Package ‘shinybusy’

October 29, 2019

Title Busy Indicator for 'Shiny' Applications

Version 0.2.0

Description Add a global indicator (spinner, progress bar, gif) in your 'shiny' applications to show the user that the server is busy.

License GPL-3 | file LICENSE

Encoding UTF-8

LazyData true

Imports htmltools, shiny, jsonlite, htmlwidgets

RoxygenNote 6.1.1

URL `https://github.com/dreamRs/shinybusy`

BugReports `https://github.com/dreamRs/shinybusy/issues`

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Fanny Meyer [aut],
Victor Perrier [aut, cre],
Silex Technologies [fnd] (https://www.silex-ip.com),
Tobias Ahlin [cph] (SpinKit CSS),
Epicmax [cph] (Epic-spinners CSS),
Chris Antonellis [cph] (freezeframe.js),
Jacobo Tabernero [cph] (nanobar.js),
Kimmo Brunfeldt [cph] (progressbar.js)

Maintainer Victor Perrier `<victor.perrier@dreamrs.fr>`

Repository CRAN

Date/Publication 2019-10-29 06:30:03 UTC

R topics documented:

add_busy_bar 2
add_busy_gif 3

add_busy_spinner 4
logo_silex 6
manual-gif 6
manual-progressbar 8
manual-spinner 9
modal-gif 11
modal-progress 12
modal-spinner 14
progress 16
spin_epic 17
spin_kit 18

Index **20**

`add_busy_bar` *Automatic busy indicator (Progress bar)*

Description

Make a progress bar appear on top of the page.

Usage

```
add_busy_bar(timeout = 1000, color = "#112446", centered = FALSE,
  height = "8px")
```

Arguments

`timeout`	Number of milliseconds after the server is busy to display the progress bar.
`color`	Progress bar color.
`centered`	Center the progress bar or not.
`height`	Height of the bar.

Examples

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

  ui <- fluidPage(

    # Use this function somewhere in UI
    add_busy_bar(color = "#FF0000"),

    headerPanel('Iris k-means clustering'),

    tags$br(),
    actionButton("quick", "Quick calculation (nothing happens)"),
    actionButton("sleep", "Long calculation (progress bar on top)")
  )

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

    observeEvent(input$quick, {
      Sys.sleep(0.1)
    })

    observeEvent(input$sleep, {
      Sys.sleep(5)
    })

  }

  shinyApp(ui, server)
}
```

`add_busy_gif` *Automatic busy indicator (GIF)*

Description

Make a GIF play when server is busy and stop when idle.

Usage

```
add_busy_gif(src, timeout = 100, position = c("top-right", "top-left",
  "bottom-right", "bottom-left", "full-page", "free"), margins = c(10,
  10), overlay_color = "rgba(0, 0, 0, 0.5)", overlay_css = NULL,
  height = "50px", width = "50px")
```

Arguments

`src`	Path to the GIF, an URL or a file in www/ folder.
`timeout`	Number of milliseconds after the server is busy to display the Gif
`position`	Where to display the spinner: `'top-right'`, `'top-left'`, `'bottom-right'`, `'bottom-left'`, `'full-page'`.
`margins`	Distance from margins, a vector of length two, where first element is distance from top/bottom, second element distance from right/left.
`overlay_color`	Background color for the overlay if `position = "full-page"`.
`overlay_css`	Additional CSS for the overlay, for example `"z-index: 1000;"` to make it appear of everything.
`height, width`	Height and width of the spinner, default to `'50px'` for both, must be specified.

Examples

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

  ui <- fluidPage(

    # Use this function somewhere in UI
    add_busy_gif(
      src = "https://jeroen.github.io/images/banana.gif",
      height = 70, width = 70
    ),

    actionButton("sleep", "Long calculation")
  )

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

    observeEvent(input$sleep, {
      Sys.sleep(5)
    })

  }

  shinyApp(ui, server)
}
```

---

add_busy_spinner *Automatic busy indicator (spinner)*

Description

Add a spinner in an application each time the server take more 100 milliseconds to respond.

