Package 'shinyLottie'

May 9, 2026

Type Package

Title Seamlessly Integrate 'Lottie' Animations into 'shiny' Applications

Version 1.0.0

Description Easily integrate and control 'Lottie' animations within 'shiny' applications', without the need for idiosyncratic expression or use of 'JavaScript'. This includes utilities for generating animation instances, controlling playback, manipulating animation properties, and more. For more information on 'Lottie', see: <https://airbnb.io/lottie/#/>. Additionally, see the official 'Lottie' GitHub repository at <https://github.com/airbnb/lottie>.

Imports shiny, jsonlite, glue, htmltools

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://camhowitt.github.io/shinyLottie/

NeedsCompilation no

Author Cameron Howitt [aut, cre, cph]

Maintainer Cameron Howitt <shinylottie@gmail.com>

Repository CRAN

Date/Publication 2024-06-21 10:30:02 UTC

Contents

`include_lottie` *Include 'Lottie' Functionality within 'shiny'*

Description

Responsible for retrieving the 'Lottie' library and initialising the necessary 'JavaScript' library. As such, this function **must** be included within the UI object of a `shinyApp` in order to enable 'shinyLottie' functionality.

Usage

```
include_lottie(version = "5.12.2")
```

Arguments

`version`	A character string specifying the version of the 'Lottie' library to source via CDN.

Value

A list of HTML tags to be included within the head element of a 'shiny' application.

Note

Calling this function initialises a global object "`window.lottieInstances`" once the DOM content is fully loaded. This is used to store the 'Lottie' animations that are created using `lottie_animation`.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
```

```
    name = "my_animation"
  )
)

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

shinyApp(ui, server)
```

`lottie_addEventListener` *Add Event Listener to 'Lottie' Animation*

Description

Adds an event listener to a 'Lottie' animation within a 'shiny' application. It is also possible to apply multiple event listeners to a single animation.

Usage

```
lottie_addEventListener(
  animation,
  event,
  target,
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`animation`	A 'Lottie' animation object created by the `lottie_animation` function or its name.
`event`	The event to listen for (e.g. `"mouseenter"`, `"mouseleave"` etc.).
`target`	The target for the event listener, either `"animation"` or `"container"`.
`...`	Additional optional event listener properties, including: `state` A character string corresponding to an animation state (either `"play"`, `"pause"`, or `"stop"`). `loop` Logical value indicating whether the animation should loop. `speed` A numeric specifying the desired animation speed. `direction` Either `1` for forward playback or `-1` for reverse playback. `setSubFrame` A logical value specifying whether a 'Lottie' animation should loop (`TRUE`) or not (`FALSE`). `playSegments` A numeric vector or list of numeric vectors indicating the segment(s) to be played.

`forceFlag` Logical value indicating whether to force the animation to play the specified segments immediately (`TRUE`) or wait until the current animation completes (`FALSE`).

`custom_js` Custom 'JavaScript' to execute when the event is triggered.

`functionName` Optional name for the event handler function (can be useful when referencing the event listener, such as with `lottie_removeEventListener`).

`session` The 'shiny' session object. Defaults to the current reactive domain.

Details

This function has several usage options:

- Supplying an animation object created by the `lottie_animation` function, and placing the resultant list object in the 'shiny' UI.
- Outside of a reactive context, supplying the name of the animation and placing the resultant `script` object in the 'shiny' UI.
- Within a reactive context, supplying the name of the animation.

When run within a reactive context, sends a custom session message `"lottie_js_runJS"` containing the function arguments.

Target Options

- `"animation"`: Attaches the event listener directly to the 'Lottie' animation instance. This is necessary when using a Lottie-specific event (e.g. "onComplete"). See https://airbnb.io/lottie/#/web for further details.
- `"container"`: Attaches the event listener to the container div of the 'Lottie' animation. This should be used when using a generic HTML event, such as "mouseenter" or "mouseleave".

Value

If used within a reactive context, the function will execute the necessary 'JavaScript'. Otherwise, it will return a `script` tag containing the 'JavaScript'.

