Package 'bsplus'

June 25, 2020

Type Package

Title Adds Functionality to the R Markdown + Shiny Bootstrap Framework

Version 0.1.2

Description The Bootstrap framework lets you add some JavaScript functionality to your web site by adding attributes to your HTML tags - Bootstrap takes care of the JavaScript <https://getbootstrap.com/javascript>. If you are using R Markdown or Shiny, you can use these functions to create collapsible sections, accordion panels, modals, tooltips, popovers, and an accordion sidebar framework (not described at Bootstrap site).

License MIT + file LICENSE

LazyData TRUE

Depends R (>= 3.3.0)

Imports htmltools, magrittr, purrr, lubridate, stringr, rmarkdown, methods, glue, jsonlite

URL https://github.com/ijlyttle/bsplus

BugReports https://github.com/ijlyttle/bsplus/issues

RoxygenNote 7.1.0

Encoding UTF-8

Suggests testthat, shiny, covr, knitr

NeedsCompilation no

Author Ian Lyttle [aut, cre, cph] (<https://orcid.org/0000-0001-9962-4849>), Schneider Electric [cph], Alex Shum [ctb], Emily Bosak [ctb]

Maintainer Ian Lyttle <ian.lyttle@schneider-electric.com>

Repository CRAN

Date/Publication 2020-06-25 06:10:03 UTC

R topics documented:

bs_accordion . . . 2
bs_accordion_sidebar . . . 3
bs_append.bsplus_carousel . . . 5
bs_button . . . 7
bs_carousel_caption . . . 8
bs_carousel_image . . . 8
bs_collapse . . . 9
bs_embed_popover . . . 10
bs_embed_tooltip . . . 11
bs_modal . . . 12
bs_panel . . . 14
bs_set_data . . . 15
render_html_fragment . . . 16
shinyInput_label_embed . . . 17
shiny_iconlink . . . 18
%>% . . . 18

Index **19**

`bs_accordion` *Accordion panel-group*

Description

An accordion is a set of collapsible panels where, at most, one panel-body is visible.

Usage

```
bs_accordion(id)

## S3 method for class 'bsplus_accordion'
bs_append(tag, title, content, ...)

## S3 method for class 'bsplus_accordion'
bs_set_opts(tag, panel_type = "primary", use_heading_link = TRUE, ...)
```

Arguments

`id`	character, unique id for accordion , also serves as root id for panels appended using `bs_append()`
`tag`	htmltools::[tag][htmltools::tag], accordion to which to append a panel
`title`	character (HTML) or htmltools::[tagList][htmltools::tagList], title for the panel heading
`content`	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the panel body

`...` other arguments (not used)

`panel_type` character, one of the standard Bootstrap types `c("default","primary","success","info","warning"`

`use_heading_link`
logical, indicates whether to make the entire panel heading clickable.

Details

All of these functions return a `bsplus_accordion` object (which is also an htmltools::[tag][htmltools::tag],), so you can compose an accordion by piping. There are three parts to this system:

1. A constructor function for the accordion, `bs_accordion()`
2. A function to set options for subsequent panels, `bs_set_opts()`
3. A function to append a panel to the group, `bs_append()`

The verb *append* is used to signify that you can append an arbitrary number of panels to an accordion.

For the constructor, `bs_accordion()`, it is your responsibility to ensure that `id` is unique among HTML elements in your page. If you have non-unique `id`'s, strange things may happen to your page.

Value

`bsplus_accordion` object (htmltools::[tag][htmltools::tag],)

See Also

http://getbootstrap.com/docs/3.3/javascript/#collapse-example-accordion

Examples

```
bs_accordion(id = "meet_the_beatles") %>%
  bs_set_opts(panel_type = "success", use_heading_link = TRUE) %>%
  bs_append(title = "John Lennon", content = "Rhythm guitar, vocals") %>%
  bs_set_opts(panel_type = "info") %>%
  bs_append(title = "Paul McCartney", content = "Bass guitar, vocals")
```

---

bs_accordion_sidebar *Accordion-sidebar panel-group*

Description

Combines Bootstrap accordion with the functionality of shiny::[sidebarLayout][shiny::sidebarLayout], allowing you to add another dimension to your **shiny** apps.

