

Package: bootstrict (via r-universe)

September 15, 2026

Title Strict Bootstrap 5.3 Widgets for Shiny

Version 0.9.9.9000

Description A complete, faithful implementation of the Bootstrap 5.3 component, layout and form library as Shiny UI functions. Every widget mirrors the Bootstrap 5.3 HTML structure one-to-one so that a designer's mockup and SASS variables can be dropped into a Shiny application with minimal deviation from Shiny's own conventions. Interactive components report their state to the server and can be controlled server-side through update functions and proxies. Bootstrap 5.3 itself is vendored and compiled with 'sass', so a designer-provided SASS variable sheet composes natively and the Bootstrap version reaching the browser is fixed by this package.

License MIT + file LICENSE

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Depends R (>= 4.1.0)

Imports htmltools (>= 0.5.4), rlang (>= 1.1.0), sass (>= 0.4.0), shiny (>= 1.7.0)

Suggests golem, knitr, rmarkdown, callr, chromote, httpuv, pkgload, testthat (>= 3.0.0), tibble, withr

VignetteBuilder knitr

Config/testthat/edition 3

URL <https://github.com/thinkr-open/bootstrict>,
<https://thinkr-open.github.io/bootstrict/>

BugReports <https://github.com/thinkr-open/bootstrict/issues>

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository <https://thinkr-open.r-universe.dev>

Date/Publication 2026-09-15 11:37:55 UTC

RemoteUrl <https://github.com/ThinkR-open/bootstrict>

RemoteRef HEAD

RemoteSha 73e0a1c623c49d478659fe2eb115eacea3a49f89

Contents

bootstrap_dep	4
bootstrap_version	5
bootstrict_dep	5
bootstrict_theme	6
bs_accordion	7
bs_alert	8
bs_badge	9
bs_blockquote	10
bs_breadcrumb	10
bs_button	11
bs_button_group	12
bs_card	13
bs_carousel	14
bs_checkbox_group_input	16
bs_checkbox_input	17
bs_close_button	18
bs_col	19
bs_collapse	20
bs_collapse_trigger	21
bs_color_input	21
bs_container	22
bs_date_input	23
bs_date_range_input	24
bs_display_heading	25
bs_download_button	26
bs_download_link	27
bs_dropdown	28
bs_feedback	30
bs_figure	31
bs_file_input	32
bs_floating_label	33
bs_form	34
bs_form_label	34
bs_form_text	35
bs_hstack	35
bs_icon_link	36
bs_img	37
bs_input_group	38
bs_lead	39
bs_list_group	39

bs_list_unstyled	41
bs_modal	41
bs_modal_trigger	43
bs_nav	44
bs_nav_dropdown	45
bs_navbar	46
bs_notify_toast	48
bs_numeric_input	49
bs_offcanvas	50
bs_offcanvas_trigger	51
bs_page	52
bs_page_item	53
bs_pagination	54
bs_pagination_numbered	55
bs_password_input	56
bs_placeholder	57
bs_popover	57
bs_progress	58
bs_radio_button_input	60
bs_radio_input	61
bs_range_input	62
bs_ratio	63
bs_row	64
bs_scrollspy	65
bs_select_input	66
bs_spinner	67
bs_table	68
bs_tabset	69
bs_text_input	70
bs_textarea_input	71
bs_toast	72
bs_toast_container	73
bs_tooltip	74
bs_valid_feedback	74
bs_visually_hidden	75
bs_vr	76
close_bs_alert	76
parse_scss_variables	77
set_bs_color_mode	78
set_bs_validation	78
show_bs_dropdown	79
show_bs_modal	80
show_bs_offcanvas	81
show_bs_toast	81
update_bs_accordion	82
update_bs_carousel	83
update_bs_collapse	84
update_bs_color	84

update_bs_date_input	85
update_bs_list_group	86
update_bs_nav	87
update_bs_pagination	87
update_bs_progress	88
update_bs_range	89
update_bs_tabset	90
update_bs_toggle_buttons	90
use_bootstrap	91
use_bootstrap_golem	92
Index	93

bootstrap_dep	<i>The Bootstrap HTML dependency for a theme</i>
---------------	--

Description

Compiles theme against the vendored Bootstrap SASS tree and returns the resulting stylesheet together with Bootstrap’s JavaScript bundle. The dependency is named "bootstrap" at the vendored version, so it supersedes the Bootstrap 3 stylesheet Shiny’s own page functions attach.

Usage

```
bootstrap_dep(theme = bootstrict_theme())
```

Arguments

theme A `bootstrict_theme()`.

Details

`bs_page()` and friends call this for you; use it directly when building a page by hand. Compiled stylesheets are cached per theme for the session.

Value

An `htmltools::htmlDependency`.

Examples

```
if (interactive()) {
  bootstrap_dep(bootstrict_theme(primary = "#ff6600"))
}
```

bootstrap_version	<i>The Bootstrap version bootstrict ships</i>
-------------------	---

Description

bootstrict vendors its own copy of Bootstrap under `inst/lib/bootstrap` rather than compiling against whatever version a theming package happens to bundle, so this is the version that actually reaches the browser.

Usage

```
bootstrap_version()
```

Value

The Bootstrap version, as a string.

Examples

```
bootstrap_version()
```

bootstrict_dep	<i>The bootstrict HTML dependency (Shiny input bindings + supporting CSS)</i>
----------------	---

Description

Loads every JavaScript file shipped in `inst/assets/js`, forcing `bootstrict-core.js` first (it defines the `window.bootstrict` namespace that the per-component binding files rely on), followed by the supporting stylesheet. Component binding files are discovered dynamically, so adding a new binding is just a matter of dropping a `.js` file in that directory. The dependency is built once per session and cached (every widget calls this, and the installed files cannot change mid-session).

Usage

```
bootstrict_dep()
```

Value

An [htmltools::htmlDependency](#).

Examples

```
bootstrict_dep()
```

bootstrict_theme	Create a Bootstrap 5.3 theme for a bootstrict UI
------------------	--

Description

Collects SASS variable overrides – from a designer’s exported sheet, from `...`, or both – for `bootstrap_dep()` to compile against the Bootstrap tree bootstrict vendors. Variables from `variables` are merged with (and overridden by) any passed through `...`.

Usage

```
bootstrict_theme(..., variables = NULL)
```

Arguments

<code>...</code>	Named SASS variables, e.g. <code>primary = "#ff6600"</code> . Names use the Bootstrap convention without the leading <code>\$</code> .
<code>variables</code>	Optional path to a <code>.scss</code> variable sheet, or a named list (as returned by <code>parse_scss_variables()</code>).

Details

Values are placed in the SASS layer that can actually compile them. A value built only from literals or from the sheet’s own variables (`$primary: #ff6600, $link-color: $primary`) goes to the *defaults* layer, where it is set before Bootstrap derives `$theme-colors` and the rest from it. A value referring to one of Bootstrap’s own variables (`$link-hover-color: shade-color($primary, 20%)`, with no `$primary` in the sheet) cannot go there – Bootstrap’s variables are not defined yet – so it goes to the *declarations* layer, after the configuration block.

One consequence is worth knowing: a theme colour redefined from one of Bootstrap’s own variables (`$secondary: $gray-600`) lands in the declarations layer, after `$theme-colors` has been built, so it will not restyle `.btn-secondary`. Give the sheet its own literal (or define the variable it refers to) when that matters.

Value

A `bootstrict_theme` object.

Examples

```
bootstrict_theme(primary = "#ff6600", "font-size-base" = "1rem")
```

bs_accordionBootstrap accordion

Description

A vertically collapsing set of panels. The value(s) of the currently open panel(s) are reported to the server as `input$id`, and can be driven server-side with [update_bs_accordion\(\)](#).

Usage

```
bs_accordion(  
  id,  
  ...,  
  open = NULL,  
  multiple = FALSE,  
  flush = FALSE,  
  class = NULL  
)
```

```
bs_accordion_panel(  
  title,  
  ...,  
  value = NULL,  
  icon = NULL,  
  class = NULL,  
  body_class = NULL  
)
```

Arguments

<code>id</code>	Accordion id; open panel value(s) available as <code>input\$id</code> .
<code>...</code>	Panels built with bs_accordion_panel() .
<code>open</code>	Panel value(s) open initially. Use <code>TRUE</code> to open all (only sensible with <code>multiple = TRUE</code>), <code>FALSE</code> / <code>NULL</code> for none.
<code>multiple</code>	If <code>TRUE</code> , panels stay open independently (Bootstrap "always open"). Otherwise opening one closes the others.
<code>flush</code>	If <code>TRUE</code> , render edge-to-edge without borders (<code>.accordion-flush</code>).
<code>class</code>	Extra classes.
<code>title</code>	Panel header content.
<code>value</code>	Panel identifier reported to the server (defaults to <code>title</code>).
<code>icon</code>	Optional icon placed before the title.
<code>body_class</code>	Extra classes for the panel body.

Value

An accordion tag.

Examples

```
bs_accordion(
  "acc",
  bs_accordion_panel("First", "Panel one body", value = "one"),
  bs_accordion_panel("Second", "Panel two body", value = "two"),
  open = "one"
)
```

bs_alert

Bootstrap alert

Description

Given an id, a dismissible alert reports whether it is still on the page as `input$id` (TRUE until it is dismissed, FALSE once it is) and can be closed from the server with [close_bs_alert\(\)](#).

Usage

```
bs_alert(..., id = NULL, color = "primary", dismissible = FALSE, class = NULL)
```

```
bs_alert_heading(..., level = 4, class = NULL)
```

```
bs_alert_link(..., href = "#", class = NULL)
```

Arguments

<code>...</code>	Alert content and named HTML attributes.
<code>id</code>	Optional alert id. Its visibility is reported as <code>input\$id</code> .
<code>color</code>	Theme colour (<code>.alert-*</code>). Defaults to "primary".
<code>dismissible</code>	If TRUE, add a close button and fade-out behaviour.
<code>class</code>	Extra classes.
<code>level</code>	Heading level (1-6).
<code>href</code>	Link target.

Value

An alert tag.

See Also

[close_bs_alert\(\)](#)

Examples

```
bs_alert("Well done!", color = "success")
bs_alert("Heads up.", color = "warning", dismissible = TRUE)
bs_alert(
  bs_alert_heading("Well done"),
  "You read ", bs_alert_link("the docs", href = "#"), ". ",
  color = "success"
)
```

bs_badge	<i>Bootstrap badge</i>
----------	------------------------

Description

A small count / labelling component rendered as a `<span class="badge">`.

Usage

```
bs_badge(..., color = "primary", pill = FALSE, class = NULL)
```

Arguments

<code>...</code>	Badge content and named HTML attributes.
<code>color</code>	Theme colour (<code>.text-bg-*</code>). Defaults to "primary".
<code>pill</code>	If TRUE, render with fully rounded corners (<code>.rounded-pill</code>).
<code>class</code>	Extra classes.

Value

A badge tag.

Examples

```
bs_badge("New", color = "success")
```

bs_blockquote	<i>Bootstrap blockquote</i>
---------------	-----------------------------

Description

Bootstrap blockquote

Usage

```
bs_blockquote(..., footer = NULL, class = NULL)
```

Arguments

...	Quote content and named HTML attributes applied to the <blockquote>.
footer	Optional source/attribution rendered in a .blockquote-footer.
class	Extra classes applied to the <blockquote>.

Value

A figure tag wrapping the blockquote.

Examples

```
bs_blockquote("A well-known quote.", footer = "Someone famous")
```

bs_breadcrumb	<i>Bootstrap breadcrumb</i>
---------------	-----------------------------

Description

A navigation hierarchy. Compose with [bs_breadcrumb_item\(\)](#).

Usage

```
bs_breadcrumb(..., divider = NULL, label = "breadcrumb", class = NULL)
```

```
bs_breadcrumb_item(..., active = FALSE, href = NULL, class = NULL)
```

Arguments

...	Breadcrumb items built with <code>bs_breadcrumb_item()</code> , plus named HTML attributes applied to the <code><nav></code> .
divider	Optional custom divider character (e.g. ">"). When a string, it is set via the <code>--bs-breadcrumb-divider</code> CSS variable on the <code><nav></code> .
label	Accessible label for the <code><nav></code> (<code>aria-label</code>).
class	Extra classes applied to the <code><nav></code> .
active	If TRUE, mark the item as the current page (no link).
href	Optional link target. Ignored when <code>active = TRUE</code> .

Value

A breadcrumb `<nav>` tag.