Usage

```
add_busy_spinner(spin = "double-bounce", color = "#112446",
  timeout = 100, position = c("top-right", "top-left", "bottom-right",
  "bottom-left", "full-page"), onstart = TRUE, margins = c(10, 10),
  height = "50px", width = "50px")
```

Arguments

spin	Style of the spinner, see spin_epic or spin_kit for possible choices. Note that for spin_epic, height and width are ignored.
color	Color for the spinner, in a valid CSS format.
timeout	Number of milliseconds after the server is busy to display the spinner.

`position`	Where to display the spinner: `'top-right'`, `'top-left'`, `'bottom-right'`, `'bottom-left'`, `'full-page'`.
`onstart`	Logical, display the spinner when the application starts ?
`margins`	Distance from margins, a vector of length two, where first element is distance from top/bottom, second element distance from right/left.
`height, width`	Height and width of the spinner, default to `'50px'` for both, must be specified.

Examples

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

  ui <- fluidPage(

    # Use this function somewhere in UI
    add_busy_spinner(spin = "double-bounce"),
    # or use a different spinner
    # add_busy_spinner(spin = "radar", margins = c(10, 20)),

    headerPanel('Iris k-means clustering'),

    sidebarLayout(
      sidebarPanel(
        selectInput('xcol', 'X Variable', names(iris)),
        selectInput('ycol', 'Y Variable', names(iris),
                    selected=names(iris)[[2]]),
        numericInput('clusters', 'Cluster count', 3,
                     min = 1, max = 9),
        actionButton("sleep", "Long calculation")
      ),
      mainPanel(
        plotOutput('plot1')
      )
    )
  )

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

    selectedData <- reactive({
      iris[, c(input$xcol, input$ycol)]
    })

    clusters <- reactive({
      kmeans(selectedData(), input$clusters)
    })

    output$plot1 <- renderPlot({
      palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
                "#FF7F00", "#FFFF33", "#A65628", "#F781BF",
                "#999999"))
      par(mar = c(5.1, 4.1, 0, 1))
      plot(selectedData(),
           col = clusters()$cluster,
           pch = 20, cex = 3)
      points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
    })

    observeEvent(input$sleep, {
      Sys.sleep(5)
    })

  }

  shinyApp(ui, server)
}
```

logo_silex *Silex logo for Shiny use*

Description

Silex logo for Shiny use

Usage

```
logo_silex()
```

Value

Path to gif

manual-gif *Manual busy indicator (GIF)*

Description

Manual busy indicator (GIF)

Usage

```
use_busy_gif(src, timeout = 100, position = c("top-right", "top-left",
  "bottom-right", "bottom-left", "full-page", "free"), margins = c(10,
  10), overlay_color = "rgba(0, 0, 0, 0.5)", overlay_css = NULL,
  height = "50px", width = "50px")

play_gif(session = shiny::getDefaultReactiveDomain())

stop_gif(session = shiny::getDefaultReactiveDomain())
```

Arguments

`src`	Path to the GIF, an URL or a file in www/ folder.
`timeout`	Number of milliseconds after the server is busy to display the Gif
`position`	Where to display the spinner: `'top-right'`, `'top-left'`, `'bottom-right'`, `'bottom-left'`, `'full-page'`.
`margins`	Distance from margins, a vector of length two, where first element is distance from top/bottom, second element distance from right/left.
`overlay_color`	Background color for the overlay if `position = "full-page"`.
`overlay_css`	Additional CSS for the overlay, for example `"z-index: 1000;"` to make it appear of everything.
`height`	Height and width of the spinner, default to `'50px'` for both, must be specified.
`width`	Height and width of the spinner, default to `'50px'` for both, must be specified.
`session`	Shiny session.

Examples

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

  ui <- fluidPage(

    # Use this function somewhere in UI
    use_busy_gif(
      src = "https://jeroen.github.io/images/banana.gif",
      height = 70, width = 70
    ),

    actionButton("play", "Play GIF"),
    actionButton("stop", "Stop GIF")
  )

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

    observeEvent(input$play, {
      play_gif()
    })

    observeEvent(input$stop, {
      stop_gif()
    })

  }

  shinyApp(ui, server)
}
```

manual-progressbar *Manual busy indicator (progress bar)*

Description

Declare `use_busy_bar` in your UI and update value server-side with `update_busy_bar`.