Usage

```
bs_accordion_sidebar(
  id,
  spec_side = c(width = 4, offset = 0),
  spec_main = c(width = 8, offset = 0),
  position = c("left", "right")
)

use_bs_accordion_sidebar()

## S3 method for class 'bsplus_accordion_sidebar'
bs_append(tag, title_side, content_side, content_main, ...)

## S3 method for class 'bsplus_accordion_sidebar'
bs_set_opts(
  tag,
  panel_type_active = "success",
  panel_type_inactive = "primary",
  use_main_enclosure = TRUE,
  ...
)
```

Arguments

id	character, unique id for accordion-sidebar , also serves as root id for panels appended using bs_append()
spec_side	numeric, column specification for sidebar panels
spec_main	numeric, column specification for main panels
position	character, indicates where to put the sidebar panels with respect to the main panels
tag	htmltools::[tag][htmltools::tag], accordion-sidebar to which to append a panel
title_side	character (HTML) or htmltools::[tagList][htmltools::tagList], title for the sidebar panel
content_side	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the sidebar panel
content_main	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the main panel
...	other arguments (not used)
panel_type_active	character, indicated bootstrap type for active-panel header, one of c("default","primary","success",
panel_type_inactive	character, indicated bootstrap type for inactive-panel header, one of c("default","primary","success"
use_main_enclosure	logical, indicates if main content is to be wrapped in a Bootstrap panel

Details

If you use a `bs_accordion_sidebar()`, you will have to call the function `use_bs_accordion_sidebar()` somewhere in your UI. This attaches some JavaScript needed for your accordion sidebar to work properly.

All of these functions return a `bsplus_accsidebar` object, (which is also an htmltools::[tag][htmltools::tag],), so you can compose an accordion sidebar by piping. There are three parts to this system:

1. A constructor function for the accordion-sidebar, `bs_accordion_sidebar()`
2. A function to set options for subsequent panels, `bs_set_opts()`
3. A function to append a panel-set to an accordion-sidebar, `bs_append()`

The verb *append* is used to signify that you can append an arbitrary number of panels-sets to an accordion-sidebar.

For the constructor, `bs_accordion_sidebar()`, it is your responsibility to ensure that `id` is unique among HTML elements in your page. If you have non-unique `id`'s, strange things may happen to your page.

Value

`bsplus_accsidebar` object (htmltools::[tag][htmltools::tag],)

Examples

```
bs_accordion_sidebar(id = "meet_the_beatles") %>%
  bs_append(
    title_side = "John Lennon",
    content_side = "Rhythm guitar, vocals",
    content_main = "Dear Prudence"
  ) %>%
  bs_append(
    title_side = "Paul McCartney",
    content_side = "Bass guitar, vocals",
    content_main = "Blackbird"
  )
## Not run:
use_bs_accordion_sidebar()

## End(Not run)
```

---

bs_append.bsplus_carousel

Carousel

Description

A carousel is used to enclose a set of (typically) images, providing controls to move slides back-and-forth.

Usage

```
## S3 method for class 'bsplus_carousel'
bs_append(tag, content, caption = NULL, ...)

bs_carousel(id, use_indicators = FALSE, use_controls = TRUE)
```

Arguments

tag	htmltools::[tag][htmltools::tag], carousel to which to append a panel
content	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the slide
caption	character (HTML) or htmltools::[tagList][htmltools::tagList], caption for the slide
...	other args (not used)
id	character, unique id for accordion , also serves as root id for slides appended using `bs_append()`
use_indicators	logical, denotes use of slide-position indicators (dots)
use_controls	logical, denotes use of controls (chevrons at sides)

Details

All of these functions return a `bsplus_carousel` object (which is also an htmltools::[tag][htmltools::tag],), so you can compose a carousel by piping. There are two parts to this system:

1. A constructor function for the carousel, `bs_carousel()`
2. A function to append a slide to the carousel, `bs_append()`

The verb *append* is used to signify that you can append an arbitrary number of slides to a carousel.