Examples

```
bs_breadcrumb(
  bs_breadcrumb_item("Home", href = "#"),
  bs_breadcrumb_item("Library", active = TRUE)
)
```

bs_button

*Bootstrap button***Description**

Renders a Bootstrap 5 button. When `id` is supplied the button is wired as a Shiny action button: `input$id` holds the click count, exactly like `shiny::actionButton()`.

Usage

```
bs_button(
  id = NULL,
  label = NULL,
  ...,
  color = "primary",
  outline = FALSE,
  size = NULL,
  disabled = FALSE,
  href = NULL,
  type = "button",
  class = NULL
)
```

Arguments

id	Optional input id. When set, the click count is reported as input\$id.
label	Button label (text or tags). Can also be passed via ...
...	Additional content and named HTML attributes.
color	One of the eight Bootstrap theme colours, or "link".
outline	If TRUE, an outline button (.btn-outline-*).
size	"sm" or "lg".
disabled	If TRUE, the button is disabled.
href	Optional URL; renders an <a> styled as a button.
type	Button type attribute ("button", "submit", "reset").
class	Extra classes.

Value

A button (or anchor) tag.

Examples

```
bs_button("go", "Go", color = "primary")
bs_button(label = "Cancel", color = "secondary", outline = TRUE)
```

bs_button_group	<i>Bootstrap button group / toolbar</i>
-----------------	---

Description

Bootstrap button group / toolbar

Usage

```
bs_button_group(..., size = NULL, vertical = FALSE, label = NULL, class = NULL)

bs_button_toolbar(..., label = NULL, class = NULL)
```

Arguments

...	Buttons (bs_button()) and named HTML attributes.
size	"sm" or "lg".
vertical	If TRUE, stack vertically (.btn-group-vertical).
label	Accessible label (aria-label).
class	Extra classes.

Value

A button group / toolbar tag.

Examples

```
bs_button_group(bs_button(label = "Left"), bs_button(label = "Right"))
bs_button_toolbar(
  bs_button_group(bs_button(label = "Cut")),
  bs_button_group(bs_button(label = "Copy"))
)
```

bs_card

Bootstrap card

Description

A flexible content container. Compose with [bs_card_header\(\)](#), [bs_card_body\(\)](#), [bs_card_footer\(\)](#), [bs_card_img\(\)](#) and the card text helpers.

Usage

```
bs_card(..., color = NULL, border = NULL, class = NULL)
```

```
bs_card_body(..., class = NULL)
```

```
bs_card_header(..., class = NULL)
```

```
bs_card_footer(..., class = NULL)
```

```
bs_card_title(..., level = 5, class = NULL)
```

```
bs_card_subtitle(..., level = 6, class = NULL)
```

```
bs_card_text(..., class = NULL)
```

```
bs_card_link(..., href = "#", class = NULL)
```

```
bs_card_img(
  src,
  position = c("top", "bottom", "overlay"),
  alt = "",
  ...,
  class = NULL
)
```

```
bs_card_img_overlay(..., class = NULL)
```

```
bs_card_group(..., class = NULL)
```

Arguments

<code>...</code>	Card content (headers, bodies, images, ...) and named HTML attributes.
<code>color</code>	Background theme colour (<code>.text-bg-*</code>), one of the Bootstrap theme colours.
<code>border</code>	Border theme colour (<code>.border-*</code>).
<code>class</code>	Extra classes.
<code>level</code>	Heading level (1-6) for the card title / subtitle.
<code>href</code>	Link target for bs_card_link() .
<code>src</code>	Image source URL.
<code>position</code>	Image placement: "top", "bottom", or "overlay" (use together with bs_card_img_overlay()).
<code>alt</code>	Image alt text.

Value

A card tag.

Examples

```
bs_card(
  bs_card_header("Featured"),
  bs_card_body(
    bs_card_title("Card title"),
    bs_card_text("Some quick example text.")
  )
)

bs_card_group(
  bs_card(
    bs_card_img("cap.png", alt = ""),
    bs_card_body(
      bs_card_title("Title"),
      bs_card_subtitle("Subtitle"),
      bs_card_link("More", href = "#")
    ),
    bs_card_footer("2 days ago")
  ),
  bs_card_img("cap.png", alt = ""), bs_card_img_overlay("Over"))
```

bs_carousel

Bootstrap carousel

Description

A slideshow component for cycling through a series of items ([bs_carousel_item\(\)](#)s). The 0-based index of the active slide is reported to the server as `input$id`, and can be driven server-side with [update_bs_carousel\(\)](#).

Usage

```

bs_carousel(
  id,
  ...,
  indicators = TRUE,
  controls = TRUE,
  fade = FALSE,
  autoplay = TRUE,
  interval = NULL,
  wrap = TRUE,
  keyboard = TRUE,
  pause = "hover",
  touch = TRUE,
  dark = FALSE,
  class = NULL
)

bs_carousel_item(
  ...,
  active = FALSE,
  interval = NULL,
  caption = NULL,
  class = NULL
)

```

Arguments

<code>id</code>	Carousel id; the active slide index is available as <code>input\$id</code> .
<code>...</code>	Items built with <code>bs_carousel_item()</code> (and named HTML attributes).
<code>indicators</code>	If TRUE, render the clickable slide indicators.
<code>controls</code>	If TRUE, render the previous / next control buttons.
<code>fade</code>	If TRUE, crossfade between slides instead of sliding (<code>.carousel-fade</code>).
<code>autoplay</code>	TRUE (default) starts cycling on load (<code>data-bs-ride="carousel"</code>); FALSE omits the attribute so the carousel only ever advances on user interaction; "resume" is Bootstrap's <code>data-bs-ride="true"</code> , which starts cycling after the first interaction.
<code>interval</code>	Cycling interval in milliseconds (sets <code>data-bs-interval</code>).
<code>wrap</code>	If FALSE, stop at the last slide instead of cycling round.
<code>keyboard</code>	If FALSE, ignore the arrow keys.
<code>pause</code>	"hover" (Bootstrap's default) pauses on mouseover; FALSE never pauses.
<code>touch</code>	If FALSE, disable swipe gestures on touch screens.
<code>dark</code>	If TRUE, the dark variant via <code>data-bs-theme="dark"</code> (the Bootstrap 5.3 idiom; <code>.carousel-dark</code> is deprecated).
<code>class</code>	Extra classes.

active	If TRUE, this item is shown first. Exactly one item per carousel is active; <code>bs_carousel()</code> defaults the first item if none is set.
caption	Optional caption content placed in a <code>.carousel-caption</code> (hidden on small screens, d-none d-md-block).

Value

A carousel tag.

Examples

```
bs_carousel(
  "demo",
  bs_carousel_item(htmltools::img(src = "1.jpg"), active = TRUE),
  bs_carousel_item(htmltools::img(src = "2.jpg"))
)
```

bs_checkbox_group_input

Bootstrap checkbox group

Description

Delegates to `shiny::checkboxGroupInput()` and rewrites the markup to Bootstrap 5 `.form-check` shape so the value stays available as `input$id`.

Usage

```
bs_checkbox_group_input(
  id,
  label = NULL,
  choices,
  selected = NULL,
  ...,
  inline = FALSE,
  reverse = FALSE,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; selected value available as <code>input\$id</code> .
label	Input label.
choices	Named or unnamed vector / list of choices.
selected	Initially selected value(s).

...	Extra attributes applied to each radio <input>.
inline	If TRUE, lay the choices out horizontally.
reverse	If TRUE, put the label before the control (.form-check-reverse, Bootstrap 5.2).
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::checkboxGroupInput\(\)](#)

Examples

```
bs_checkbox_group_input("opts", "Options", c("A", "B", "C"), selected = "A")
```

bs_checkbox_input	<i>Bootstrap checkbox input</i>
-------------------	---------------------------------

Description

Bootstrap checkbox input

Usage

```
bs_checkbox_input(
  id,
  label = NULL,
  value = FALSE,
  ...,
  switch = FALSE,
  reverse = FALSE,
  help = NULL,
  width = NULL
)

bs_switch_input(
  id,
  label = NULL,
  value = FALSE,
  ...,
  reverse = FALSE,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial checked state.
...	Extra attributes applied to the <input> element.
switch	If TRUE, render as a toggle switch (<code>.form-switch</code>).
reverse	If TRUE, put the label before the control (<code>.form-check-reverse</code> , Bootstrap 5.2).
help	Help text rendered below the control (<code>.form-text</code>).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::checkboxInput\(\)](#)

Examples

```
bs_checkbox_input("agree", "I agree", TRUE)
bs_switch_input("dark", "Dark mode")
bs_checkbox_input("dark", "Dark mode", switch = TRUE)
```

bs_close_button	<i>Bootstrap close button</i>
-----------------	-------------------------------

Description

Bootstrap close button

Usage

```
bs_close_button(..., label = "Close", white = FALSE, class = NULL)
```

Arguments

...	Named HTML attributes (e.g. <code>data-bs-dismiss</code>).
label	Accessible label.
white	If TRUE, the variant for dark backgrounds (<code>data-bs-theme="dark"</code> on the button — the Bootstrap 5.3 idiom; <code>.btn-close-white</code> is deprecated).
class	Extra classes.

Value

A button tag.

Examples

```
bs_close_button()
bs_close_button(`data-bs-dismiss` = "alert", white = TRUE)
```

bs_col	<i>Bootstrap grid column</i>
--------	------------------------------

Description

Bootstrap grid column

Usage

```
bs_col(
  ...,
  width = NULL,
  sm = NULL,
  md = NULL,
  lg = NULL,
  xl = NULL,
  xxl = NULL,
  offset = NULL,
  order = NULL,
  align_self = NULL,
  class = NULL
)
```

Arguments

...	Content, and named HTML attributes.
width	Base column span: an integer 1-12, "auto", or TRUE for an equal-width column. NULL (default) yields a bare .col.
sm, md, lg, xl, xxl	Per-breakpoint spans (integer, "auto" or TRUE).
offset	Column offset. A single value applies at the base breakpoint; a named list sets per-breakpoint offsets (e.g. list(md = 2)).
order	Column order (order-*): integer 0-5, "first" or "last". A named list sets per-breakpoint order.
align_self	Vertical self-alignment: "start", "center", "end".
class	Extra classes.

Value

A column tag.

Examples

```
bs_col(width = 6, md = 4, "content")
```

bs_collapse	<i>Bootstrap collapse</i>
-------------	---------------------------

Description

A toggleable container that shows or hides content. Pair it with a `bs_collapse_trigger()` (declarative, no server round trip) and/or drive it from the server with `update_bs_collapse()`. Its shown/hidden state is reported to the server as `input$id` (TRUE when visible).

Usage

```
bs_collapse(id, ..., open = FALSE, horizontal = FALSE, class = NULL)
```

Arguments

<code>id</code>	Collapse id; visibility state available as <code>input\$id</code> .
<code>...</code>	Collapse content and named HTML attributes.
<code>open</code>	If TRUE, the collapse is visible initially (<code>.show</code>).
<code>horizontal</code>	If TRUE, collapse transitions width instead of height (<code>.collapse-horizontal</code>).
<code>class</code>	Extra classes.

Value

A collapse tag.

Examples

```
bs_collapse("more", "Hidden content revealed on toggle.")
```

bs_collapse_trigger	<i>Trigger a collapse from the UI</i>
---------------------	---------------------------------------

Description

Renders a control that toggles the `bs_collapse()` whose id matches target, using Bootstrap's declarative `data-bs-toggle="collapse"` attributes (no server round trip required).

Usage

```
bs_collapse_trigger(target, ..., button = TRUE, expanded = FALSE, class = NULL)
```

Arguments

target	Id of the <code>bs_collapse()</code> to toggle.
...	Content (label) and named HTML attributes.
button	If TRUE (default), render a <code><button class="btn"></code> ; otherwise render an <code></code> .
expanded	Initial expanded state of the trigger. Set to TRUE when the target is a <code>bs_collapse(open = TRUE)</code> so the initial <code>aria-expanded / .collapsed</code> state is correct.
class	Extra classes.

Value

A button (or anchor) tag.

Examples

```
bs_collapse_trigger("more", "Toggle")
```

bs_color_input	<i>Bootstrap colour input</i>
----------------	-------------------------------

Description

A native `<input type="color" class="form-control form-control-color">` whose value (a `"#rrggbb"` string) is reported as `input$id`. Drive it server-side with `update_bs_color()`.

