

# Package ‘hover’

March 20, 2021

**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

**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)

**NeedsCompilation** no

**Author** Tyler Littlefield [aut, cre] (Creator of Shiny Wrapper),  
Ian Lunn [ctb, cph] (Author of included CSS code),  
Danube Huynhle [ctb]

**Maintainer** Tyler Littlefield <[tylerlittlefield@hey.com](mailto:tylerlittlefield@hey.com)>

**Repository** CRAN

**Date/Publication** 2021-03-20 17:20:02 UTC

## R topics documented:

animations . . . . .	2
hover_action_button . . . . .	2
hover_download_button . . . . .	3
hover_reload_button . . . . .	5
use_hover . . . . .	6
<b>Index</b>	<b>8</b>

---

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 it's icon using [Hover.css](#)

**Usage**

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

**Arguments**

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

outputId	The name of the output slot that the downloadHandler is assigned to.
label	The label that should appear on the button.
button_animation	The name of the button animation.
icon_animation	The name of the icon animation.
class	Additional CSS classes to apply to the tag, if any.
...	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](#)