For the constructor, `bs_carousel()`, it is your responsibility to ensure that `id` is unique among HTML elements in your page. If you have non-unique `id`'s, strange things may happen to your page.

Value

`bsplus_carousel` object (htmltools::[tag][htmltools::tag],)

See Also

http://getbootstrap.com/docs/3.3/javascript/#carousel, `bs_carousel_image()`, `bs_carousel_caption()`

Examples

```
bs_carousel(id = "with_the_beatles") %>%
  bs_append(content = bs_carousel_image(src = "img/john.jpg")) %>%
  bs_append(content = bs_carousel_image(src = "img/paul.jpg")) %>%
  bs_append(content = bs_carousel_image(src = "img/george.jpg")) %>%
  bs_append(content = bs_carousel_image(src = "img/ringo.jpg"))
```

bs_button *Button*

Description

This function makes it a little easier to make Bootstrap-friendly buttons; it wraps `htmltools::tags` for buttons.

Usage

```
bs_button(
  label,
  button_type = c("default", "primary", "success", "info", "warning", "danger"),
  button_size = c("default", "large", "small", "extra-small"),
  ...
)
```

Arguments

`label`	character (HTML), button label
`button_type`	character, one of the standard Bootstrap types
`button_size`	character, size of the button
`...`	attributes (named arguments) and children (unnamed arguments) of the button, passed to

Value

Object with S3 class, `shiny.tag`, <button/>.

See Also

http://getbootstrap.com/docs/3.3/css/#buttons

Examples

```
bs_button("Click me", button_type = "primary", button_size = "small")
```

bs_carousel_caption *Carousel caption*

Description

Helper function to generate HTML for a carousel caption.

Usage

```
bs_carousel_caption(title = NULL, body = NULL)
```

Arguments

title	character, caption title
body	character, caption body

Value

htmltools::[tag][htmltools::tag] for carousel caption

See Also

bs_carousel()

bs_carousel_image *Carousel image*

Description

Helper function to generate HTML for a carousel image.

Usage

```
bs_carousel_image(...)
```

Arguments

...	additional arguments passed to htmltools::[tag][htmltools::tag]$img, typically includes src

Details

This function wraps htmltools::[tag][htmltools::tag]$img, but adding a class to center the image in the carousel.

Value

htmltools::[tag][htmltools::tag], </img>

See Also

bs_carousel()

`bs_collapse` *Collapsible element*

Description

This is useful for content that you may wish to be hidden when the page is initialized, but that can be revealed (and subsequently hidden) by clicking a button or a link.

Usage

```
bs_collapse(id, content = NULL, show = FALSE)

bs_attach_collapse(tag, id_collapse)
```

Arguments

`id`	character, unique id for the collapsible
`content`	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the collapsible
`show`	logical, indicates if collapsible is shown when page is initialized
`tag`	htmltools::[tag][htmltools::tag], button or link to which to attach a collapsible
`id_collapse`	character, `id` of the collapsible to attach

Details

There are two parts to this system:

1. A collapsible , created using `bs_collapse()`
2. At least one button (<button/>) or link (<a/>) to which the `id` of the collapsible is attached, using `bs_attach_collapse()`

The verb *attach* is used to signify that we are attaching the `id` of our collapsible to the tag in question (a button or a link). Note that you can attach the `id` of a collapsible to more than one button or link.

It is your responsibility to ensure that `id` is unique among HTML elements in your page. If you have non-unique `id`'s, strange things may happen to your page.

Value

`bs_collapse()` htmltools::[tag][htmltools::tag],

`bs_attach_collapse()` htmltools::[tag][htmltools::tag], modified copy of `tag` (button or link)

See Also

https://getbootstrap.com/docs/3.3/javascript/#collapse

Examples

```
library("htmltools")

bs_collapse(id = "id_yeah", "Yeah Yeah Yeah")

bs_button("She Loves You", button_type = "primary") %>%
  bs_attach_collapse("id_yeah")
```

---

bs_embed_popover *Popover*

Description

A popover can be a useful way to add a somewhat-verbose explanation to a tag.