Usage

```
bs_color_input(
  id,
  label = NULL,
  value = "#000000",
  ...,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial colour as a hex string (e.g. "#0d6efd").
...	Extra attributes applied to the <input> element.
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

Examples

```
bs_color_input("col", "Pick a colour", value = "#0d6efd")
```

bs_container	<i>Bootstrap container</i>
--------------	----------------------------

Description

Bootstrap container

Usage

```
bs_container(..., fluid = FALSE, breakpoint = NULL, class = NULL)
```

Arguments

...	Content, and named HTML attributes.
fluid	If TRUE, a full-width .container-fluid. Ignored when breakpoint is set.
breakpoint	One of "sm", "md", "lg", "xl", "xxl" for a responsive .container-{breakpoint}.
class	Extra classes.

Value

A container tag.

See Also

```
bs\_row\(\), bs\_col\(\)
```

Examples

```
bs_container(bs_row(bs_col("a"), bs_col("b")))
```

bs_date_input	<i>Bootstrap date input</i>
---------------	-----------------------------

Description

A native `<input type="date">` with the Bootstrap `.form-control` class, as the Bootstrap 5.3 forms reference has it: the browser supplies the calendar.

Usage

```
bs_date_input(
  id,
  label = NULL,
  value = NULL,
  ...,
  min = NULL,
  max = NULL,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

<code>id</code>	Input id; selected value available as <code>input\$id</code> .
<code>label</code>	Input label.
<code>value</code>	Initial date (a <code>Date</code> or <code>"yyyy-mm-dd"</code> string), or <code>NULL</code> for an empty field.
<code>...</code>	Extra attributes applied to each radio <code><input></code> .
<code>min, max</code>	Earliest / latest selectable date.
<code>size</code>	Control size: <code>"sm"</code> or <code>"lg"</code> .
<code>help</code>	Help text rendered below the control (<code>.form-text</code>).
<code>width</code>	CSS width (e.g. <code>"100%"</code> , <code>"200px"</code>).

Details

This does **not** delegate to `shiny::dateInput()`, which loads `bootstrap-datepicker` – a third-party stylesheet whose calendar markup (`.datepicker`, `.datepicker-days`, ...) appears nowhere in the Bootstrap documentation and which a designer's SASS sheet cannot reach. The consequence is that `format`, `language`, `weekstart` and `datesdisabled` are gone: the browser owns the presentation, and there is no Bootstrap way to ask it for another. Use `update_bs_date_input()` rather than `shiny::updateDateInput()`.

Value

A form control tag.

See Also

[update_bs_date_input\(\)](#), [bs_date_range_input\(\)](#)

Examples

```
bs_date_input("day", "Pick a day", value = "2026-06-26")
```

bs_date_range_input	<i>Bootstrap date range input</i>
---------------------	-----------------------------------

Description

Two native date fields joined in an `.input-group`, with a separator between them. Bootstrap has no date-range component, so this is the assembly its docs prescribe; see [bs_date_input\(\)](#) for why neither field delegates to shiny.

Usage

```
bs_date_range_input(
  id,
  label = NULL,
  start = NULL,
  end = NULL,
  ...,
  min = NULL,
  max = NULL,
  separator = "to",
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

<code>id</code>	Input id; selected value available as <code>input\$id</code> .
<code>label</code>	Input label.
<code>start, end</code>	Initial start / end dates.
<code>...</code>	Extra attributes applied to each radio <code><input></code> .
<code>min, max</code>	Earliest / latest selectable date.
<code>separator</code>	Text shown between the two fields, in an <code>.input-group-text</code> .
<code>size</code>	Control size: "sm" or "lg".
<code>help</code>	Help text rendered below the control (<code>.form-text</code>).
<code>width</code>	CSS width (e.g. "100%", "200px").

Value

A form control tag. input\$id is a length-2 Date.

See Also

[update_bs_date_range_input\(\)](#), [bs_date_input\(\)](#)

Examples

```
bs_date_range_input("range", "Period", start = "2026-01-01")
```

bs_display_heading	<i>Bootstrap display heading</i>
--------------------	----------------------------------

Description

A larger, slightly more opinionated heading style (`.display-*`).

Usage

```
bs_display_heading(..., level = 1, class = NULL)
```

Arguments

<code>...</code>	Heading content and named HTML attributes.
<code>level</code>	Display size / heading level (1-6).
<code>class</code>	Extra classes.

Value

A heading tag.

Examples

```
bs_display_heading("Display heading", level = 2)
```

bs_download_button *Bootstrap download button*

Description

Delegates to `shiny::downloadButton()` and adapts its markup to Bootstrap 5.

Usage

```
bs_download_button(
  id,
  label = "Download",
  ...,
  icon = NULL,
  color = "primary",
  outline = FALSE,
  size = NULL,
  class = NULL
)
```

Arguments

id	Output id, matching the <code>shiny::downloadHandler()</code> assigned to <code>output\$id</code> .
label	Button label (text or tags).
...	Additional content and named HTML attributes, forwarded to the anchor.
icon	Optional icon tag placed before the label. <code>NULL</code> (the default) renders no icon.
color	One of the eight Bootstrap theme colours, or "link".
outline	If <code>TRUE</code> , an outline button (<code>.btn-outline-*</code>).
size	"sm" or "lg".
class	Extra classes.

Details

shiny hardcodes `class = "btn btn-default shiny-download-link"`. `.btn-default` is a Bootstrap 3 class with no Bootstrap 5 equivalent, so the button renders with the `.btn` box but no colour, border or hover state. We drop it and substitute a real 5.3 variant (`.btn-primary`, `.btn-outline-secondary`, ...), matching `bs_button()`'s API.

Everything else is left to shiny: the `shiny-download-link` class, the `href` / `target` / `download` attributes and the auto-enable dance (the anchor ships `.disabled` + `aria-disabled` and shiny's client enables it once the matching `shiny::downloadHandler()` is registered) are untouched, so `output$id <- downloadHandler(...)` works exactly as usual.

Unlike shiny, `icon` defaults to `NULL`: shiny's default is a Font Awesome icon, which is outside Bootstrap 5 and therefore outside this package.

Value

An anchor tag styled as a Bootstrap button.

See Also

`shiny::downloadHandler()`, `bs_download_link()`, `bs_button()`

Examples

```
bs_download_button("report", "Download CSV")
bs_download_button("report", "Export", color = "secondary", outline = TRUE)
```

bs_download_link	<i>Bootstrap download link</i>
------------------	--------------------------------

Description

Delegates to `shiny::downloadLink()`, the plain-text counterpart of `bs_download_button()`. shiny's markup is already Bootstrap-5 clean here (no `.btn-default`), so this only layers on the 5.3 link utilities: `color` emits a `.link-*` colour class.

Usage

```
bs_download_link(id, label = "Download", ..., color = NULL, class = NULL)
```

Arguments

<code>id</code>	Output id, matching the <code>shiny::downloadHandler()</code> assigned to <code>output\$id</code> .
<code>label</code>	Button label (text or tags).
<code>...</code>	Additional content and named HTML attributes, forwarded to the anchor.
<code>color</code>	Optional theme colour, rendered as <code>.link-*</code> .
<code>class</code>	Extra classes.

Value

An anchor tag.

See Also

`shiny::downloadHandler()`, `bs_download_button()`

Examples

```
bs_download_link("report", "Download the raw data")
bs_download_link("report", "Export", color = "danger")
```

`bs_dropdown`*Bootstrap dropdown*

Description

A toggleable menu of links, headers and dividers. Compose the menu with `bs_dropdown_item()`, `bs_dropdown_divider()`, `bs_dropdown_header()` and `bs_dropdown_text()`. Bootstrap drives the toggle; give an item an id to wire it as a Shiny action button (`input$id`).

Usage

```
bs_dropdown(  
  label,  
  ...,  
  id = NULL,  
  color = "secondary",  
  outline = FALSE,  
  size = NULL,  
  split = FALSE,  
  direction = "down",  
  dark = FALSE,  
  align = NULL,  
  auto_close = TRUE,  
  offset = NULL,  
  reference = NULL,  
  class = NULL  
)  
  
bs_dropdown_item(  
  ...,  
  id = NULL,  
  href = "#",  
  active = FALSE,  
  disabled = FALSE,  
  class = NULL  
)  
  
bs_dropdown_divider(class = NULL)  
  
bs_dropdown_header(..., level = 6, class = NULL)  
  
bs_dropdown_text(..., class = NULL)
```

Arguments

`label` Toggle button label (text or tags).

...	Menu contents (dropdown items, dividers, headers, text) and named HTML attributes (forwarded to the wrapper).
id	Optional input id. When set, the item becomes a Shiny action button and its click count is reported as <code>input\$id</code> .
color	Toggle button theme colour, one of the Bootstrap theme colours or "link".
outline	If TRUE, an outline toggle button (<code>.btn-outline-*</code>).
size	Toggle button size: "sm" or "lg".
split	If TRUE, render a split button (a normal action button plus a separate toggle caret).
direction	Menu drop direction: "down", "up", "end" or "start".
dark	If TRUE, a dark dropdown via <code>data-bs-theme="dark"</code> on the wrapper (the Bootstrap 5.3 idiom; <code>.dropdown-menu-dark</code> is deprecated).
align	Menu alignment. "end" right-aligns the menu (<code>.dropdown-menu-end</code>); a named list such as <code>list(lg = "end")</code> produces a responsive alignment (<code>.dropdown-menu-lg-end</code>).
auto_close	When the menu closes on its own: TRUE (the default, inside or outside), "inside", "outside", or FALSE for manual only.
offset	Menu offset from its toggle, as "x,y" in pixels.
reference	What the menu is positioned against: "toggle" (the default), "parent", or a CSS selector. Bootstrap reads all three from the <i>toggle</i> , not from the wrapper, which is why they are arguments rather than something to pass through ...
class	Extra classes for the wrapper.
href	Link target.
active	If TRUE, mark the item active (<code>.active</code>).
disabled	If TRUE, mark the item disabled (<code>.disabled</code>).
level	Heading level (1-6) for the dropdown header.

Details

Give the dropdown itself an id and its open state is reported as `input$id` (TRUE while the menu is open), and it can be driven from the server with [show_bs_dropdown\(\)](#), [hide_bs_dropdown\(\)](#) and [toggle_bs_dropdown\(\)](#).

Value

A dropdown tag.

See Also

[show_bs_dropdown\(\)](#), [bs_nav_dropdown\(\)](#)

Examples

```
bs_dropdown("Menu", bs_dropdown_item("Action", id = "act"))
bs_dropdown(
  "Menu",
  bs_dropdown_header("Actions"),
  bs_dropdown_item("Edit", id = "edit"),
  bs_dropdown_divider(),
  bs_dropdown_text("Signed in as Colin")
)
```

bs_feedback

Attach validation feedback to a form control

Description

Bootstrap only shows a `.valid-feedback` / `.invalid-feedback` message when it is a *following sibling* of the control carrying `.is-valid` / `.is-invalid`. A feedback div placed after a `bs_*_input()` is a sibling of shiny's input container, not of the control inside it, so it would never display. `bs_feedback()` inserts the messages in the right place, whatever the control's internal structure, and is the supported way to validate a bootstrap input.

Usage

```
bs_feedback(input, valid = NULL, invalid = NULL, state = NULL)
```

Arguments

input	A control tag tree produced by a <code>bs_*_input()</code> constructor.
valid	Message shown when the control is valid, or NULL for none.
invalid	Message shown when the control is invalid, or NULL.
state	Initial state: "valid", "invalid" or NULL (neutral).

Details

Declare the messages in the UI and switch the state from the server with [set_bs_validation\(\)](#).

Value

The input, with the feedback messages attached.

See Also

[set_bs_validation\(\)](#), [bs_valid_feedback\(\)](#)

Examples

```
bs_feedback(  
  bs_text_input("user", "Username"),  
  invalid = "Please choose a username."  
)
```

bs_figure	<i>Bootstrap figure</i>
-----------	-------------------------

Description

Group an image with a caption. Compose `bs_figure()` with `bs_figure_img()` and `bs_figure_caption()`.

Usage

```
bs_figure(..., class = NULL)  
  
bs_figure_img(src, ..., alt = "", class = NULL)  
  
bs_figure_caption(..., class = NULL)
```

Arguments

<code>...</code>	Figure content (image, caption) and named HTML attributes.
<code>class</code>	Extra classes.
<code>src</code>	Image source URL.
<code>alt</code>	Alternative text. Defaults to "", which marks the image decorative; an <code></code> with no alt at all is announced by its file name instead.

Value

A figure tag.

Examples

```
bs_figure(  
  bs_figure_img("photo.jpg", alt = "A photo"),  
  bs_figure_caption("A caption for the image.")  
)
```

bs_file_input

*Bootstrap file input***Description**

Delegates to `shiny::fileInput()` for the upload plumbing, then emits the Bootstrap 5.3 markup: a plain `<input class="form-control" type="file">`.

Usage

```
bs_file_input(
  id,
  label = NULL,
  ...,
  multiple = FALSE,
  accept = NULL,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

<code>id</code>	Input id; selected value available as <code>input\$id</code> .
<code>label</code>	Input label.
<code>...</code>	Extra attributes applied to the <code><input type="file"></code> element (e.g. <code>capture</code> , <code>webkitdirectory</code>).
<code>multiple</code>	Allow selecting more than one file.
<code>accept</code>	Character vector of accepted MIME types / extensions.
<code>size</code>	Control size: "sm" or "lg".
<code>help</code>	Help text rendered below the control (<code>.form-text</code>).
<code>width</code>	CSS width (e.g. "100%", "200px").