Usage

```
use_busy_bar(color = "#112446", centered = FALSE, height = "8px")

update_busy_bar(value, session = shiny::getDefaultReactiveDomain())
```

Arguments

color	Progress bar color.
centered	Center the progress bar or not.
height	Height of the bar.
value	The new value for the progress bar.
session	Shiny session.

Examples

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

  ui <- fluidPage(
    tags$h2("Manual nanobar"),
    use_busy_bar(color = "#01DF01", height = "15px"),
    actionButton(inputId = "go", label = "Go"),
    sliderInput(
      inputId = "set", label = "Set progress",
      min = 0, value = 0, max = 100
    )
  )

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

    observeEvent(input$go, {
      update_busy_bar(0)
      for (i in 1:100) {
        Sys.sleep(0.1)
        update_busy_bar(i)
      }
    })

    observeEvent(input$set, {
      update_busy_bar(input$set)
    })

  }

  shinyApp(ui, server)
}
```

manual-spinner *Manual busy indicator (spinner)*

Description

Declare `use_busy_spinner` in your UI and show/hide server-side with `show_spinner`/`hide_spinner`.

Usage

```
use_busy_spinner(spin = "double-bounce", color = "#112446",
  position = c("top-right", "top-left", "bottom-right", "bottom-left",
  "full-page"), margins = c(10, 10), spin_id = NULL, height = "50px",
  width = "50px")

show_spinner(spin_id = NULL,
  session = shiny::getDefaultReactiveDomain())

hide_spinner(spin_id = NULL,
  session = shiny::getDefaultReactiveDomain())
```

Arguments

`spin`	Style of the spinner, see `spin_epic` or `spin_kit` for possible choices. Note that for `spin_epic`, height and width are ignored.
`color`	Color for the spinner, in a valid CSS format.
`position`	Where to display the spinner: `'top-right'`, `'top-left'`, `'bottom-right'`, `'bottom-left'`, `'full-page'`.
`margins`	Distance from margins, a vector of length two, where first element is distance from top/bottom, second element distance from right/left.
`spin_id`	An explicit id for the spinner, useful if you want to use multiple spinners.
`height`	Height and width of the spinner, default to `'50px'` for both, must be specified.
`width`	Height and width of the spinner, default to `'50px'` for both, must be specified.
`session`	Shiny session.

Examples

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

  ui <- fluidPage(

    # Use this function somewhere in UI
    use_busy_spinner(spin = "fading-circle"),

    headerPanel('Iris k-means clustering'),

    sidebarLayout(
      sidebarPanel(
        selectInput('xcol', 'X Variable', names(iris)),
        selectInput('ycol', 'Y Variable', names(iris),
                    selected=names(iris)[[2]]),
        numericInput('clusters', 'Cluster count', 3,
                     min = 1, max = 9),
        actionButton("sleep", "Long calculation")
      ),
      mainPanel(
        plotOutput('plot1')
      )
    )
  )

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

    selectedData <- reactive({
      iris[, c(input$xcol, input$ycol)]
    })

    clusters <- reactive({
      kmeans(selectedData(), input$clusters)
    })

    output$plot1 <- renderPlot({
      palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
                "#FF7F00", "#FFFF33", "#A65628", "#F781BF",
                "#999999"))

      par(mar = c(5.1, 4.1, 0, 1))
      plot(selectedData(),
           col = clusters()$cluster,
           pch = 20, cex = 3)
      points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
    })

    observeEvent(input$sleep, {
      show_spinner()
      Sys.sleep(5)
      hide_spinner()
    })

  }

  shinyApp(ui, server)
}
```

---

modal-gif *Show a modal with a GIF*

Description

Make a pop-up window appear from the server with a GIF during long computation, remove it when finished.