Note

Using the `custom_js` argument, it is possible to assign 'shiny' input values when an event is triggered, see `lottie_removeEventListener` for an example.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  # Create an 'animation' event listener that prints a message to the
  # browser console after each loop
  lottie_animation(
    path = "shinyLottie/example.json",
```

```
    name = "my_animation"
  ) |>
  lottie_addEventListener(
    event = "loopComplete",
    target = "animation",
    custom_js = "console.log('Animation Complete!');"
  ),
  # Create a 'container' event listener that plays an animation when
  # hovering over the button, and another that pauses the animation
  # when hovering stops
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "button",
    width = "200px",
    height = "100px",
    loop = TRUE,
    autoplay = FALSE,
  ) |> lottie_button(inputId = "lottieButton", label = "Lottie",
                     height = "200px", width = "200px") |>
    lottie_addEventListener("mouseenter", "container", state = "play") |>
    lottie_addEventListener("mouseleave", "container", state = "pause")
)

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

shinyApp(ui, server)
```

`lottie_animation` *Generate 'Lottie' Animation for a 'shiny' application*

Description

Generates a 'Lottie' animation for use within a 'shiny' application.

Usage

```
lottie_animation(
  path,
  name,
  loop = TRUE,
  autoplay = TRUE,
  renderer = "svg",
  width = "400px",
  height = "400px",
  ...,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

path	Either a URL or local file path (see Note).
name	A character string specifying the name to give to the animation.
loop	Logical indicating whether the animation should loop.
autoplay	Logical indicating whether the animation should autoplay.
renderer	The renderer to use for the animation, either "svg", "canvas", or "html".
width	The width of the animation container. This is validated using validateCssUnit.
height	The height of the animation container. This is validated using validateCssUnit.
...	Additional animation options, including: speed A numeric specifying the desired animation speed. direction Either 1 for forward playback or -1 for reverse playback. setSubFrame A logical value specifying whether a 'Lottie' animation should loop (TRUE) or not (FALSE). playSegments A numeric vector or list of numeric vectors indicating the segment(s) to be played. forceFlag Logical value indicating whether to force the animation to play the specified segments immediately (TRUE) or wait until the current animation completes (FALSE).
session	The 'shiny' session object. Defaults to the current reactive domain.

Value

A list containing the following elements:

div An HTML div element serving as the 'Lottie' animation container.

script A script tag containing the 'JavaScript' to initialise the 'Lottie' animation.

Note

When using a local file path, you may need to use addResourcePath.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  )
)

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

shinyApp(ui, server)
```

lottie_animation_methods *'Lottie' Animation Methods*

Description

These functions provide methods for modifying the playback options of existing 'Lottie' animations.

Details

- lottie_setSpeed: Set the speed of the animation.
- lottie_setDirection: Set the direction of the animation.
- lottie_setLoop: Set whether the animation should loop.
- lottie_goToAndStop: Move to a specific frame or time and stop.
- lottie_goToAndPlay: Move to a specific frame or time and play.
- lottie_playSegments: Play specific segments of the animation.
- lottie_setSubframe: Set whether to use subframes when rendering the animation.
- lottie_destroy: Destroy the specified animation instance.

lottie_button *Convert a 'Lottie' Animation to a Button*

Description

Wraps a 'Lottie' animation within a button element for use in 'shiny' applications.

Usage

```
lottie_button(
  animation,
  inputId,
  label = NULL,
  width = NULL,
  height = NULL,
  ...
)
```

Arguments

animation	A 'Lottie' animation created by lottie_animation.
inputId	The 'shiny' input slot that will be used to access the value.
label	Optional text label to display below the animation inside the button.
width	Width of the button. This is validated using validateCssUnit.
height	Height of the button. This is validated using validateCssUnit.
...	Additional named attributes to pass to the button element. Same behaviour as actionButton.

Value

An HTML button element enclosing the animation input object.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation",
    height = "100px",
    width = "100px"
  ) |> lottie_button(inputId = "my_button")
)

server <- function(input, output, session) {
  observeEvent(input$my_button, {
    print("Button pressed")
  })
}

shinyApp(ui, server)
```

---

lottie_callMethod *Call a 'Lottie' Method*

Description

Call a method for a specific 'Lottie' animation or all 'Lottie' animations.