Details

shiny builds the Bootstrap 3 compound widget instead – a "Browse" button next to a readonly text box showing the file name, with the real input hidden off-screen – which is not in the Bootstrap 5.3 docs at all. Only shiny's own element is kept, so uploads, the progress bar and `input$id` are unchanged; the browser draws the button and the file name itself, which is what the 5.3 reference relies on.

Value

A form control tag.

See Also

`shiny::fileInput()`

Examples

```
bs_file_input("upload", "Upload a file", accept = ".csv")
```

bs_floating_label	<i>Bootstrap floating label</i>
-------------------	---------------------------------

Description

Reshape a control built by one of the `bs*_input()` constructors (e.g. `bs_text_input()`, `bs_select_input()`) into a Bootstrap 5 floating-label group, where the label animates into the control's border. The existing control and its Shiny input wiring are preserved (the element is moved, not rebuilt), so the value still reports as `input$id`.

Usage

```
bs_floating_label(input, label = NULL, class = NULL)
```

Arguments

<code>input</code>	A control tag tree produced by a <code>bs*_input()</code> constructor.
<code>label</code>	Floating label text. If <code>NULL</code> , the control's existing label text is reused.
<code>class</code>	Extra classes.

Value

A `.form-floating` wrapper tag.

Examples

```
bs_floating_label(bs_text_input("email", "Email address"))
```

bs_form	<i>Bootstrap form element</i>
---------	-------------------------------

Description

A plain <form> container for grouping form controls.

Usage

```
bs_form(..., novalidate = FALSE, class = NULL)
```

Arguments

- ... Form content (controls, layout, buttons) and named HTML attributes.
- novalidate If TRUE, add the novalidate attribute to disable the browser’s native validation UI (useful with custom validation feedback).
- class Extra classes.

Value

A form tag.

Examples

```
bs_form(  
  bs_text_input("email", "Email"),  
  bs_button("Submit", color = "primary")  
)
```

bs_form_label	<i>Bootstrap form label</i>
---------------	-----------------------------

Description

A <label class="form-label"> tied to a control via its for attribute.

Usage

```
bs_form_label(for_id, ..., class = NULL)
```

Arguments

- for_id The id of the control this label describes (rendered as the for attribute).
- ... Label content and named HTML attributes.
- class Extra classes.

Value

A label tag.

Examples

```
bs_form_label("email", "Email address")
```

bs_form_text	<i>Bootstrap form help text</i>
--------------	---------------------------------

Description

Muted helper text rendered below a control (`.form-text`).

Usage

```
bs_form_text(..., class = NULL)
```

Arguments

<code>...</code>	Help content and named HTML attributes.
<code>class</code>	Extra classes.

Value

A div tag.

Examples

```
bs_form_text("Must be 8-20 characters long.")
```

bs_hstack	<i>Bootstrap stacks (horizontal / vertical flex layouts)</i>
-----------	--

Description

Shorthand flex helpers added in Bootstrap 5.1. `bs_hstack()` lays children out in a row, `bs_vstack()` in a column.

Usage

```
bs_hstack(..., gap = NULL, class = NULL)
```

```
bs_vstack(..., gap = NULL, class = NULL)
```

Arguments

<code>...</code>	Content, and named HTML attributes.
<code>gap</code>	Spacing between items (<code>gap=*</code>), an integer 0-5.
<code>class</code>	Extra classes.

Value

A stack tag.

Examples

```
bs_hstack(bs_button(label = "A"), bs_button(label = "B"), gap = 2)
bs_vstack(bs_alert("one"), bs_alert("two"), gap = 3)
```

bs_icon_link	<i>Bootstrap icon link</i>
--------------	----------------------------

Description

The Bootstrap 5.3 icon-link helper: an inline flex `<a>` that pairs an icon (e.g. an SVG) with text, with an optional hover translation of the icon (`.icon-link-hover`).

Usage

```
bs_icon_link(..., href = "#", hover = FALSE, class = NULL)
```

Arguments

<code>...</code>	Link content (icon and text) and named HTML attributes.
<code>href</code>	Link target.
<code>hover</code>	If TRUE, translate the icon on hover (<code>.icon-link-hover</code>).
<code>class</code>	Extra classes.

Value

An anchor tag.

Examples

```
bs_icon_link("Icon link", href = "#")
```

bs_img*Bootstrap image*

Description

Bootstrap image

Usage

```
bs_img(  
  src,  
  ...,  
  fluid = FALSE,  
  thumbnail = FALSE,  
  rounded = FALSE,  
  object_fit = NULL,  
  alt = "",  
  class = NULL  
)
```

Arguments

src	Image source URL.
...	Extra named HTML attributes applied to the .
fluid	Make the image responsive (.img-fluid).
thumbnail	Render with a rounded thumbnail border (.img-thumbnail).
rounded	Add rounded corners (.rounded).
object_fit	How the image fills its box (Bootstrap 5.3 .object-fit-* utility): "contain", "cover", "fill", "scale" (scale-down) or "none".
alt	Alternative text. Defaults to "", which marks the image decorative; an with no alt at all is announced by its file name instead.
class	Extra classes.

Value

An image tag.

Examples

```
bs_img("logo.png", alt = "Logo", fluid = TRUE)
```

bs_input_group	<i>Bootstrap input group</i>
----------------	------------------------------

Description

Groups one or more form controls together with add-ons (text, buttons, dropdowns) on a single line.

Usage

```
bs_input_group(..., size = NULL, class = NULL)
```

```
bs_input_group_text(..., class = NULL)
```

Arguments

...	Controls and add-ons (e.g. bs_input_group_text() , bs_text_input() , bs_button()) and named HTML attributes.
size	Control size: "sm" or "lg".
class	Extra classes.

Details

When a full `bs*_input()` is passed, only its bare control is kept so it sits flush with the add-ons: the input's label and help text are dropped (compose them outside the group). Inputs whose Shiny binding lives on their container — [bs_date_input\(\)](#), [bs_date_range_input\(\)](#), [bs_file_input\(\)](#), [bs_radio_input\(\)](#), [bs_checkbox_group_input\(\)](#) — cannot be unwrapped this way and raise an error.

Value

An input-group tag.

Examples

```
bs_input_group(  
  bs_input_group_text("@"),  
  bs_text_input("user", placeholder = "Username")  
)
```

bs_lead	<i>Bootstrap lead paragraph</i>
---------	---------------------------------

Description

A standout opening paragraph (`.lead`).

Usage

```
bs_lead(..., class = NULL)
```

Arguments

<code>...</code>	Paragraph content and named HTML attributes.
<code>class</code>	Extra classes.

Value

A paragraph tag.

Examples

```
bs_lead("This is a lead paragraph.")
```

bs_list_group	<i>Bootstrap list group</i>
---------------	-----------------------------

Description

A flexible container for displaying a series of content. Compose with [bs_list_group_item\(\)](#). When `id` is supplied the group becomes selectable: clicking an action item (`action = TRUE` or `href`) reports that item's value (its `data-value`) as `input$id`, and the selection can be driven server-side with [update_bs_list_group\(\)](#).

Usage

```
bs_list_group(  
  id = NULL,  
  ...,  
  flush = FALSE,  
  numbered = FALSE,  
  horizontal = FALSE,  
  class = NULL  
)  
  
bs_list_group_item(  
  ...  
)
```

```

    ...,
    value = NULL,
    active = FALSE,
    disabled = FALSE,
    color = NULL,
    action = FALSE,
    href = NULL,
    class = NULL
  )

```

Arguments

<code>id</code>	Optional list group id. When set, the active item's value is reported as <code>input\$id</code> and the group is wired for selection.
<code>...</code>	List items built with <code>bs_list_group_item()</code> , plus named HTML attributes.
<code>flush</code>	If TRUE, render edge-to-edge without an outer border and rounded corners (<code>.list-group-flush</code>).
<code>numbered</code>	If TRUE, render a numbered list (<code><ol class="list-group list-group-numbered"></code>).
<code>horizontal</code>	Lay items out horizontally. TRUE for <code>.list-group-horizontal</code> , or a break-point string ("sm", "md", "lg", "xl", "xxl") for <code>.list-group-horizontal-{bp}</code> .
<code>class</code>	Extra classes.
<code>value</code>	Item value reported to the server when selected (its <code>data-value</code>). Only meaningful in a selectable group (when the parent <code>bs_list_group()</code> has an <code>id</code>).
<code>active</code>	If TRUE, mark the item as the active/selected one.
<code>disabled</code>	If TRUE, mark the item disabled.
<code>color</code>	Contextual theme colour (<code>.list-group-item-*</code>).
<code>action</code>	If TRUE, render an actionable <code><button></code> item (<code>.list-group-item-action</code>). Ignored when <code>href</code> is supplied (which always renders an actionable <code><a></code>).
<code>href</code>	Link target; renders the item as an <code><a></code> (always actionable).

Value

A list group tag.

Examples

```

bs_list_group(
  bs_list_group_item("An item"),
  bs_list_group_item("A second item", active = TRUE)
)
bs_list_group_item("A link item", href = "#", value = "a")

```

bs_list_unstyled	<i>Bootstrap unstyled / inline lists</i>
------------------	--

Description

Remove a list's default styling (`.list-unstyled`) or lay items out inline (`.list-inline`). Each unnamed ... argument becomes one `<li>`; named arguments become attributes of the `<ul>`.

Usage

```
bs_list_unstyled(..., class = NULL)
```

```
bs_list_inline(..., class = NULL)
```

Arguments

...	List items (unnamed) and named HTML attributes for the <code></code> .
class	Extra classes.

Value

An unordered list tag.

Examples

```
bs_list_unstyled("First", "Second", "Third")
bs_list_inline("One", "Two", "Three")
```

bs_modal	<i>Bootstrap modal</i>
----------	------------------------

Description

A faithful Bootstrap 5 modal dialog. The modal's open/closed state is reported to the server as `input$id` (TRUE when shown), and can be driven server-side with [show_bs_modal\(\)](#), [hide_bs_modal\(\)](#) and [toggle_bs_modal\(\)](#).

Usage

```
bs_modal(
  id,
  ...,
  title = NULL,
  footer = NULL,
  size = NULL,
  centered = FALSE,
```

```

    scrollable = FALSE,
    fullscreen = FALSE,
    backdrop = TRUE,
    keyboard = TRUE,
    class = NULL
)

bs_modal_header(..., class = NULL)

bs_modal_title(..., level = 1, class = NULL)

bs_modal_body(..., class = NULL)

bs_modal_footer(..., class = NULL)

```

Arguments

<code>id</code>	Modal id; open state available as <code>input\$id</code> .
<code>...</code>	Modal body content and named HTML attributes applied to the root.
<code>title</code>	Optional header title. When supplied, a <code>.modal-header</code> with a title and a close button is rendered.
<code>footer</code>	Optional footer content (e.g. buttons), placed in a <code>.modal-footer</code> .
<code>size</code>	Dialog size: "sm", "lg" or "xl".
<code>centered</code>	If TRUE, vertically centre the dialog (<code>.modal-dialog-centered</code>).
<code>scrollable</code>	If TRUE, scroll long bodies inside the dialog (<code>.modal-dialog-scrollable</code>).
<code>fullscreen</code>	TRUE for an always-fullscreen modal, or a breakpoint ("sm"/"md"/"lg"/"xl"/"xxl") for fullscreen below that breakpoint (<code>.modal-fullscreen-{bp}-down</code>).
<code>backdrop</code>	Backdrop behaviour: TRUE (backdrop shown, dismiss on outside click), "static" (backdrop shown, do not dismiss on outside click) or FALSE (no backdrop at all).
<code>keyboard</code>	If FALSE, the modal cannot be closed with the Escape key.
<code>class</code>	Extra classes.
<code>level</code>	Heading level (1-6) for the modal title.