Usage

```
show_modal_gif(src, text = NULL, height = "100px", width = "100px",
  modal_size = "s", session = shiny::getDefaultReactiveDomain())

remove_modal_gif(session = getDefaultReactiveDomain())
```

Arguments

`src`	Path to the GIF, an URL or a file in www/ folder.
`text`	Additional text to appear under the spinner.
`height, width`	Height and width of the spinner, default to `'50px'` for both, must be specified.
`modal_size`	One of `"s"` for small (the default) , `"m"` for medium, or `"l"` for large.
`session`	The `session` object passed to function given to `shinyServer`.

Examples

```
if (interactive()) {

  library(shiny)
  library(shinybusy)

  ui <- fluidPage(

    tags$h1("Modal with spinner"),
    actionButton("sleep1", "Launch a long calculation"),
    actionButton("sleep2", "And another one")
  )

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

    observeEvent(input$sleep1, {
      show_modal_gif(
        src = "https://jeroen.github.io/images/banana.gif"
      )
      Sys.sleep(5)
      remove_modal_gif()
    })

    observeEvent(input$sleep2, {
      show_modal_gif(
        src = "https://jeroen.github.io/images/banana.gif",
        width = "300px", height = "300px",
        modal_size = "m",
        text = "Please wait..."
      )
      Sys.sleep(5)
      remove_modal_gif()
    })

  }

  shinyApp(ui, server)

}
```

modal-progress — *Show a modal with a progress bar*

Description

Make a pop-up window appear from the server with a spinner during long computation, remove it when finished.

Usage

```
show_modal_progress_line(value = 0, text = "auto", color = "#112446",
  stroke_width = 4, easing = "linear", duration = 1000,
  trail_color = "#eee", trail_width = 1, height = "15px",
  session = shiny::getDefaultReactiveDomain())

show_modal_progress_circle(value = 0, text = "auto",
  color = "#112446", stroke_width = 4, easing = "linear",
  duration = 1000, trail_color = "#eee", trail_width = 1,
  height = "200px", session = shiny::getDefaultReactiveDomain())

remove_modal_progress(session = getDefaultReactiveDomain())

update_modal_progress(value, text = NULL,
  session = shiny::getDefaultReactiveDomain())
```

Arguments

`value`	Initial value or new value to set.
`text`	Text to display.
`color`	Main color.
`stroke_width`	Main width.
`easing`	CSS animation to use, ex.: `"linear"`, `"easeIn"`, `"easeOut"`, `"easeInOut"`.
`duration`	Animation duration (in milliseconds).
`trail_color`	Color of shape behind the main bar.
`trail_width`	Width of shape behind the main bar.
`height`	Container height.
`session`	The `session` object passed to function given to `shinyServer`.

Examples

```
if (interactive()) {

  library(shiny)
  library(shinybusy)

  ui <- fluidPage(

    tags$h1("Modal with progress bar"),
    actionButton("sleep1", "Launch a long calculation"),
    actionButton("sleep2", "And another one (different line options)"),
    tags$br(),
    actionButton("sleep3", "With a circle progress bar"),
    actionButton("sleep4", "With different circle options)")
  )

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

    observeEvent(input$sleep1, {
      show_modal_progress_line()
      for (i in 1:100) {
        update_modal_progress(
          value = i / 100
        )
        Sys.sleep(0.1)
      }
      remove_modal_progress()
    })

    observeEvent(input$sleep2, {
      show_modal_progress_line(
        color = "#DF0101",
        duration = 900,
        easing = "easeOut",
        text = "Starting computation"
      )
      Sys.sleep(0.1)
      for (i in 1:100) {
        update_modal_progress(
          value = i / 100,
          text = paste("Process", trunc(i/10), sprintf("(%02d%%)", i))
        )
        Sys.sleep(0.15)
      }
      remove_modal_progress()
    })


    observeEvent(input$sleep3, {
      show_modal_progress_circle()
      for (i in 1:100) {
        update_modal_progress(
          value = i / 100
        )
        Sys.sleep(0.1)
      }
      remove_modal_progress()
    })

    observeEvent(input$sleep4, {
      show_modal_progress_circle(
        color = "#DF0101",
        duration = 900,
        easing = "easeOut",
        text = "Starting computation",
        height = "300px"
      )
      Sys.sleep(0.1)
      for (i in 1:100) {
        update_modal_progress(
          value = i / 100,
          text = paste("Process", trunc(i/10), sprintf("(%02d%%)", i))
        )
        Sys.sleep(0.15)
      }
      remove_modal_progress()
    })

  }

  shinyApp(ui, server)

}
```

