Package ‘shinyglide’

February 15, 2023

Type Package

Title Glide Component for Shiny Applications

Version 0.1.4

Date 2023-02-15

Maintainer Julien Barnier <julien.barnier@cnrs.fr>

Description Insert Glide JavaScript component into Shiny applications for carousel or assistant-like user interfaces.

License GPL (>= 3)

VignetteBuilder knitr

URL https://juba.github.io/shinyglide/, https://github.com/juba/shinyglide

BugReports https://github.com/juba/shinyglide/issues

Encoding UTF-8

Imports shiny (>= 1.2.0), htmltools

Suggests knitr, rmarkdown

RoxygenNote 7.1.2

NeedsCompilation no

Author Julien Barnier [aut, cre]

Repository CRAN

Date/Publication 2023-02-15 10:40:02 UTC

R topics documented:

firstButton *Create a glide control only shown on first or last screen*

Description

Create a glide control only shown on first or last screen

Usage

```
firstButton(class = c("btn", "btn-default"), ...)

lastButton(class = c("btn", "btn-success"), ...)
```

Arguments

`class`	CSS classes of the control. The needed class is automatically added.
`...`	content of the control

Details

These controls generate an <a> tag, so you can use `href` attributes.

`firstButton` is only shown on the first screen of the app, and `finalButton` only on the last screen.

Examples

```
firstButton("Go to website", href = "https://example.com", class = "btn btn-primary")
```

glide *Glide component creation*

Description

Insert a glide component in the current shiny app UI

Usage

```
glide(
  ...,
  id = NULL,
  next_label = paste("Next", shiny::icon("chevron-right", lib = "glyphicon")),
 previous_label = paste(shiny::icon("chevron-left", lib = "glyphicon"), "Back"),
  loading_label = span(span(class = "shinyglide-spinner"), span("Loading")),
  loading_class = "loading",
  disable_type = c("disable", "hide"),
  height = "100%",
  keyboard = TRUE,
  swipe = TRUE,
  custom_controls = NULL,
  controls_position = c("bottom", "top")
)
```

Arguments

`...`	content of the glide.
`id`	optional HTML id of the glide root element.
`next_label`	label to be used in the "next" control.
`previous_label`	label to be used in the "back" control.
`loading_label`	label to be used in the "next" control when the next screen is still loading.
`loading_class`	class to add to the "next" control when the next screen is still loading.
`disable_type`	either to "disable" or "hide" the next or back control when it is disabled by a condition.
`height`	height of the glide (something like "400px" or "100%").
`keyboard`	set this to FALSE to disable keyboard navigation.
`swipe`	set this to FALSE to disable swipe navigation.
`custom_controls`	custom HTML or shiny tags to be used for the controls. If ‘NULL“, use the default ones.
`controls_position`	either to place the default or custom controls on "top" or "bottom" of the glide.

See Also

screen nextButton prevButton firstButton lastButton

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

ui <- fixedPage(
 h3("Simple shinyglide app"),
 glide(
    screen(
      p("First screen.")
    ),
    screen(
      p("Second screen.")
    )
  )
)
server <- function(input, output, session) {
}

shinyApp(ui, server)

}
```

---

glideControls *Default controls layout*

Description

Creates an horizontal layout with both "previous" and "next" contents side by side.

Usage

```
glideControls(previous_content = prevButton(), next_content = nextButton())
```

Arguments

`previous_content`	Content of the "previous" (left) zone.
`next_content`	Content of the "next" (right) zone.

Examples

```
glideControls(
  prevButton("Back"),
  list(
    lastButton(href = "https://example.com", "Go to website"),
    nextButton("Next")
  )
)
```

---

nextButton *Code for the default controls*

Description

This generates the code of the default controls, and can be used in custom controls.

Usage

```
nextButton(class = c("btn", "btn-primary"))

prevButton(class = c("btn", "btn-default"))
```

Arguments

`class`	control CSS classes. The needed class is automatically added.

Details

`prevButton` is hidden on the first screen, while `nextButton` is hidden on the last one. The buttons labels are set with the `next_label` and `previous_label` arguments of `glide()`.

See Also

glide

`screen` *Screen creation*

Description

Insert a new screen into a glide component.

Usage

```
screen(
  ...,
  next_label = NULL,
  previous_label = NULL,
  next_condition = NULL,
  previous_condition = NULL,
  class = NULL
)
```

Arguments

`...`	content of the screen.
`next_label`	specific label of the "next" control for this screen. If `NULL`, use the default one for the current glide.
`previous_label`	specific label of the "back" control for this screen. If `NULL`, use the default one for the current glide.
`next_condition`	condition for the "next" control to be enabled. Same syntax as `shiny::conditionalPanel`.
`previous_condition`	condition for the "back" control to be enabled. Same syntax as `shiny::conditionalPanel`.
`class`	screen CSS classes. `glide__slide` is automatically added.

Details

This function inserts a new "screen" into an existing `glide` component. It can only be used inside a `glide()` call, in a shiny app UI.

See Also

glide

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

ui <- fixedPage(
 h3("Simple shinyglide app"),
 glide(
    screen(
      next_label = "Go next",
      next_condition = "input.x > 0",
      p("First screen."),
      numericInput("x", "x", value = 0)
    ),
    screen(
      p("Final screen."),
    )
  )
)

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

shinyApp(ui, server)

}
```

`screenOutput` *Create a screen output element*

Description

Insert a screen output element in a shiny app UI. This must be used with a `renderUI` reactive expression in the app server.

Usage

```
screenOutput(
  outputId,
  next_label = NULL,
  prev_label = NULL,
  next_condition = NULL,
  prev_condition = NULL,
  class = NULL,
  ...
)
```

Arguments

`outputId`	output variable to read the value from
`next_label`	specific label of the "next" control for this screen. If `NULL`, use the default one for the current glide.
`prev_label`	specific label of the "back" control for this screen. If `NULL`, use the default one for the current glide.
`next_condition`	condition for the "next" control to be enabled. Same syntax as `shiny::conditionalPanel`.
`prev_condition`	condition for the "back" control to be enabled. Same syntax as `shiny::conditionalPanel`.
`class`	screen CSS classes. `glide__slide` is automatically added.
`...`	other arguments to pass to the container tag function.

Details

Important : for this to work, you have to add a `outputOptions(output, id, suspendWhenHidden = FALSE)` in your app server. See example.

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

ui <- fixedPage(
 h3("Simple shinyglide app"),
 glide(
    screen(
      p("First screen."),
    ),
    screenOutput("screen"),
    screen(
      p("Final screen."),
    )
  )
)

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

  output$screen <- renderUI({
    p("Second screen.")
  })
  outputOptions(output, "screen", suspendWhenHidden = FALSE)

}

shinyApp(ui, server)

}
```

Index