Details

For full control over the dialog structure (in place of the `title` and `footer` shortcuts) compose the body yourself with `bs_modal_header()`, `bs_modal_body()` and `bs_modal_footer()`.

Value

A modal tag.

Examples

```
bs_modal("info", "Modal body text.", title = "Heads up")

# Or compose the dialog yourself.
bs_modal(
  "info",
  bs_modal_header(bs_modal_title("Heads up")),
  bs_modal_body("Modal body text."),
  bs_modal_footer(bs_button("ok", "OK", color = "primary"))
)
```

bs_modal_trigger	<i>Trigger a modal from the UI</i>
------------------	------------------------------------

Description

Renders a button that opens the modal whose id matches `target`, using Bootstrap's declarative `data-bs-toggle="modal"` attributes (no server round trip required).

Usage

```
bs_modal_trigger(target, ..., class = NULL)
```

Arguments

<code>target</code>	Id of the <code>bs_modal()</code> to open.
<code>...</code>	Content (label) and named HTML attributes, forwarded to <code>bs_button()</code> .
<code>class</code>	Extra classes.

Value

A button tag.

Examples

```
bs_modal_trigger("info", "Open modal", color = "primary")
```

bs_nav	<i>Bootstrap navigation list</i>
--------	----------------------------------

Description

A navigation container rendered as a Bootstrap `<ul class="nav">`. Compose it with `bs_nav_item()` and `bs_nav_link()`. For tabs that also switch panels use `bs_tabset()` instead.

Usage

```
bs_nav(
  ...,
  id = NULL,
  type = NULL,
  fill = FALSE,
  justified = FALSE,
  vertical = FALSE,
  class = NULL
)

bs_nav_item(..., class = NULL)

bs_nav_link(
  ...,
  href = "#",
  active = FALSE,
  disabled = FALSE,
  id = NULL,
  value = NULL,
  class = NULL
)
```

Arguments

<code>...</code>	Navigation items (<code>bs_nav_item()</code> / <code>bs_nav_link()</code>) and named HTML attributes.
<code>id</code>	Optional element id.
<code>type</code>	Visual style: "tabs", "pills" or "underline" (Bootstrap 5.3) — default NULL for a plain nav).
<code>fill</code>	If TRUE, items expand to fill available width (<code>.nav-fill</code>).
<code>justified</code>	If TRUE, items get equal width (<code>.nav-justified</code>).
<code>vertical</code>	If TRUE, stack items vertically (<code>.flex-column</code>).
<code>class</code>	Extra classes.
<code>href</code>	Link target.
<code>active</code>	If TRUE, mark the link as the active page (adds <code>.active</code> and <code>aria-current="page"</code>).

disabled	If TRUE, mark the link disabled (.disabled).
value	Value reported as input\$id by the enclosing bs_nav() when this link is active. Defaults to the link's text.

Details

Given an id, the nav reports the value of the active link as input\$id and is driven from the server with [update_bs_nav\(\)](#). Clicking a link makes it active without leaving the page, so a nav becomes a plain selector; give each link a value.

Value

A nav tag.

See Also

[update_bs_nav\(\)](#), [bs_tabset\(\)](#)

Examples

```
bs_nav(
  bs_nav_item(bs_nav_link("Active", active = TRUE)),
  bs_nav_item(bs_nav_link("Link")),
  type = "tabs"
)
```

bs_nav_dropdown

Bootstrap dropdown inside a nav or a navbar

Description

The dropdown a designer draws in a navbar: an `<li class="nav-item dropdown">` whose toggle is a `.nav-link`, not a button. [bs_dropdown\(\)](#) builds the standalone, button-triggered menu, which is invalid as a direct child of the `<ul class="navbar-nav">` that [bs_navbar_nav\(\)](#) and [bs_nav\(\)](#) produce, and renders as a grey button rather than a nav link.

Usage

```
bs_nav_dropdown(
  label,
  ...,
  active = FALSE,
  disabled = FALSE,
  align = NULL,
  dark = FALSE,
  id = NULL,
  class = NULL
)
```

Arguments

label	Toggle text.
...	Menu content (unnamed) and named HTML attributes applied to the .
active	If TRUE, mark the toggle as the active page (.active and aria-current="page").
disabled	If TRUE, mark the toggle disabled.
align	Menu alignment: "end", or a named list per breakpoint (e.g. list(lg = "end")).
dark	If TRUE, render a dark menu (data-bs-theme="dark", the Bootstrap 5.3 idiom).
id	Optional dropdown id. Its open state is reported as input\$id, and it can be driven with show_bs_dropdown() and friends.
class	Extra classes for the .

Details

Fill it with the same items as [bs_dropdown\(\)](#): [bs_dropdown_item\(\)](#), [bs_dropdown_header\(\)](#), [bs_dropdown_divider\(\)](#), [bs_dropdown_text\(\)](#).

Value

An tag, ready to drop into [bs_navbar_nav\(\)](#) or [bs_nav\(\)](#).

See Also

[bs_dropdown\(\)](#), [bs_navbar_nav\(\)](#), [bs_nav\(\)](#)

Examples

```
bs_navbar_nav(  
  bs_nav_item(bs_nav_link("Home", active = TRUE)),  
  bs_nav_dropdown("More", bs_dropdown_item("Settings"))  
)
```

bs_navbar	<i>Bootstrap navbar</i>
-----------	-------------------------

Description

A responsive navigation header. Compose with [bs_navbar_brand\(\)](#), [bs_navbar_nav\(\)](#) (containing [bs_nav_item\(\)](#) / [bs_nav_link\(\)](#) from the nav-tabs group), [bs_navbar_text\(\)](#) and, for a menu, [bs_nav_dropdown\(\)](#) (not [bs_dropdown\(\)](#), which builds the standalone button-triggered menu and is invalid inside <ul class="navbar-nav">). The navbar collapses behind a toggler below the expand breakpoint.

Usage

```

bs_navbar(
  ...,
  brand = NULL,
  id = NULL,
  expand = "lg",
  bg = NULL,
  theme = NULL,
  placement = NULL,
  fluid = TRUE,
  class = NULL
)

bs_navbar_brand(..., href = "#", class = NULL)

bs_navbar_nav(
  ...,
  id = NULL,
  scroll = FALSE,
  scroll_height = NULL,
  class = NULL
)

bs_navbar_text(..., class = NULL)

```

Arguments

...	Navbar content (brand, nav lists, text, ...) placed inside the collapsible container, plus named HTML attributes for the <nav>.
brand	Optional brand element, typically bs_navbar_brand() , rendered before the toggler so it stays visible when collapsed.
id	Optional id. When set, the active link's value is reported as input\$id and can be set with update_bs_nav() , exactly as for bs_nav() .
expand	Breakpoint at which the navbar expands: one of "sm", "md", "lg", "xl", "xxl", or TRUE to always expand (.navbar-expand). Use FALSE/NULL to never expand (always collapsed).
bg	Background colour (.bg-*): one of the Bootstrap theme colours, or "body", "body-secondary", "body-tertiary" (the Bootstrap 5.3 navbar default), "white", "black", "transparent".
theme	Colour scheme applied via data-bs-theme (the Bootstrap 5.3 colour-modes idiom; .navbar-light/.navbar-dark are deprecated): "light" or "dark".
placement	Fixed/sticky placement: one of "fixed-top", "fixed-bottom", "sticky-top", "sticky-bottom".
fluid	If TRUE (default) use a full-width .container-fluid, otherwise a fixed-width .container.
class	Extra classes for the <nav>.

href	Link target for the brand.
scroll	If TRUE, enable vertical scrolling within the collapsed navbar nav (. navbar-nav-scroll, Bootstrap 5.1).
scroll_height	Max scroll height (sets --bs-scroll-height), e.g. "75vh" or "200px". Only applies when scroll = TRUE.

Value

A navbar tag.

Examples

```
bs_navbar(brand = bs_navbar_brand("Navbar"), bg = "light", theme = "light")
bs_navbar_brand("Acme", href = "/")
bs_navbar_nav(bs_nav_item(bs_nav_link("Home", active = TRUE)))
bs_navbar_text("Signed in as Mark Otto")
```

bs_notify_toast	<i>Pop a transient toast notification from the server</i>
-----------------	---

Description

Builds a toast on the client and shows it, much like `shiny::showNotification()`. A toast container is created on demand at placement if one is not already present, and the toast removes itself from the DOM once hidden.

Usage

```
bs_notify_toast(
  body,
  ...,
  title = NULL,
  color = NULL,
  autohide = TRUE,
  delay = 5000,
  placement = "top-end",
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

body	Notification body text. Plain text only (it is inserted with <code>textContent</code> client-side, so markup is not interpreted); tags raise an error.
...	Reserved for future extensions; must be empty.
title	Optional header title. Plain text only, like body.
color	Optional theme colour applied as a <code>.text-bg-*</code> background.

autohide	If TRUE (default), hide the toast automatically after delay milliseconds. Use FALSE for a persistent notification the user must dismiss.
delay	Delay in milliseconds before the toast auto-hides.
placement	Container placement (see bs_toast_container()).
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) bs_notify_toast("Saved!", title = "Status", color = "success")
```

bs_numeric_input	<i>Bootstrap numeric input</i>
------------------	--------------------------------

Description

Bootstrap numeric input

Usage

```
bs_numeric_input(
  id,
  label = NULL,
  value = NULL,
  ...,
  min = NULL,
  max = NULL,
  step = NULL,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial value.
...	Extra attributes applied to the <input> element.
min, max, step	Numeric bounds and step.
size	Control size: "sm" or "lg".
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

`shiny::numericInput()`

Examples

```
bs_numeric_input("n", "How many?", value = 1, min = 0, max = 10)
```

bs_offcanvas	<i>Bootstrap offcanvas</i>
--------------	----------------------------

Description

A hidden sidebar panel that slides in from an edge of the viewport. The open/closed state is reported to the server as `input$id` (TRUE when shown), and can be driven server-side with `show_bs_offcanvas()`, `hide_bs_offcanvas()` and `toggle_bs_offcanvas()`. Open it declaratively from the UI with `bs_offcanvas_trigger()`.

Usage

```
bs_offcanvas(  
  id,  
  ...,  
  title = NULL,  
  placement = "start",  
  backdrop = TRUE,  
  scroll = FALSE,  
  responsive = NULL,  
  class = NULL  
)
```

Arguments

id	Offcanvas id; open state available as <code>input\$id</code> .
...	Offcanvas body content and named HTML attributes applied to the root.
title	Optional header title. When supplied, an <code>.offcanvas-header</code> with a title and a close button is rendered.
placement	Edge the panel slides in from: "start" (left), "end" (right), "top" or "bottom".
backdrop	Backdrop behaviour: TRUE (default, dismiss on outside click), FALSE (no backdrop) or "static" (backdrop that does not dismiss on outside click).
scroll	If TRUE, allow body scrolling while the offcanvas is open.

responsive	Optional breakpoint ("sm", "md", "lg", "xl", "xxl"). Below it the panel behaves as an offcanvas; at or above it the content is shown inline (Bootstrap 5.2 responsive offcanvas).
class	Extra classes.

Value

An offcanvas tag.

Examples

```
bs_offcanvas("menu", "Sidebar content.", title = "Menu")
bs_offcanvas("nav", "Shown inline on lg+.", responsive = "lg")
```

bs_offcanvas_trigger *Trigger an offcanvas from the UI*

Description

Renders a button that opens the offcanvas whose id matches target, using Bootstrap's declarative data-bs-toggle="offcanvas" attributes (no server round trip required).

Usage

```
bs_offcanvas_trigger(target, ..., class = NULL)
```

Arguments

target	Id of the bs_offcanvas() to open.
...	Content (label) and named HTML attributes, forwarded to bs_button() .
class	Extra classes.

Value

A button tag.

Examples

```
bs_offcanvas_trigger("menu", "Open menu")
```

Description

Shiny's page constructors wired to the Bootstrap bootstrict vendors, the bootstrict dependency and a theme. Use these as the outermost call of a Shiny UI.

Usage

