Package 'shinybusy'

August 19, 2019

Title Busy Indicator for 'Shiny' Applications

Version 0.1.3

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

RoxygenNote 6.1.1

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

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

Suggests testthat

NeedsCompilation no

Author Fanny Meyer [aut],
Victor Perrier [aut, cre],
Silex Technologies [fnd] (https://www.silex-ip.com),
Tobias Ahlin [cph] (spin.css),
Chris Antonellis [cph] (freezeframe.js),
Jacobo Tabernero [cph] (nanobar.js)

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

Repository CRAN

Date/Publication 2019-08-19 17:40:02 UTC

R topics documented:

`add_busy_bar` *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)
```

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.

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

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

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

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

  }

  shinyApp(ui, server)
}
```

add_busy_gif *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 *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, choice between : `circle`, `bounce`, `folding-cube`, `rotating-plane`, `cube-grid`, `fading-circle`, `double-bounce`, `dots`, `cube`.
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 = "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, {
      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-progressbar` *Manual 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)

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

Arguments

`color`	Progress bar color.
`centered`	Center the progress bar or not.
`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(),
    actionButton(inputId = "go", label = "Go")
  )

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

    observeEvent(input$go, {
      update_busy_bar(10)
      Sys.sleep(1)
      update_busy_bar(20)
      Sys.sleep(1)
      update_busy_bar(40)
      Sys.sleep(1)
```

```
      update_busy_bar(60)
      Sys.sleep(1)
      update_busy_bar(80)
      Sys.sleep(1)
      update_busy_bar(100)
    })

  }

  shinyApp(ui, server)
}
```

`manual-spinner` *Manual 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, choice between : `circle`, `bounce`, `folding-cube`, `rotating-plane`, `cube-grid`, `fading-circle`, `double-bounce`, `dots`, `cube`.
`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, width`	Height and width ot 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)
}
```

Index