`modal-spinner` *Show a modal with a spinner*

Description

Make a pop-up window appear from the server with a spinner during long computation, remove it when finished.

Usage

```
show_modal_spinner(spin = "double-bounce", color = "#112446",
  text = NULL, session = shiny::getDefaultReactiveDomain())

remove_modal_spinner(session = getDefaultReactiveDomain())
```

Arguments

`spin`	Style of the spinner, see spin_epic or spin_kit for possible choices.
`color`	Color for the spinner, in a valid CSS format.
`text`	Additional text to appear under the spinner.
`session`	The `session` object passed to function given to `shinyServer`.

Examples

```
if (interactive()) {

  library(shiny)
  library(shinybusy)

  ui <- fluidPage(

    tags$h1("Modal with spinner"),
    actionButton("sleep1", "Launch a long calculation"),
    actionButton("sleep2", "And another one")
  )

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

    observeEvent(input$sleep1, {
      show_modal_spinner()
      Sys.sleep(5)
      remove_modal_spinner()
    })

    observeEvent(input$sleep2, {
      show_modal_spinner(
        spin = "cube-grid",
        color = "firebrick",
        text = "Please wait..."
      )
      Sys.sleep(5)
      remove_modal_spinner()
    })
  }

  shinyApp(ui, server)

}
```

progress   *Create progress indicator*

Description

Bar, circle or semicircle to show progress. Can be used outside Shiny. In Shiny you can set progress value server-side.

Usage

```
progress_line(value = 0, color = "#112446", stroke_width = 4,
  easing = "linear", duration = 1000, trail_color = "#eee",
  trail_width = 1, text = "auto", text_color = "#000",
  width = "100%", height = "15px", shiny_id = NULL)

progress_circle(value = 0, color = "#112446", stroke_width = 4,
  easing = "easeInOut", duration = 1400, trail_color = "#eee",
  trail_width = 1, text = "auto", text_color = "#000",
  width = "200px", height = "200px", shiny_id = NULL)

progress_semicircle(value = 0, color = "#112446", stroke_width = 4,
  easing = "easeInOut", duration = 1400, trail_color = "#eee",
  trail_width = 1, text = "auto", text_color = "#000",
  width = "200px", height = "100px", shiny_id = NULL)

update_progress(shiny_id, value, text = NULL,
  session = shiny::getDefaultReactiveDomain())
```

Arguments

`value`	Initial value or new value to set.
`color`	Main color.
`stroke_width`	Main width.
`easing`	CSS animation to use, ex.: `"linear"`, `"easeIn"`, `"easeOut"`, `"easeInOut"`.
`duration`	Animation duration (in milliseconds).
`trail_color`	Color of shape behind the main bar.
`trail_width`	Width of shape behind the main bar.
`text`	Text to display.
`text_color`	Text color.

`width`	Container width.
`height`	Container height.
`shiny_id`	Id to use in Shiny application.
`session`	Shiny session.

Value

an `htmlwidget` object.

`spin_epic` *Epic spinners*

Description

Via https://epic-spinners.epicmax.co/.