```
bs_page(
  ...,
  title = NULL,
  theme = bootstrict_theme(),
  color_mode = NULL,
  lang = "en"
)

bs_page_fluid(
  ...,
  title = NULL,
  theme = bootstrict_theme(),
  color_mode = NULL,
  lang = "en"
)

bs_page_fillable(
  ...,
  title = NULL,
  theme = bootstrict_theme(),
  color_mode = NULL,
  lang = "en"
)
```

Arguments

...	UI elements, and named HTML attributes for the page body.
title	Page title (browser tab).
theme	A bootstrict_theme() . Defaults to stock Bootstrap.
color_mode	Initial Bootstrap colour mode: "light", "dark", or "auto" to follow the operating system. A mode the user later chose is remembered in the browser and wins over this initial value. Switch it from the server with set_bs_color_mode() , and read the mode in force as <code>input\$bootstrict_color_mode</code> .
lang	Document language (<html lang>).

Value

A UI definition.

Examples

```
if (interactive()) {
  bs_page(
    theme = bootstrict_theme(primary = "#ff6600"),
    bs_container(bs_card(bs_card_body("Hello")))
  )
  bs_page_fluid(bs_container("Full width", fluid = TRUE))
  bs_page_fillable(bs_container("Fills the viewport"))
}
```

bs_page_item	<i>Bootstrap pagination item</i>
--------------	----------------------------------

Description

A single `<li class="page-item">` carrying a `.page-link` anchor. Use inside `bs_pagination()`.

Usage

```
bs_page_item(
  ...,
  value = NULL,
  href = "#",
  active = FALSE,
  disabled = FALSE,
  class = NULL
)
```

Arguments

- | | |
|----------|---|
| ... | Link content (text or tags) and named HTML attributes applied to the <code></code> . |
| value | Value reported as <code>input\$id</code> by the enclosing <code>bs_pagination()</code> when this item is active. Defaults to the item's text. |
| href | Link target. |
| active | If TRUE, mark as the current page (<code>.active</code> , <code>aria-current="page"</code>). |
| disabled | If TRUE, render as a non-interactive link (<code>.disabled</code> , <code>tabindex="-1"</code> , <code>aria-disabled="true"</code>). |
| class | Extra classes for the <code></code> . |

Value

A `<li>` `page-item` tag.

Examples

```
bs_page_item("1", href = "#", active = TRUE)
```

bs_pagination	<i>Bootstrap pagination</i>
---------------	-----------------------------

Description

A list of page links, built from [bs_page_item\(\)](#)s. For a quick numbered pager use [bs_pagination_numbered\(\)](#).

Usage

```
bs_pagination(  
  ...,  
  id = NULL,  
  size = NULL,  
  align = NULL,  
  label = "Page navigation",  
  class = NULL  
)
```

Arguments

...	Page items built with bs_page_item() , plus named HTML attributes applied to the .
id	Optional pager id. The active item's value is reported as input\$id.
size	Size modifier: "sm" or "lg" (.pagination-sm/.pagination-lg).
align	Horizontal alignment of the pager: "start", "center" or "end" (maps to .justify-content-*).
label	Accessible label for the surrounding <nav> (aria-label).
class	Extra classes for the .

Details

Given an id, the pager reports the value of the active item as input\$id and is driven from the server with [update_bs_pagination\(\)](#). Clicking an item makes it active without leaving the page, so the pager drives the app rather than a URL.

Value

A <nav> pagination tag.

See Also

```
update\_bs\_pagination\(\)
```

Examples

```

bs_pagination(
  bs_page_item("Previous", href = "#"),
  bs_page_item("1", href = "#", active = TRUE),
  bs_page_item("2", href = "#"),
  bs_page_item("Next", href = "#")
)

```

bs_pagination_numbered

Build a numbered pager

Description

Convenience wrapper around [bs_pagination\(\)](#) that lays out a "Previous" control, page numbers 1..n (the current one active) and a "Next" control.

Usage

```

bs_pagination_numbered(
  n,
  current = 1,
  ...,
  id = NULL,
  href_template = NULL,
  size = NULL,
  align = NULL,
  label = "Page navigation",
  class = NULL
)

```

Arguments

n	Total number of pages.
current	Currently active page number (1-based).
...	Additional named HTML attributes forwarded to bs_pagination() 's .
id	Optional pager id. The active page number is reported as input\$id and can be set with update_bs_pagination() .
href_template	Optional sprintf()-style template used to build each page's href from its page number, e.g. "?page=%d". When NULL, every link uses "#".
size	Size modifier passed to bs_pagination() .
align	Alignment passed to bs_pagination() .
label	Accessible label passed to bs_pagination() .
class	Extra classes for the .

Value

A <nav> pagination tag.

Examples

```
bs_pagination_numbered(5, current = 2)
```

bs_password_input	<i>Bootstrap password input</i>
-------------------	---------------------------------

Description

Bootstrap password input

Usage

```
bs_password_input(
  id,
  label = NULL,
  value = "",
  ...,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial value.
...	Extra attributes applied to the <input> element.
size	Control size: "sm" or "lg".
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::passwordInput\(\)](#)

Examples

```
bs_password_input("pw", "Password", help = "At least 8 characters.")
```

bs_placeholder	<i>Bootstrap placeholder</i>
----------------	------------------------------

Description

Loading placeholders ("skeletons") that mimic the shape of content while it loads. Use `bs_placeholder()` for an individual placeholder and wrap one or more in `bs_placeholder_glow()` or `bs_placeholder_wave()` to animate them.

Usage

```
bs_placeholder(..., width = NULL, color = NULL, size = NULL, class = NULL)

bs_placeholder_glow(..., class = NULL)

bs_placeholder_wave(..., class = NULL)
```

Arguments

...	Additional named HTML attributes (and, for the wrappers, child content) applied to the element.
width	Column count (integer 1-12) controlling the placeholder width via the grid (<code>.col-*</code>).
color	Background theme colour (<code>.bg-*</code>), one of the Bootstrap theme colours.
size	Placeholder size, one of "lg", "sm" or "xs"; NULL for the default size.
class	Extra classes.

Value

A placeholder tag.

Examples

```
bs_placeholder_glow(bs_placeholder(width = 6))
bs_placeholder_wave(bs_placeholder(width = 4))
```

bs_popover	<i>Add a Bootstrap popover to a UI element</i>
------------	--

Description

Decorates an existing tag with the data attributes Bootstrap needs to render a popover. Popovers are initialised client-side by bootstrick (Bootstrap does not auto-initialise them). A tag that is already a data-API trigger — a `bs_modal_trigger()`, `bs_offcanvas_trigger()`, `bs_collapse_trigger()` or a dropdown toggle — keeps its own `data-bs-toggle`; the popover works alongside it.

Usage

```
bs_popover(
  tag,
  content,
  ...,
  title = NULL,
  placement = "right",
  trigger = "click",
  html = FALSE
)
```

Arguments

tag	A UI element (a shiny.tag) to attach the popover to.
content	Popover body content (or HTML, when html = TRUE).
...	Extra named attributes applied to tag.
title	Optional popover header.
placement	Popover placement: "auto", "top", "right", "bottom" or "left".
trigger	How the popover is triggered (e.g. "click", "hover", "focus", "manual").
html	If TRUE, allow HTML content in the popover (data-bs-html).

Value

The decorated tag, with the bootstrict dependency attached.

Examples

```
bs_popover(shiny::tags$button("Click me"), "Popover body", title = "Heads up")
```

bs_progress

Bootstrap progress

Description

A progress display. [bs_progress_bar\(\)](#) builds one bar (a .progress track with its .progress-bar); [bs_progress\(\)](#) finalises it, or stacks several bars into a Bootstrap 5.3 .progress-stacked group. Progress is display-only (it reports no value to the server); drive it from the server with [update_bs_progress\(\)](#).

Usage

```
bs_progress(..., height = NULL, class = NULL)

bs_progress_bar(
  value = 0,
  ...,
```

```

    min = 0,
    max = 100,
    color = NULL,
    striped = FALSE,
    animated = FALSE,
    label = NULL,
    aria_label = NULL,
    id = NULL,
    class = NULL
  )

```

Arguments

...	One or more <code>bs_progress_bar()</code> s and named HTML attributes. With two or more bars, the group renders as <code>.progress-stacked</code> .
height	CSS height of the progress track(s) (e.g. <code>"20px"</code> , <code>"1rem"</code>).
class	Extra classes.
value	Current value of the bar.
min, max	Lower and upper bounds of the scale.
color	Bar theme colour (<code>.bg-*</code>), one of the Bootstrap theme colours.
striped	If TRUE, apply the striped variant (<code>.progress-bar-striped</code>).
animated	If TRUE, animate the stripes (<code>.progress-bar-animated</code>).
label	Text shown inside the bar.
aria_label	Accessible name of the progress track (<code>aria-label</code>).
id	Id of the <code>.progress</code> track; required to target it with <code>update_bs_progress()</code> .

Value

A progress tag.

Examples

```

bs_progress(bs_progress_bar(value = 25, id = "load"))
# stacked (Bootstrap 5.3):
bs_progress(
  bs_progress_bar(value = 15, color = "success"),
  bs_progress_bar(value = 30, color = "danger")
)
bs_progress_bar(value = 75, color = "success", striped = TRUE)

```

bs_radio_button_input *Bootstrap toggle button groups*

Description

The segmented control from the Bootstrap "Button group" page: a row of buttons backed by hidden radio or checkbox inputs (`.btn-check`). `bs_radio_button_input()` picks one value, `bs_checkbox_button_input()` picks any number.

Usage

```
bs_radio_button_input(
  id,
  label = NULL,
  choices,
  selected = NULL,
  ...,
  color = "primary",
  outline = TRUE,
  size = NULL,
  vertical = FALSE,
  class = NULL
)

bs_checkbox_button_input(
  id,
  label = NULL,
  choices,
  selected = NULL,
  ...,
  color = "primary",
  outline = TRUE,
  size = NULL,
  vertical = FALSE,
  class = NULL
)
```

Arguments

<code>id</code>	Input id; the selection is available as <code>input\$id</code> (a single string for radio buttons, a character vector — possibly empty — for checkboxes).
<code>label</code>	Label shown above the group, or <code>NULL</code> for none.
<code>choices</code>	Character vector of values. Names, when present, are used as the button labels.
<code>selected</code>	Initially selected value(s). Defaults to the first choice for radio buttons and to none for checkboxes.
<code>...</code>	Named HTML attributes applied to the <code>.btn-group</code> .

color	Button theme colour.
outline	If TRUE (the default, as in the Bootstrap examples), outline buttons (<code>.btn-outline-*</code>).
size	Button size: "sm" or "lg".
vertical	If TRUE, stack the buttons (<code>.btn-group-vertical</code>).
class	Extra classes for the <code>.btn-group</code> .

Details

These are native controls, not restyled shiny inputs, so drive them with `update_bs_toggle_buttons()` rather than `shiny::updateRadioButtons()`.

Value

A form control tag.

See Also

`update_bs_toggle_buttons()`, `bs_button_group()`

Examples

```
bs_radio_button_input("size", "Size", c(Small = "s", Large = "l"))
bs_checkbox_button_input("opts", "Options", c("a", "b"))
```

bs_radio_input	<i>Bootstrap radio button group</i>
----------------	-------------------------------------

Description

Delegates to `shiny::radioButtons()` and rewrites the markup to Bootstrap 5 `.form-check` shape so the value stays available as `input$id`.

Usage

```
bs_radio_input(
  id,
  label = NULL,
  choices,
  selected = NULL,
  ...,
  inline = FALSE,
  reverse = FALSE,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; selected value available as input\$id.
label	Input label.
choices	Named or unnamed vector / list of choices.
selected	Initially selected value.
...	Extra attributes applied to each radio <input>.
inline	If TRUE, lay the choices out horizontally.
reverse	If TRUE, put the label before the control (.form-check-reverse, Bootstrap 5.2).
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::radioButtons\(\)](#)

Examples

```
bs_radio_input("size", "Size", c("S", "M", "L"), selected = "M")
```

bs_range_input	<i>Bootstrap range (slider) input</i>
----------------	---------------------------------------

Description

A native <input type="range" class="form-range"> whose value is reported to the server as input\$id. Drive it server-side with [update_bs_range\(\)](#).

Usage

```
bs_range_input(
  id,
  label = NULL,
  value = NULL,
  min = 0,
  max = 100,
  step = NULL,
  ...,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial value. If NULL, the browser initializes the control at its midpoint $((\text{min} + \text{max}) / 2)$, which is what input\$id reports.
min, max, step	Numeric bounds and step.
...	Extra attributes applied to the <input> element.
help	Help text rendered below the control (<code>.form-text</code>).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

Examples

```
bs_range_input("vol", "Volume", value = 50, min = 0, max = 100)
```

bs_ratio	<i>Fixed aspect-ratio container</i>
----------	-------------------------------------

Description

Wraps an embedded element (image, iframe, video, ...) so it keeps a fixed aspect ratio (`.ratio`).