Usage

```
bs_embed_popover(tag, title = NULL, content = NULL, placement = "top", ...)

use_bs_popover()
```

Arguments

`tag`	htmltools::[tag][htmltools::tag], generally <button/> or <a/>, into which to embed the popover
`title`	character, title for the popover, generally text
`content`	character, content for the popover body, can be HTML
`placement`	character, placement of the popover with respect to `tag`
`...`	other named arguments, passed to `bs_set_data()`

Details

To activate the use of popovers in your page, you will need to call the `use_bs_popover()` function somewhere.

The verb *embed* is used to signify that you are embedding information into a `tag`. This implies that you can embed, at most, one "thing" into a particular `tag`. You should not, for example, expect to embed both a tooltip and a popover into a `tag`.

Value

htmltools::[tag][htmltools::tag], modified copy of `tag`

See Also

`bs_embed_tooltip()`, http://getbootstrap.com/docs/3.3/javascript/#popovers

Examples

```
library("htmltools")

bs_button("A button") %>%
  bs_embed_popover(title = "I'm a popover", content = "Really!")
```

---

bs_embed_tooltip *Tooltip*

Description

A tooltip can be a useful way to add a few words of explanation to a tag.

Usage

```
bs_embed_tooltip(tag, title = "", placement = "top", ...)

use_bs_tooltip()
```

Arguments

`tag`	htmltools::[tag][htmltools::tag], generally <button/> or <a/>, into which to embed the tooltip
`title`	character, title for the tooltip
`placement`	character, placement of the tooltip with respect to `tag`
`...`	other named arguments, passed to `bs_set_data()`

Details

To activate the use of tooltips in your page, you will need to call the `use_bs_tooltip()` function somewhere.

The verb *embed* is used to signify that you are embedding information into a `tag`. This implies that you can embed, at most, one "thing" into a particular `tag`. You should not, for example, expect to embed both a tooltip and a popover into a `tag`.

Value

htmltools::[tag][htmltools::tag], modified copy of `tag`

See Also

bs_embed_popover(), http://getbootstrap.com/docs/3.3/javascript/#tooltips

Examples

```
library("htmltools")
bs_button("I'm a button") %>%
  bs_embed_tooltip(title = "I'm a tooltip")
```

---

bs_modal *Modal window*

Description

Modal windows are useful to make detailed explanations, and are typically attached to buttons or links. Thus, there are two parts to this system:

Usage

```
bs_modal(
  id,
  title,
  body,
  footer = bs_modal_closebutton(label = "Close"),
  size = c("medium", "large", "small")
)

bs_modal_closebutton(label = "Close", title)

bs_attach_modal(tag, id_modal)
```

Arguments

id	character, unique id for the modal window
title	character, title for the modal window (this argument is deprecated for bs_modal_closebutton, use label instead)
body	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the body of the modal window
footer	character (HTML) or htmltools::[tagList][htmltools::tagList], content for the footer of the modal window
size	character, size of the modal window
label	character (HTML), label for the close-button
tag	htmltools::[tag][htmltools::tag], button or link to which to attach the modal window
id_modal	character, unique id of modal window to attach

Details

1. A modal window, created using `bs_modal()`
2. At least one button or link to which the `id` of the modal window is attached, using `bs_attach_modal()`

The verb *attach* is used to signify that we are attaching the `id` of our modal window to the `tag` in question (generally a button or a link). This implies that you can attach the `id` of a modal window to more than one button or link.

It is your responsibility to ensure that `id` is unique among HTML elements in your page. If you have non-unique `id`'s, strange things may happen to your page.

Your code may be cleaner if you can import the content for the modal body from an external source. Here, the function shiny::[includeMarkdown][shiny::includeMarkdown] be useful.

