitress$new("#plot")$auto(20, 2000)}
```

Method inc():

Usage:

```
Waitress$inc(value)
```

Arguments:

value Value to increase waitress to.

Details: Increase the waitress by a percentage.

Examples:

```
\dontrun{Waitress$new("#plot")$inc(30)}
```

Method close():

Usage:

```
Waitress$close()
```

Details: Close the waitress.

Examples:

```
\dontrun{Waitress$new("#plot")$close()}
```

Method getMin():

Usage:

```
Waitress$getMin()
```

Details: Get minimum value

Method getMax():

Usage:

```
Waitress$getMax()
```

Details: Get maximum value

Method getValue():

Usage:

```
Waitress$getValue()
```

Details: Get current value

Method print():

Usage:

```
Waitress$print()
```

Details: Print the waitress.

Examples:

```
\dontrun{Waitress$new("#plot")$hide()}
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
Waitress$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```

## -----
## Method `Waitress$new`
## -----

## Not run: Waitress$new("#plot")

## -----
## Method `Waitress$start`
## -----

## Not run: Waitress$new("#plot")$start()

## -----
## Method `Waitress$notify`
## -----

## Not run: Waitress$new()$notify()

## -----
## Method `Waitress$set`
## -----

## Not run: Waitress$new("#plot")$set(20)

## -----
## Method `Waitress$auto`
## -----

## Not run: Waitress$new("#plot")$auto(20, 2000)

## -----
## Method `Waitress$inc`
## -----

## Not run: Waitress$new("#plot")$inc(30)

## -----
## Method `Waitress$close`
## -----

## Not run: Waitress$new("#plot")$close()

## -----
## Method `Waitress$print`
## -----

## Not run: Waitress$new("#plot")$hide()

```

Description

Report progress with attendant.

Usage

```
withProgressAttendant(
  expr,
  ...,
  session = getDefaultReactiveDomain(),
  env = parent.frame(),
  quoted = FALSE
)

setProgressAttendant(
  value = 1,
  text = NULL,
  session = getDefaultReactiveDomain()
)

incProgressAttendant(
  value = 1,
  text = NULL,
  session = getDefaultReactiveDomain()
)
```

Arguments

<code>expr</code>	The work to be done. This expression should contain calls to <code>setProgressAttendant</code> or <code>incProgressAttendant</code> .
<code>...</code>	Passed to the <code>Attendant</code> constructor (<code>Attendant\$new()</code>).
<code>session</code>	The Shiny session object, as provided by <code>shinyServer</code> to the server function. The default is to automatically find the session by using the current reactive domain.
<code>env</code>	The environment in which <code>expr</code> should be evaluated.
<code>quoted</code>	Whether <code>expr</code> is a quoted expression (this is not common).
<code>value</code>	Value to set the waitress to or increase it by.
<code>text</code>	Text to display on the progress bar.

`withProgressWaitress` *Report Progress Waitress*

Description

Report progress with waitress.

Usage

```
withProgressWaitress(
  expr,
  ...,
  session = getDefaultReactiveDomain(),
  env = parent.frame(),
  quoted = FALSE
)

setProgressWaitress(value = 1, session = getDefaultReactiveDomain())

incProgressWaitress(value = 1, session = getDefaultReactiveDomain())
```

Arguments

<code>expr</code>	The work to be done. This expression should contain calls to <code>setProgressWaitress</code> or <code>incProgressWaitress</code> .
<code>...</code>	Passed to the <code>Waitress</code> constructor (<code>Waitress\$new()</code>).
<code>session</code>	The Shiny session object, as provided by <code>shinyServer</code> to the server function. The default is to automatically find the session by using the current reactive domain.
<code>env</code>	The environment in which <code>expr</code> should be evaluated.
<code>quoted</code>	Whether <code>expr</code> is a quoted expression (this is not common).
<code>value</code>	Value to set the waitress to or increase it by.

withWaiter

*With Waiter***Description**

Adds a waiter to a reactive UI element. The waiter is displayed when the element is invalidated then is removed when the element receives a new value.

Usage