Usage

```
bs_ratio(..., ratio = "16x9", class = NULL)
```

Arguments

...	The embedded element and named HTML attributes.
ratio	Aspect ratio: one of "1x1", "4x3", "16x9", "21x9".
class	Extra classes.

Value

A <div> tag.

Examples

```
bs_ratio(shiny::tags$iframe(src = "https://example.com"), ratio = "16x9")
```

bs_row	<i>Bootstrap grid row</i>
--------	---------------------------

Description

Bootstrap grid row

Usage

```
bs_row(  
  ...,  
  cols = NULL,  
  gutters = NULL,  
  gx = NULL,  
  gy = NULL,  
  justify = NULL,  
  align = NULL,  
  class = NULL  
)
```

Arguments

...	Columns (bs_col()) and named HTML attributes.
cols	Number of equal-width columns per row (row-cols-*). A single value applies at all breakpoints; a named list (e.g. list(sm = 1, md = 2)) sets per-breakpoint counts.
gutters, gx, gy	Gutter width 0-5. gutters sets both axes; gx/gy override the horizontal / vertical gutter.
justify	Horizontal alignment of columns: "start", "center", "end", "around", "between", "evenly".
align	Vertical alignment of columns: "start", "center", "end".
class	Extra classes.

Value

A row tag.

Examples

```
bs_row(bs_col("a"), bs_col("b"), gutters = 3)
```

bs_scrollspy	<i>Bootstrap scrollspy container</i>
--------------	--------------------------------------

Description

Wraps content in a scrollable region whose nav (identified by `target`) is updated as the user scrolls.

Usage

```
bs_scrollspy(
  target,
  ...,
  id = NULL,
  offset = NULL,
  root_margin = NULL,
  threshold = NULL,
  smooth = TRUE,
  class = NULL
)
```

Arguments

<code>target</code>	Id (without #) of the nav / list-group that scrollspy drives.
<code>...</code>	Scrollable content and named HTML attributes.
<code>id</code>	Optional container id. The href of the currently active nav link is reported as <code>input\$id</code> . Defaults to a unique auto-generated id (an id is required for bootstrict to initialise scrollspy, including inside <code>renderUI()</code> — Bootstrap only auto-initialises on full page load).
<code>offset</code>	Pixels from the top to offset link activation (<code>data-bs-offset</code>). Deprecated in Bootstrap 5.3, which drives scrollspy with an <code>IntersectionObserver</code> : use <code>root_margin</code> instead.
<code>root_margin</code>	The observer's root margin (<code>data-bs-root-margin</code>), Bootstrap's replacement for <code>offset</code> . Defaults to <code>"0px 0px -25%"</code> .
<code>threshold</code>	Visibility ratios at which a section becomes active (<code>data-bs-threshold</code>), e.g. <code>c(0, 0.5, 1)</code> .
<code>smooth</code>	If TRUE, enable smooth scrolling (<code>data-bs-smooth-scroll</code>).
<code>class</code>	Extra classes.

Value

A scrollspy container tag, with the bootstrict dependency attached.

Examples

```
bs_scrollspy("nav-menu", shiny::tags$h4("Section"), offset = 100)
```

bs_select_input	<i>Bootstrap select input</i>
-----------------	-------------------------------

Description

Renders a native Bootstrap 5 `<select class="form-select">` (selectize is disabled to stay faithful to the Bootstrap markup).

Usage

```
bs_select_input(
  id,
  label = NULL,
  choices = NULL,
  selected = NULL,
  ...,
  multiple = FALSE,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as <code>input\$id</code> .
label	Input label.
choices	Named or unnamed vector / list of choices.
selected	Initially selected value(s).
...	Extra attributes applied to the <code><input></code> element.
multiple	Allow multiple selection.
size	Control size: "sm" or "lg".
help	Help text rendered below the control (<code>.form-text</code>).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::selectInput\(\)](#)

Examples

```
bs_select_input("fruit", "Fruit", c("Apple", "Pear"))
```

`bs_spinner`*Bootstrap spinner*

Description

An animated loading indicator. Render either a spinning border (`type = "border"`) or a pulsing dot (`type = "grow"`), optionally tinted with a theme colour and shrunk to the small variant.

Usage

```
bs_spinner(  
  type = c("border", "grow"),  
  color = NULL,  
  size = NULL,  
  label = "Loading...",  
  ...,  
  class = NULL  
)
```

Arguments

<code>type</code>	Spinner style: "border" (default) or "grow".
<code>color</code>	Theme colour (<code>. text-*</code>), one of the Bootstrap theme colours.
<code>size</code>	Spinner size; only "sm" (small) is accepted, NULL for the default size.
<code>label</code>	Visually hidden text announced to assistive technology.
<code>...</code>	Additional named HTML attributes applied to the spinner element.
<code>class</code>	Extra classes.

Value

A spinner tag.

Examples

```
bs_spinner(type = "border", color = "primary")
```

bs_table	<i>Bootstrap table</i>
----------	------------------------

Description

Render a faithful Bootstrap 5 `<table class="table">`. Pass a data frame or matrix in `data` to have the header and body built automatically from its column names and rows; otherwise supply `<thead>/<tbody>` markup (or any table children) through `...`

Usage

```
bs_table(  
  data = NULL,  
  ...,  
  striped = FALSE,  
  bordered = FALSE,  
  borderless = FALSE,  
  hover = FALSE,  
  small = FALSE,  
  variant = NULL,  
  responsive = FALSE,  
  align = NULL,  
  caption = NULL,  
  caption_top = FALSE,  
  head_variant = NULL,  
  group_divider = FALSE,  
  row_variant = NULL,  
  rownames = NULL,  
  class = NULL  
)
```

Arguments

<code>data</code>	A data frame or matrix to render. When <code>NULL</code> , the unnamed <code>...</code> arguments are used as the table's children instead.
<code>...</code>	Manual table children (when <code>data</code> is <code>NULL</code>) and named HTML attributes applied to the <code><table></code> element.
<code>striped</code>	Zebra-striping: <code>TRUE</code> or <code>"rows"</code> for <code>.table-striped</code> , <code>"columns"</code> for <code>.table-striped-columns</code> .
<code>bordered</code>	Add borders on all sides (<code>.table-bordered</code>).
<code>borderless</code>	Remove all borders (<code>.table-borderless</code>).
<code>hover</code>	Enable a hover state on rows (<code>.table-hover</code>).
<code>small</code>	Make the table more compact (<code>.table-sm</code>).
<code>variant</code>	Theme colour for the whole table (<code>.table-*</code>), one of the Bootstrap theme colours.
<code>responsive</code>	Make the table scroll horizontally on small devices. Use <code>TRUE</code> for <code>.table-responsive</code> , or a breakpoint string (<code>"sm"</code> , <code>"md"</code> , <code>"lg"</code> , <code>"xl"</code> , <code>"xxl"</code>) for <code>.table-responsive-{bp}</code> .

align	Vertical alignment of cells (<code>.align=*</code>), e.g. "middle", "top", "bottom".
caption	Optional table caption text rendered in a <code><caption></code> .
caption_top	If TRUE, place the caption above the table (<code>.caption-top</code>).
head_variant	Theme colour for the generated <code><thead></code> (<code>.table-light</code> , <code>.table-dark</code> , or any theme colour).
group_divider	If TRUE, add a thicker border between the header and the body (<code>.table-group-divider</code> on the generated <code><tbody></code>).
row_variant	Per-row theme colour for the generated rows (<code>.table-*</code> on the <code><tr></code>). A character vector recycled to the number of rows; NA leaves a row unstyled. "active" is also accepted.
rownames	Render each row's name as the reference <code><th scope="row"></code> header cell. NULL (the default) does so when data carries real row names, as <code>mtcars</code> does; TRUE forces it (numbering the rows when there are no names, like Bootstrap's own example); FALSE never does.
class	Extra classes.

Value

A table tag (wrapped in a responsive container when responsive is set).

Examples

```
bs_table(head(mtcars), striped = TRUE, hover = TRUE)
```

bs_tabset	<i>Bootstrap tabset</i>
-----------	-------------------------

Description

An interactive set of tabbed panels. The data-value of the currently shown panel is reported to the server as `input$id`, and the active tab can be driven server-side with `update_bs_tabset()`.

Usage

```
bs_tabset(  
  id,  
  ...,  
  type = "tabs",  
  selected = NULL,  
  fill = FALSE,  
  justified = FALSE,  
  vertical = FALSE,  
  class = NULL  
)  
  
bs_tab_panel(title, ..., value = NULL, icon = NULL, class = NULL)
```

Arguments

id	Tabset id; the active panel value is available as <code>input\$id</code> .
...	Panels built with <code>bs_tab_panel()</code> .
type	Visual style: "tabs" (default), "pills" or "underline" (Bootstrap 5.3).
selected	Value of the panel shown initially (defaults to the first).
fill	If TRUE, tabs expand to fill available width (<code>.nav-fill</code>).
justified	If TRUE, tabs get equal width (<code>.nav-justified</code>).
vertical	If TRUE, lay tabs out vertically beside the content.
class	Extra classes for the wrapper.
title	Tab label content.
value	Panel identifier reported to the server (defaults to title).
icon	Optional icon placed before the title.

Value

A tabset tag.

Examples

```
bs_tabset(
  "tabs",
  bs_tab_panel("Home", "Home content", value = "home"),
  bs_tab_panel("Profile", "Profile content", value = "profile"),
  selected = "profile"
)
```

bs_text_input

Bootstrap text input

Description

Bootstrap text input

Usage

```
bs_text_input(
  id,
  label = NULL,
  value = "",
  ...,
  placeholder = NULL,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as input\$id.
label	Input label.
value	Initial value.
...	Extra attributes applied to the <input> element.
placeholder	Placeholder text.
size	Control size: "sm" or "lg".
help	Help text rendered below the control (.form-text).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::textInput\(\)](#)

Examples

```
bs_text_input("name", "Your name", placeholder = "Jane Doe")
```

bs_textarea_input	<i>Bootstrap textarea input</i>
-------------------	---------------------------------

Description

Bootstrap textarea input

Usage

```
bs_textarea_input(
  id,
  label = NULL,
  value = "",
  ...,
  placeholder = NULL,
  rows = NULL,
  cols = NULL,
  size = NULL,
  help = NULL,
  width = NULL
)
```

Arguments

id	Input id; value available as <code>input\$id</code> .
label	Input label.
value	Initial value.
...	Extra attributes applied to the <code><input></code> element.
placeholder	Placeholder text.
rows, cols	Visible rows / columns.
size	Control size: "sm" or "lg".
help	Help text rendered below the control (<code>.form-text</code>).
width	CSS width (e.g. "100%", "200px").

Value

A form control tag.

See Also

[shiny::textAreaInput\(\)](#)

Examples

```
bs_textarea_input("bio", "Bio", rows = 3, placeholder = "A few words")
```

bs_toast

Bootstrap toast

Description

A lightweight, dismissible notification. The toast's visibility is reported to the server as `input$id` (TRUE when shown), and can be driven server-side with [show_bs_toast\(\)](#) / [hide_bs_toast\(\)](#). Place one or more toasts inside a [bs_toast_container\(\)](#) to position them on screen, or push transient notifications entirely from the server with [bs_notify_toast\(\)](#).

Usage

```
bs_toast(
  id,
  ...,
  title = NULL,
  icon = NULL,
  autohide = TRUE,
  delay = 5000,
  animation = TRUE,
  class = NULL
)
```

Arguments

id	Toast id; shown state available as input\$id.
...	Toast body content and named HTML attributes applied to the root.
title	Optional header title. When supplied, a .toast-header with the title and a close button is rendered.
icon	Optional icon placed before the title in the header.
autohide	If TRUE (default), hide the toast automatically after delay milliseconds.
delay	Delay in milliseconds before auto-hiding (when autohide).
animation	If FALSE, disable the fade animation.
class	Extra classes.