If you want to compose your own footer for the modal window, the function `bs_modal_closebutton()` can be useful.

Value

`bs_modal()` htmltools::[tag][htmltools::tag],

`bs_attach_modal()` htmltools::[tag][htmltools::tag], modified copy of `tag`

`bs_modal_closebutton()` htmltools::[tag][htmltools::tag], <button/>

See Also

shiny::[includeMarkdown][shiny::includeMarkdown]

Examples

```
library("htmltools")
library("shiny")

bs_modal(id = "modal", title = "I'm a modal", body = "Yes, I am.")
bs_button("Click for modal") %>%
  bs_attach_modal(id_modal = "modal")

bs_modal(
  id = "modal_large",
  title = "I'm a modal",
  size = "large",
  body = includeMarkdown(system.file("markdown", "modal.md", package = "bsplus"))
)
bs_button("Click for modal") %>%
  bs_attach_modal(id_modal = "modal_large")
```

bs_panel *Panel*

Description

This function makes it a little easier to make Bootstrap-friendly panels; it wraps `htmltools::tags` for panels

Usage

```
bs_panel(
  id = NULL,
  panel_type = c("default", "primary", "success", "info", "warning", "danger"),
  heading = NULL,
  body = NULL,
  ...,
  footer = NULL
)
```

Arguments

`id`	character, unique identifier
`panel_type`	character, one of the standard Bootstrap types
`heading`	character (HTML) or htmltools::tagList(), content for the heading
`body`	character (HTML) or htmltools::tagList(), content for the body
`...`	character (HTML) or htmltools::tagList(), other content
`footer`	character (HTML) or htmltools::tagList(), content for the footer

Value

Object with S3 class, `shiny.tag`,

See Also

http://getbootstrap.com/docs/3.3/css/#panels

Examples

```
library("htmltools")

bs_panel(
  panel_type = "primary",
  heading = tags$h3("title"),
  body = tags$p("Some very important content")
)
```

bs_set_data *Sets Bootstrap data- and aria- attributes.*

Description

Helper function to manage attributes for Bootstrap's JavaScript components.

Usage

```
bs_set_data(tag, ...)

bs_set_aria(tag, ...)
```

Arguments

tag	htmltools::[tag][htmltools::tag]
...	named arguments used to set the attributes of tag

Details

One of the mechanisms used by the API for Boostrap JavaScript-components is an html elements' attributes. These attribute names are prefixed with "data-" or "aria-", depending on the function.

When expressed in html, attributes themselves have the properties:

- Logical values are expressed as "true" or "false".
- Time durations are expressed as number of milliseconds.
- Vector (non scalar) values are expressed in a space-delimited list.

The purpose of this function is to let you express these values in ways familiar to you as an R user. For example:

- Logical values can be expressed as logicals: TRUE or FALSE.
- Time durations can be expressed using lubridate durations.
- Vector (non scalar) values can be expressed as vectors.

Note that this returns a modified copy of the tag sent to it, so it is pipeable.

Value

htmltools::[tag][htmltools::tag], modified copy of tag

See Also

Bootstrap JavaScript Components

Examples

```
library("htmltools")
library("lubridate")

tags$div() %>%
  bs_set_data(
    target = "#foobar",
    delay = dseconds(1),
    placement = c("right", "auto")
  ) %>%
  bs_set_aria(expanded = FALSE)
```

`render_html_fragment` *Renders and returns an HTML fragment*

Description

This is a wrapper around the rmarkdown::[render][rmarkdown::render] function. The principal difference is that the function is designed to return an HTML fragment (rather than writing to a file). This function is useful to populate the content of a modal window.

Usage

```
render_html_fragment(input, output_format = rmarkdown::html_fragment(), ...)
```

Arguments

`input`	character, path to input file
`output_format`	rmarkdown output format, provided so you can specify arguments
`...`	other arguments passed to rmarkdown::[render][rmarkdown::render]

Details

This function is being deprecated in favor of shiny::[includeMarkdown][shiny::includeMarkdown]

Value

htmltools::[tag][htmltools::tag]

Examples

```
## Not run:
my_file <- system.file("markdown", "modal.md", package = "bsplus")
render_html_fragment(my_file)

## End(Not run)
```

shinyInput_label_embed *Embed an element into the label of a Shiny-input tag*

Description

The element embedded into the Shiny input will be pulled to the right edge of the label.