Usage

```
lottie_callMethod(
  name = "all",
  method,
  argument = "",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

name	A character string specifying the name of the 'Lottie' animation to query. The default of "all" will retrieve the specified property from all 'Lottie' animations within the 'shiny' application.
method	A character string specifying the name of the method to call.
argument	A character string specifying any optional arguments to pass to the method.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message "lottie_js_callMethod" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("callMethod", "Call Method (Reverse Direction)")
)

server <- function(input, output, session) {
  observeEvent(input$callMethod, {
    lottie_callMethod(name = "my_animation", method = "setDirection", argument = "-1")
  })
}
shinyApp(ui, server)
```

lottie_destroy *Destroy a 'Lottie' Animation*

Description

Permanently destroy a specific 'Lottie' animation or all 'Lottie' animations.

Usage

```
lottie_destroy(name = "all", session = shiny::getDefaultReactiveDomain())
```

Arguments

name	A character string specifying the name of the 'Lottie' animation to destroy. The default of "all" will destroy all animations within the 'shiny' application.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message "lottie_js_destroy" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

See Also

lottie_animation_methods for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("destroy", "Destroy Animation")
)

server <- function(input, output, session) {
  observeEvent(input$destroy, {
    lottie_destroy("my_animation")
  })
}
```

```
shinyApp(ui, server)
```

`lottie_getProperty` *Get a Property of a 'Lottie' Animation*

Description

Get a property from a specific 'Lottie' animation or all 'Lottie' animations.

Usage

```
lottie_getProperty(
  property,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`property`	A character string specifying the name of the property to retrieve.
`name`	A character string specifying the name of the 'Lottie' animation to query. The default of `"all"` will retrieve the specified property from all animations within the 'shiny' application.
`session`	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message `"lottie_js_getProperty"` containing the function arguments.

Value

The return value(s) can be retrieved from within a reactive context by accessing the input object of the 'shiny' session, where the value has been assigned as the property name. For example, if accessing the `playCount` property, the return value can be retrieved via `input$playCount`.

If `name = "all"` has been specified, then the return object will be a list, with named elements corresponding to the animation names.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
```

```
  ),
  actionButton("getProperty", "Update Play Count"),
  textOutput("playCountOutput")
)

server <- function(input, output, session) {
  observeEvent(input$getProperty, {
    lottie_getProperty(name = "my_animation", property = "playCount")
  })

  observe({
    req(input$playCount)
    output$playCountOutput <- renderText({
      paste("Play Count:", input$playCount)
    })
  })
}

 shinyApp(ui, server)
```

`lottie_navigate_frame` *Navigate to a Specific Animation Frame*

Description

Navigate to a specific frame or time and either stop or play the animation.

Usage

```
lottie_goToAndStop(
  value,
  isFrame = TRUE,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)

lottie_goToAndPlay(
  value,
  isFrame = TRUE,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`value`	A numeric value specifying the frame or time to go to.
`isFrame`	A logical value indicating whether `value` is a frame number (`TRUE`) or time (`FALSE`).

name	A character string specifying the name of the 'Lottie' animation to control. The default of "all" will control all animations within the 'shiny' application.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

lottie_goToAndStop moves the animation to a specific frame or time, then stops it. Sends a custom session message "lottie_js_goToAndStop" containing the function arguments.

lottie_goToAndPlay moves the animation to a specific frame or time, then continues playback. Sends a custom session message "lottie_js_goToAndPlay" containing the function arguments.

Value

These functions are called for a side effect, and so there is no return value.

See Also

lottie_animation_methods for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("goToAndStop", "Go To Frame 10 And Stop"),
  actionButton("goToAndPlay", "Go To Frame 10 And Play")
)

server <- function(input, output, session) {
  observeEvent(input$goToAndStop, {
    lottie_goToAndStop(value = 10, isFrame = TRUE, name = "my_animation")
  })

  observeEvent(input$goToAndPlay, {
    lottie_goToAndPlay(value = 10, isFrame = TRUE, name = "my_animation")
  })
}

shinyApp(ui, server)
```

lottie_playback_controls

Control Playback of 'Lottie' Animations

Description

Control the playback of 'Lottie' animations within a 'shiny' application.

Usage