```
withWaiter(element, html = spin_1(), color = "#333e48", image = "")
```

Arguments

<code>element</code>	A reactive element, e.g.: <code>uiOutput</code> , or <code>plotOutput</code> .
<code>html</code>	HTML content of waiter, generally a spinner, see spinners .
<code>color</code>	Background color of loading screen.
<code>image</code>	Path to background image.

Index

Attendant, 2
attendantBar, 3, 5
autoWaiter, 6

bs4_spinner (spinners), 18
bs5_spinner (spinners), 18

Garcon (garcon), 7
garcon, 7

Hostess (hostess), 10
hostess, 10
hostess_bubble (hostessLoader), 14
hostess_gradient (hostessLoader), 14
hostess_loader (hostessLoader), 14
hostess_loader(), 13
hostess_stripe (hostessLoader), 14
hostessLoader, 14
httr::progress, 17
httr_progress, 17

incProgressAttendant
 (withProgressAttendant), 35
incProgressWaitress
 (withProgressWaitress), 36

preview_spinner, 18

req, 23, 28

setProgressAttendant
 (withProgressAttendant), 35
setProgressWaitress
 (withProgressWaitress), 36
shiny::addResourcePath(), 12, 15
spin_1 (spinners), 18
spin_2 (spinners), 18
spin_3 (spinners), 18
spin_3circles (spinners), 18
spin_3k (spinners), 18
spin_4 (spinners), 18
spin_5 (spinners), 18
spin_6 (spinners), 18
spin_atebits (spinners), 18
spin_balance (spinners), 18
spin_ball (spinners), 18
spin_chasing_dots (spinners), 18
spin_circle (spinners), 18
spin_circle_square (spinners), 18
spin_circles (spinners), 18
spin_clock (spinners), 18
spin_cube_grid (spinners), 18
spin_dots (spinners), 18
spin_double_bounce (spinners), 18
spin_dual_circle (spinners), 18
spin_dual_ring (spinners), 18
spin_ellipsis (spinners), 18
spin_facebook (spinners), 18
spin_fading_circles (spinners), 18
spin_fill (spinners), 18
spin_flower (spinners), 18
spin_flowers (spinners), 18
spin_folding_cube (spinners), 18
spin_gauge (spinners), 18
spin_google (spinners), 18
spin_half (spinners), 18
spin_heart (spinners), 18
spin_heartbeat (spinners), 18
spin_hexdots (spinners), 18
spin_hourglass (spinners), 18
spin_inner_circles (spinners), 18
spin_loader (spinners), 18
spin_loaders (spinners), 18
spin_orbit (spinners), 18
spin_orbiter (spinners), 18
spin_pixel (spinners), 18
spin_plus (spinners), 18
spin_pong (spinners), 18
spin_pulsar (spinners), 18
spin_pulse (spinners), 18

`spin_pushing_shapes` (spinners), 18
`spin_puzzle` (spinners), 18
`spin_refresh` (spinners), 18
`spin_rhombus` (spinners), 18
`spin_ring` (spinners), 18
`spin_ripple` (spinners), 18
`spin_rotate` (spinners), 18
`spin_rotating_plane` (spinners), 18
`spin_seven_circle` (spinners), 18
`spin_solar` (spinners), 18
`spin_square_circle` (spinners), 18
`spin_squares` (spinners), 18
`spin_terminal` (spinners), 18
`spin_three_bounce` (spinners), 18
`spin_throbber` (spinners), 18
`spin_timer` (spinners), 18
`spin_wandering_cubes` (spinners), 18
`spin_wave` (spinners), 18
`spin_whirly` (spinners), 18
`spin_wobblebar` (spinners), 18
spinners, 6, 18, 23, 26, 28–30, 37
steward, 22

transparent, 22
triggerWaiter, 23

`use_garcon` (garcon), 7
`use_hostess` (hostess), 10
`use_steward` (steward), 22
`use_waiter` (waiter), 24
`use_waitress` (waitress), 30
useAttendant, 24
`useGarcon` (garcon), 7
`useHostess` (hostess), 10
`useSteward` (steward), 22
`useWaiter` (waiter), 24
`useWaitress` (waitress), 30

validate, 23, 28

Waiter (waiterClass), 27
waiter, 22, 24
`waiter_get_theme` (waiterTheme), 29
`waiter_hide` (waiter), 24
`waiter_hide_on_render` (waiter), 24
`waiter_on_busy` (waiter), 24
`waiter_preloader` (waiter), 24
`waiter_set_theme` (waiterTheme), 29
`waiter_show`, 18
`waiter_show` (waiter), 24
`waiter_show_on_load` (waiter), 24
`waiter_unset_theme` (waiterTheme), 29
`waiter_update` (waiter), 24
`waiter_use` (waiter), 24
waiterClass, 27
`waiterHideOnRender` (waiter), 24
`waiterOnBusy` (waiter), 24
`waiterPreloader` (waiter), 24
`waiterShowOnLoad` (waiter), 24
waiterTheme, 29
Waitress (waitressClass), 31
waitress, 30
waitressClass, 31
withProgressAttendant, 35
withProgressWaitress, 36
withWaiter, 37