Usage

```
shinyInput_label_embed(tag, element)
```

Arguments

tag	Shiny input, such as shiny::[numericInput][shiny::numericInput]
element	htmltools::[tag][htmltools::tag] to be embedded into label of tag

Details

To promote consistency, the following convention is proposed:

For links (activated by clicking), embed a `shiny::icon("info-circle")`; this is the default for `shiny_iconlink()`. For elements activated by hovering, embed a `shiny::icon("info")`.

Value

Shiny input, modified copy of tag

See Also

`shiny_iconlink()`

Examples

```
library("shiny")

numericInput(inputId = "foo", label = "Enter a number", value = 0) %>%
  shinyInput_label_embed(
    shiny_iconlink() %>%
    bs_embed_popover(title = "Number", content = "Not a complex number")
  )
```

shiny_iconlink *Create link containing Shiny icon*

Description

You can use this helper function to wrap link element around a shiny::[icon][shiny::icon]. It may be useful to attach a modal window to (or embed a popover into) into such a link.

Usage

```
shiny_iconlink(name = "info-circle", id = NULL, ...)
```

Arguments

name	character, name of the icon, passed to shiny::[icon][shiny::icon]
id	character, option ID for the link
...	other arguments passed to shiny::[icon][shiny::icon]

Value

htmltools::[tag][htmltools::tag], <a/>

See Also

shinyInput_label_embed(), shiny::[icon][shiny::icon], bs_attach_modal(), bs_embed_popover(), bs_embed_tooltip()

Examples

```
shiny_iconlink()

shiny_iconlink() %>%
  bs_embed_popover(title = "Help!", content = "I need somebody")
```

%>% *Pipe*

Description

Like dplyr, bsplus also uses the pipe function, \%>\% to turn function composition into a series of imperative statements.

Arguments

lhs, rhs	An object and a function to apply to it

Index

%>%, 18

bs_accordion, 2
bs_accordion_sidebar, 3
bs_append.bsplus_accordion
        *(*bs_accordion*)*, 2
bs_append.bsplus_accordion_sidebar
        *(*bs_accordion_sidebar*)*, 3
bs_append.bsplus_carousel, 5
bs_attach_collapse *(*bs_collapse*)*, 9
bs_attach_modal *(*bs_modal*)*, 12
bs_attach_modal(), *18*
bs_button, 7
bs_carousel
        *(*bs_append.bsplus_carousel*)*, 5
bs_carousel(), *8, 9*
bs_carousel_caption, 8
bs_carousel_caption(), *6*
bs_carousel_image, 8
bs_carousel_image(), *6*
bs_collapse, 9
bs_embed_popover, 10
bs_embed_popover(), *12*, *18*
bs_embed_tooltip, 11
bs_embed_tooltip(), *11*, *18*
bs_modal, 12
bs_modal_closebutton *(*bs_modal*)*, 12
bs_panel, 14
bs_set_aria *(*bs_set_data*)*, 15
bs_set_data, 15
bs_set_opts.bsplus_accordion
        *(*bs_accordion*)*, 2
bs_set_opts.bsplus_accordion_sidebar
        *(*bs_accordion_sidebar*)*, 3

htmltools::tagList(), *14*
htmltools::tags, *7*, *14*

render_html_fragment, 16

shiny_iconlink, 18
shiny_iconlink(), *17*
shinyInput_label_embed, 17
shinyInput_label_embed(), *18*

use_bs_accordion_sidebar
        *(*bs_accordion_sidebar*)*, 3
use_bs_popover *(*bs_embed_popover*)*, 10
use_bs_tooltip *(*bs_embed_tooltip*)*, 11