```
lottie_play(name = "all", session = shiny::getDefaultReactiveDomain())

lottie_pause(name = "all", session = shiny::getDefaultReactiveDomain())

lottie_stop(name = "all", session = shiny::getDefaultReactiveDomain())
```

Arguments

name	A character string specifying the name of the 'Lottie' animation to control. The default of "all" will control all animations within the 'shiny' application.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

Each function sends a corresponding custom session message containing the function arguments:

- Play: "lottie_js_play"
- Pause: "lottie_js_pause"
- Stop: "lottie_js_stop"

Value

These functions are called for a side effect, and so there is no return value.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("play", "Play Animation"),
  actionButton("pause", "Pause Animation"),
  actionButton("stop", "Stop Animation")
```

```
)

server <- function(input, output, session) {
  observeEvent(input$play, {
    lottie_play(name = "my_animation")
  })

  observeEvent(input$pause, {
    lottie_pause(name = "my_animation")
  })

  observeEvent(input$stop, {
    lottie_stop(name = "my_animation")
  })
}

shinyApp(ui, server)
```

`lottie_playSegments` *Play Specific Segments of a 'Lottie' Animation*

Description

Play specific segments of a 'Lottie' animation.

Usage

```
lottie_playSegments(
  segments,
  forceFlag = TRUE,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`segments`	A numeric vector or list of numeric vectors indicating the segment(s) to be played.
`forceFlag`	Logical value indicating whether to force the animation to play the specified segments immediately (`TRUE`) or wait until the current animation completes (`FALSE`).
`name`	A character string specifying the name of the 'Lottie' animation to control. The default of `"all"` will control all animations within the 'shiny' application.
`session`	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message `"lottie_js_playSegments"` containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

Note

To play a single segment, `segments` should be a numeric vector of length 2 that represents the start and end frames. To play multiple segments, provide a list containing multiple numeric vectors of length 2. Note that if the animation is set to be looped, only the final segment will be repeated.

See Also

`lottie_animation_methods` for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation",
    loop = FALSE,
    speed = 0.5 # Slowed to make effects clearer
  ),
  actionButton("playSegments1", "Play Frames 1 - 10"),
  # Will not work if animation has less than 40 frames
  actionButton("playSegments2", "Play Frames 1 - 10 and 30 - 40")
)

server <- function(input, output, session) {
  observeEvent(input$playSegments1, {
    lottie_playSegments(segments = c(1, 10), forceFlag = TRUE,
      name = "my_animation")
  })

  observeEvent(input$playSegments2, {
    lottie_playSegments(segments = list(c(1, 10), c(30, 40)),
      forceFlag = TRUE, name = "my_animation")
  })
}

shinyApp(ui, server)
```

lottie_removeEventListener *Remove Event Listener from 'Lottie' Animation*

Description

Removes an event listener from a 'Lottie' animation within a 'shiny' application.

Usage

```
lottie_removeEventListener(
  name,
  event,
  target,
  functionName = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

name	A character string specifying the name of the 'Lottie' animation.
event	The event to listen for (e.g. "mouseenter", "mouseleave" etc.).
target	The target for the event listener, either "animation" or "container".
functionName	Optional name of the event handler function to remove. Should only be used if a functionName was specified when calling lottie_addEventListener.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

When run within a reactive context, sends a custom session message "lottie_js_runJS" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  # Create an 'animation' event that updates the 'playCount' input value
  # value after each loop
  lottie_animation(
    path = "shinyLottie/example.json",
```

```
    name = "my_animation"
  ) |>
    lottie_addEventListener(
      event = "loopComplete",
      target = "animation",
      custom_js = "Shiny.setInputValue('playCount',
      lottieInstances.my_animation.playCount, {priority: 'event'});"
    ),
  actionButton("removeEventListener", "Remove Event Listener")
)

server <- function(input, output, session) {
  # Notifications demonstrate that eventListener is active
  observeEvent(input$playCount, {
    showNotification(paste("Animation played", input$playCount, "times"), duration = 1)
  })

  # Removing the event listener ceases the notifications
  observeEvent(input$removeEventListener, {
    lottie_removeEventListener(name = "my_animation", event = "loopComplete",
                               target = "animation")
  })
}

shinyApp(ui, server)
```

`lottie_setDirection` *Adjust 'Lottie' Animation Direction*

Description

Adjust the playback direction of an existing 'Lottie' animation.

