

Package: hover (via r-universe)

August 25, 2024

Title CSS Animations for 'shiny' Button Elements
Version 0.1.1
Description A wrapper around a CSS library called 'Hover.css',
intended for use in 'shiny' applications.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Roxygen list(markdown = TRUE)
RoxygenNote 7.1.1
Imports shiny, htmltools
URL <https://github.com/r4fun/hover>
BugReports <https://github.com/r4fun/hover/issues>
Suggests testthat
Depends R (>= 2.10)
Repository <https://r4fun.r-universe.dev>
RemoteUrl <https://github.com/r4fun/hover>
RemoteRef HEAD
RemoteSha bd5bfa085bdd12a95a699a3bfc71116ff0e28af7

Contents

animations	2
hover_action_button	2
hover_download_button	3
hover_reload_button	5
use_hover	6

Index	8
--------------	----------

animations	<i>List of all available animations</i>
------------	---

Description

A list containing all available animations.

Usage

```
animations
```

Format

An object of class `list` of length 7.

Details

Source: <https://github.com/IanLunn/Hover>

hover_action_button	<i>Action button with button and icon animations</i>
---------------------	--

Description

Animate an `actionButton` and its icon using [Hover.css](#)

Usage

```
hover_action_button(  
  inputId,  
  label,  
  icon = NULL,  
  button_animation = NULL,  
  icon_animation = NULL,  
  width = NULL,  
  ...  
)
```

Arguments

<code>inputId</code>	The input slot that will be used to access the value.
<code>label</code>	The contents of the button or link—usually a text label, but you could also use any other HTML, like an image.
<code>icon</code>	An optional <code>icon()</code> to appear on the button.

button_animation	The name of the button animation.
icon_animation	The name of the icon animation.
width	The width of the input, e.g. '400px', or '100%'; see validateCssUnit().
...	Named attributes to be applied to the button or link.

Source

<https://github.com/IanLunn/Hover>

Examples

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

  ui <- fluidPage(
    use_hover(),
    hover_action_button(
      inputId = "btn",
      label = "hello hover!",
      icon = icon("refresh"),
      button_animation = "rotate",
      icon_animation = "spin"
    )
  )

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

  }

  shinyApp(ui, server)
}
```

hover_download_button *Download button with button and icon animations*

Description

Animate a downloadButton and it's icon using [Hover.css](#)

Usage

```
hover_download_button(
  outputId,
  label = "Download",
  button_animation = NULL,
  icon_animation = NULL,
```

```

    class = NULL,
    ...
  )

```

Arguments

<code>outputId</code>	The name of the output slot that the <code>downloadHandler</code> is assigned to.
<code>label</code>	The label that should appear on the button.
<code>button_animation</code>	The name of the button animation.
<code>icon_animation</code>	The name of the icon animation.
<code>class</code>	Additional CSS classes to apply to the tag, if any.
<code>...</code>	Other arguments to pass to the container tag function.

Source

<https://github.com/IanLunn/Hover>

Examples

```

if (interactive()) {
  library(shiny)
  library(hover)

  ui <- fluidPage(
    use_hover(),
    hover_download_button(
      outputId = "downloadData",
      label = "Download",
      button_animation = "rotate",
      icon_animation = "spin"
    )
  )

  server <- function(input, output) {
    # Our dataset
    data <- mtcars

    output$downloadData <- downloadHandler(
      filename = function() {
        paste("data-", Sys.Date(), ".csv", sep="")
      },
      content = function(file) {
        write.csv(data, file)
      }
    )
  }

  shinyApp(ui, server)
}

```

hover_reload_button *Reload button with button and icon animations*

Description

Animate a reload button and its icon using [Hover.css](#). Note that a reload button is just a `shiny::actionButton` with `onClick` behavior to reload or refresh a web browser.

Usage

```
hover_reload_button(  
  inputId,  
  label,  
  icon = NULL,  
  button_animation = NULL,  
  icon_animation = NULL,  
  width = NULL,  
  ...  
)
```

Arguments

<code>inputId</code>	The input slot that will be used to access the value.
<code>label</code>	The contents of the button or link—usually a text label, but you could also use any other HTML, like an image.
<code>icon</code>	An optional <code>icon()</code> to appear on the button.
<code>button_animation</code>	The name of the button animation.
<code>icon_animation</code>	The name of the icon animation.
<code>width</code>	The width of the input, e.g. <code>'400px'</code> , or <code>'100%'</code> ; see <code>validateCssUnit()</code> .
<code>...</code>	Named attributes to be applied to the button or link.

Source

<https://github.com/IanLunn/Hover>

Examples

```
if (interactive()) {  
  library(shiny)  
  library(hover)  
  
  ui <- fluidPage(  
    use_hover(),  
    hover_reload_button(  
      inputId = "btn",
```

```
      label = "hello hover!",
      icon = icon("refresh"),
      button_animation = "rotate",
      icon_animation = "spin"
    )
  )

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

  }

  shinyApp(ui, server)
}
```

use_hover

Use the hover package

Description

Enables hover by including the CSS file necessary for the animations.

Usage

```
use_hover(popback = FALSE)
```

Arguments

popback If true, buttons 'pop back', contrary to default shiny behavior.

Details

By default, shiny buttons don't 'pop back'. This is for accessibility reasons. For more information see here: <https://github.com/rstudio/shiny/issues/2500>.

Examples

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

  ui <- fluidPage(
    use_hover(),
    hover_action_button(
      inputId = "btn",
      label = "hello hover!",
      icon = icon("refresh"),
      button_animation = "rotate",
      icon_animation = "spin"
    )
  )
}
```

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

Index

* **datasets**

animations, [2](#)

animations, [2](#)

hover_action_button, [2](#)

hover_download_button, [3](#)

hover_reload_button, [5](#)

use_hover, [6](#)