Usage

```
spin_epic(spin = c("flower", "pixel", "hollow-dots",
  "intersecting-circles", "orbit", "radar", "scaling-squares",
  "half-circle", "trinity-rings", "fulfilling-square",
  "circles-to-rhombuses", "semipolar", "self-building-square",
  "swapping-squares", "fulfilling-bouncing-circle", "fingerprint",
  "spring", "atom", "looping-rhombuses", "breeding-rhombus"),
  color = "#112446")
```

Arguments

`spin`	Name of the spinner.
`color`	Color of the spinner.

Value

an HTML tag.

Examples

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

  ui <- fluidPage(
    tags$h2("Epic spinner demo"),
    lapply(
      X = c(
        "flower", "pixel", "hollow-dots",
        "intersecting-circles", "orbit", "radar",
        "scaling-squares", "half-circle",
        "fulfilling-square", "circles-to-rhombuses"
      ),
      FUN = function(x) {
        tags$div(
          style = "display: table-cell; width: 150px; height: 100px; margin: 10px;",
          tags$b(x),
          spin_epic(x, color = "#08298A")
        )
      }
    ),
    tags$hr(),
    lapply(
      X = c(
        "semipolar", "self-building-square", "swapping-squares",
        "fulfilling-bouncing-circle", "fingerprint", "spring",
        "atom", "looping-rhombuses", "breeding-rhombus", "trinity-rings"
      ),
      FUN = function(x) {
        tags$div(
          style = "display: table-cell; width: 150px; height: 100px; margin: 10px;",
          tags$b(x),
          spin_epic(x, color = "#08298A")
        )
      }
    )
  )

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

  }

  shinyApp(ui, server)
}
```

spin_kit *SpinKit spinners*

Description

Via https://tobiasahlin.com/spinkit/.

Usage

```
spin_kit(spin = c("double-bounce", "circle", "bounce", "folding-cube",
  "rotating-plane", "cube-grid", "fading-circle", "dots", "cube"),
  color = "#112446", style = NULL)
```

Arguments

`spin`	Name of the spinner.
`color`	Color of the spinner.
`style`	If not `NULL`, add a div container with specified style.

Value

an HTML tag.

Examples

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

  ui <- fluidPage(
    tags$h2("SpinKit demo"),
    fluidRow(lapply(
      X = c(
        "circle", "bounce", "folding-cube", "rotating-plane", "cube-grid",
        "fading-circle", "double-bounce", "dots", "cube"
      ),
      FUN = function(x) {
        column(
          width = 2,
          tags$b(x),
          tags$div(
            style = "width: 60px; height: 60px; position: relative;",
            spin_kit(spin = x)
          )
        )
      }
    ))
  )

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

  }

  shinyApp(ui, server)
}
```

Index

add_busy_bar, 2
add_busy_gif, 3
add_busy_spinner, 4

hide_spinner *(manual-spinner)*, 9

logo_silex, 6

manual-gif, 6
manual-progressbar, 8
manual-spinner, 9
modal-gif, 11
modal-progress, 12
modal-spinner, 14

play_gif *(manual-gif)*, 6
progress, 16
progress_circle *(progress)*, 16
progress_line *(progress)*, 16
progress_semicircle *(progress)*, 16

remove_modal_gif *(modal-gif)*, 11
remove_modal_progress *(modal-progress)*, 12
remove_modal_spinner *(modal-spinner)*, 14

show_modal_gif *(modal-gif)*, 11
show_modal_progress_circle *(modal-progress)*, 12
show_modal_progress_line *(modal-progress)*, 12
show_modal_spinner *(modal-spinner)*, 14
show_spinner *(manual-spinner)*, 9
spin_epic, *4*, *9*, *15*, 17
spin_kit, *4*, *9*, *15*, 18
stop_gif *(manual-gif)*, 6

update_busy_bar *(manual-progressbar)*, 8
update_modal_progress *(modal-progress)*, 12
update_progress *(progress)*, 16
use_busy_bar *(manual-progressbar)*, 8
use_busy_gif *(manual-gif)*, 6
use_busy_spinner *(manual-spinner)*, 9