Usage

```
lottie_setDirection(
  direction = 1,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`direction`	Either `1` for forward playback or `-1` for reverse playback.
`name`	A character string specifying the name of the 'Lottie' animation to control. The default of `"all"` will control all animations within the 'shiny' application.
`session`	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message "lottie_js_setDirection" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

See Also

lottie_animation_methods for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("forwards", "Play Forwards"),
  actionButton("backwards", "Play Backwards")
)

server <- function(input, output, session) {
  observeEvent(input$forwards, {
    lottie_setDirection(direction = 1, name = "my_animation")
  })

  observeEvent(input$backwards, {
    lottie_setDirection(direction = -1, name = "my_animation")
  })
}

shinyApp(ui, server)
```

---

lottie_setLoop *Adjust 'Lottie' Animation Looping*

Description

Adjust the looping behaviour of a 'Lottie' animation.

Usage

```
lottie_setLoop(flag, name = "all", session = shiny::getDefaultReactiveDomain())
```

Arguments

flag	Logical value specifying whether a 'Lottie' animation should loop (TRUE) or not (FALSE).
name	A character string specifying the name of the 'Lottie' animation to control. The default of "all" will control all animations within the 'shiny' application.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message "lottie_js_setLoop" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

See Also

lottie_animation_methods for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("play", "Play"),
  actionButton("loopOn", "Loop On"),
  actionButton("loopOff", "Loop Off")
)

server <- function(input, output, session) {
  observeEvent(input$play, {
    # Non-looped animations can not be resumed without first being stopped
    lottie_stop(name = "my_animation")
    lottie_play(name = "my_animation")
  })

  observeEvent(input$loopOn, {
    lottie_setLoop(flag = TRUE, name = "my_animation")
  })

  observeEvent(input$loopOff, {
    lottie_setLoop(flag = FALSE, name = "my_animation")
  })
}
```

```
shinyApp(ui, server)
```

`lottie_setSpeed` *Adjust 'Lottie' Animation Speed*

Description

Adjust the speed of an existing 'Lottie' animation.

Usage

```
lottie_setSpeed(
  speed = 1,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`speed`	A numeric specifying the desired animation speed.
`name`	A character string specifying the name of the 'Lottie' animation to control. The default of `"all"` will control all animations within the 'shiny' application.
`session`	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message `"lottie_js_setSpeed"` containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

Note

A speed of 1 will apply the default animation speed. Use a value between 0 and 1 for a slower animation speed. Applying a negative speed will also reverse the playback direction.

See Also

`lottie_animation_methods` for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  numericInput("speed", "Speed", value = 1),
  actionButton("updateSpeed", "Update Speed")
)

server <- function(input, output, session) {
  observeEvent(input$updateSpeed, {
    lottie_setSpeed(speed = input$speed, name = "my_animation")
  })
}

shinyApp(ui, server)
```

---

lottie_setSubframe *Set 'Lottie' Animation Subframe Rendering*

Description

Adjust the subframe rendering of a 'Lottie' animation.

Usage

```
lottie_setSubframe(
  flag,
  name = "all",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

flag	A logical value specifying whether a 'Lottie' animation should use subframe rendering (TRUE) or not (FALSE).
name	A character string specifying the name of the 'Lottie' animation to control. The default of "all" will control all animations within the 'shiny' application.
session	The 'shiny' session object. Defaults to the current reactive domain.

Details

Sends a custom session message "lottie_js_setSubframe" containing the function arguments.

Value

This function is called for a side effect, and so there is no return value.

See Also

lottie_animation_methods for similar methods.

Examples

```
library(shiny)
library(shinyLottie)

ui <- fluidPage(
  include_lottie(),
  lottie_animation(
    path = "shinyLottie/example.json",
    name = "my_animation"
  ),
  actionButton("subframeOn", "Subframe On"),
  actionButton("subframeOff", "Subframe Off")
)

server <- function(input, output, session) {
  observeEvent(input$subframeOn, {
    lottie_setSubframe(flag = TRUE, name = "my_animation")
  })

  observeEvent(input$subframeOff, {
    lottie_setSubframe(flag = FALSE, name = "my_animation")
  })
}

shinyApp(ui, server)
```

Index