Value

A toast tag.

Examples

```
bs_toast("hello", "Hello, world!", title = "Bootstrict")
```

bs_toast_container	<i>Position toasts on screen</i>
--------------------	----------------------------------

Description

A fixed-position container that holds and lays out one or more `bs_toast()`s.

Usage

```
bs_toast_container(..., placement = "top-end", class = NULL)
```

Arguments

...	Toasts and named HTML attributes applied to the container.
placement	Where to anchor the container. One of "top-start", "top-center", "top-end", "middle-start", "middle-center", "middle-end", "bottom-start", "bottom-center" or "bottom-end".
class	Extra classes.

Value

A toast container tag.

Examples

```
bs_toast_container(
  bs_toast("hello", "Hi there", title = "Greeting"),
  placement = "top-end"
)
```

bs_tooltip	<i>Add a Bootstrap tooltip to a UI element</i>
------------	--

Description

Decorates an existing tag with the data attributes Bootstrap needs to render a tooltip. Tooltips are initialised client-side by bootstrick (Bootstrap does not auto-initialise them). A tag that is already a data-API trigger — a `bs_modal_trigger()`, `bs_offcanvas_trigger()`, `bs_collapse_trigger()` or a dropdown toggle — keeps its own data-bs-toggle; the tooltip works alongside it.

Usage

```
bs_tooltip(tag, title, ..., placement = "top", html = FALSE, trigger = NULL)
```

Arguments

tag	A UI element (a shiny.tag) to attach the tooltip to.
title	Tooltip text (or HTML, when html = TRUE).
...	Extra named attributes applied to tag.
placement	Tooltip placement: "auto", "top", "right", "bottom" or "left".
html	If TRUE, allow HTML content in the tooltip (data-bs-html).
trigger	How the tooltip is triggered (e.g. "hover focus", "click", "manual"); NULL uses the Bootstrap default.

Value

The decorated tag, with the bootstrick dependency attached.

Examples

```
bs_tooltip(shiny::tags$button("Hover me"), "Tooltip text")
```

bs_valid_feedback	<i>Bootstrap validation feedback</i>
-------------------	--------------------------------------

Description

Inline messages shown next to a control to convey its validation state. `bs_valid_feedback()` renders `.valid-feedback`; `bs_invalid_feedback()` renders `.invalid-feedback`.

Usage

```
bs_valid_feedback(..., class = NULL)

bs_invalid_feedback(..., class = NULL)
```

Arguments

... Feedback content and named HTML attributes.
class Extra classes.

Value

A div tag.

Examples

```
bs_valid_feedback("Looks good!")  
bs_invalid_feedback("Please choose a username.")
```

bs_visually_hidden	<i>Visually hidden text</i>
--------------------	-----------------------------

Description

Renders content available to assistive technologies but hidden from sighted users (`.visually-hidden`).

Usage

```
bs_visually_hidden(..., class = NULL)
```

Arguments

... Content and named HTML attributes.
class Extra classes.

Value

A `<span>` tag.

Examples

```
bs_visually_hidden("Loading, please wait")
```

`bs_vr`*Vertical rule*

Description

A vertical divider (`.vr`), the vertical counterpart of `<hr>`.

Usage

```
bs_vr(class = NULL)
```

Arguments

`class` Extra classes.

Value

A `<div>` tag.

Examples

```
bs_vr()
```

`close_bs_alert`*Close an alert from the server*

Description

Dismisses the `bs_alert()` registered under `id`, exactly as its close button does. Bootstrap removes the element, so the alert cannot be brought back: render it from a `renderUI()` if it has to come and go.

Usage

```
close_bs_alert(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

`id` Alert id, as passed to `bs_alert()`.
`session` The Shiny session.

Value

Nothing, called for its side effect.

See Also[bs_alert\(\)](#)**Examples**

```
if (interactive()) close_bs_alert("saved")
```

parse_scss_variables *Parse a SASS/SCSS variable sheet into a named list*

Description

Reads a `_variables.scss` style file (the kind a designer exports) and extracts top-level `$name: value;` declarations into a named list suitable for passing to [bootstrict_theme\(\)](#). Trailing `!default` / `!global` flags and line/block comments are stripped. Values are returned verbatim as strings (Sass resolves them at compile time), so maps, functions and colour expressions all pass straight through.

Usage

```
parse_scss_variables(path)
```

Arguments

path	Path to a <code>.scss</code> file (SCSS syntax, <code>\$name: value;</code> — the indented <code>.sass</code> syntax has no semicolons and cannot be parsed).
------	---

Details

The file is scanned rather than read line by line, so a declaration may span as many lines as it likes (`$theme-colors`, `$grid-breakpoints` and `$spacers` always do), and a `;`, `//` or `/* */` inside a quoted string or an unquoted `url()` is read as data. Anything that is not a top-level variable declaration — `@use/@import`, a rule block — is ignored. A final declaration with no trailing `;` is still read.

Value

A named list of Sass variable values. Names use the Bootstrap convention without the leading `$` (e.g. `primary`, `font-family-base`).

Examples

```
tmp <- tempfile(fileext = ".scss")
writeLines(c("$primary: #ff6600;", "$border-radius: 0.5rem !default;"), tmp)
parse_scss_variables(tmp)
```

set_bs_color_mode	<i>Switch the Bootstrap colour mode from the server</i>
-------------------	---

Description

Sets the Bootstrap 5.3 colour mode (data-bs-theme) on the document root, switching every component between light and dark. Set the initial mode with the color_mode argument of `bs_page()`.

Usage

```
set_bs_color_mode(mode, session = shiny::getDefaultReactiveDomain())
```

Arguments

mode	"light", "dark", or "auto" to follow the operating system.
session	The Shiny session.

Details

A choice is remembered in the browser's localStorage, so it survives a reload; "auto" forgets it and follows the operating system again. The mode actually in force is reported as `input$bootstrap_color_mode`, which is "light" or "dark" even when the preference is "auto".

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) set_bs_color_mode("dark")
```

set_bs_validation	<i>Set a control's validation state from the server</i>
-------------------	---

Description

Toggles `.is-valid` / `.is-invalid` on the control registered under `id` and, when `message` is supplied, replaces the text of the matching feedback message declared with `bs_feedback()`.

Usage

```
set_bs_validation(
  id,
  state = c("valid", "invalid", "none"),
  message = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

id	Input id, as passed to the <code>bs_*_input()</code> constructor.
state	"valid", "invalid", or "none" to clear the state.
message	Optional replacement text for the feedback message. Plain text: it is set with <code>textContent</code> .
session	The Shiny session.

Value

Nothing, called for its side effect.

See Also

[bs_feedback\(\)](#)

Examples

```
if (interactive()) {
  set_bs_validation("user", "invalid", "That name is taken.")
}
```

show_bs_dropdown	<i>Open, close or toggle a dropdown from the server</i>
------------------	---

Description

Drives the [bs_dropdown\(\)](#) or [bs_nav_dropdown\(\)](#) registered under id. Its open state is reported back as `input$id`.

Usage

```
show_bs_dropdown(id, session = shiny::getDefaultReactiveDomain())
```

```
hide_bs_dropdown(id, session = shiny::getDefaultReactiveDomain())
```

```
toggle_bs_dropdown(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

id	Dropdown id.
session	The Shiny session.

Value

Nothing, called for their side effect.

See Also[bs_dropdown\(\)](#)**Examples**

```
if (interactive()) {  
  show_bs_dropdown("menu")  
  hide_bs_dropdown("menu")  
  toggle_bs_dropdown("menu")  
}
```

`show_bs_modal`*Control a modal from the server*

Description

Show, hide or toggle a [bs_modal\(\)](#) from server code.

Usage

```
show_bs_modal(id, session = shiny::getDefaultReactiveDomain())  
  
hide_bs_modal(id, session = shiny::getDefaultReactiveDomain())  
  
toggle_bs_modal(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

<code>id</code>	Modal id (namespaced automatically inside modules).
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) show_bs_modal("info")  
if (interactive()) hide_bs_modal("info")  
if (interactive()) toggle_bs_modal("info")
```

show_bs_offcanvas	<i>Control an offcanvas from the server</i>
-------------------	---

Description

Show, hide or toggle a `bs_offcanvas()` from server code.

Usage

```
show_bs_offcanvas(id, session = shiny::getDefaultReactiveDomain())  
  
hide_bs_offcanvas(id, session = shiny::getDefaultReactiveDomain())  
  
toggle_bs_offcanvas(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

<code>id</code>	Offcanvas id (namespaced automatically inside modules).
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) show_bs_offcanvas("menu")  
if (interactive()) hide_bs_offcanvas("menu")  
if (interactive()) toggle_bs_offcanvas("menu")
```

show_bs_toast	<i>Control a toast from the server</i>
---------------	--

Description

Show or hide a `bs_toast()` from server code.

Usage

```
show_bs_toast(id, session = shiny::getDefaultReactiveDomain())  
  
hide_bs_toast(id, session = shiny::getDefaultReactiveDomain())
```

Arguments

<code>id</code>	Toast id (namespaced automatically inside modules).
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) show_bs_toast("hello")
if (interactive()) hide_bs_toast("hello")
```

update_bs_accordion	<i>Control an accordion from the server</i>
---------------------	---

Description

Control an accordion from the server

Usage

```
update_bs_accordion(
  id,
  open = NULL,
  close = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

id	Accordion id (namespaced automatically inside modules).
open	Panel value(s) to open. Use TRUE to open all (sensible with multiple = TRUE); FALSE/NULL is a no-op.
close	Panel value(s) to close. Use TRUE to close all; FALSE/NULL is a no-op.
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) {
  update_bs_accordion("acc", open = "two")
  update_bs_accordion("acc", close = TRUE)
}
```

update_bs_carousel	<i>Control a carousel from the server</i>
--------------------	---

Description

Control a carousel from the server

Usage

```
update_bs_carousel(  
  id,  
  to = NULL,  
  slide = NULL,  
  action = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id	Carousel id (namespaced automatically inside modules).
to	0-based index of the slide to cycle to. Takes precedence over slide when both are supplied.
slide	Direction to advance: "next" or "prev".
action	"pause" stops cycling, "cycle" resumes it.
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
## Not run:  
update_bs_carousel("demo", to = 2)  
update_bs_carousel("demo", slide = "next")  
  
## End(Not run)
```

update_bs_collapse	<i>Control a collapse from the server</i>
--------------------	---

Description

Show, hide or toggle a `bs_collapse()` from server code.

Usage

```
update_bs_collapse(
  id,
  action = c("toggle", "show", "hide"),
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

<code>id</code>	Collapse id (namespaced automatically inside modules).
<code>action</code>	One of "toggle", "show" or "hide".
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) update_bs_collapse("more", "show")
```

update_bs_color	<i>Update a colour input from the server</i>
-----------------	--

Description

Routes through `shiny::session`'s `sendInputMessage()` so the colour binding's `receiveMessage` updates the swatch.

Usage

```
update_bs_color(id, value, session = shiny::getDefaultReactiveDomain())
```

Arguments

<code>id</code>	Colour input id (namespaced automatically inside modules).
<code>value</code>	New colour as a hex string (e.g. "#0d6efd").
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) update_bs_color("col", "#198754")
```

update_bs_date_input *Set a date input from the server*

Description

The native counterpart of `shiny::updateDateInput()`, which cannot reach these controls.

Usage

```
update_bs_date_input(  
  id,  
  value = NULL,  
  min = NULL,  
  max = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)  
  
update_bs_date_range_input(  
  id,  
  start = NULL,  
  end = NULL,  
  min = NULL,  
  max = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id	Input id.
value	New date, or NA to clear the field.
min, max	New bounds.
session	The Shiny session.
start, end	New start / end dates, or NA to clear either field.

Value

Nothing, called for its side effect.

See Also

[bs_date_input\(\)](#)

Examples

```
if (interactive()) update_bs_date_input("day", value = Sys.Date())
```

update_bs_list_group *Control a list group selection from the server*

Description

Activates the item whose value matches selected in a selectable [bs_list_group\(\)](#) (one created with an id).

Usage

```
update_bs_list_group(
  id,
  selected = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

id	List group id (namespaced automatically inside modules).
selected	Item value to activate. NULL (the default) leaves the current selection unchanged; <code>character(0)</code> clears it (deselects every item and resets <code>input\$id</code> to NULL).
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
## Not run:
update_bs_list_group("my_group", selected = "two")
update_bs_list_group("my_group", selected = character(0)) # clear

## End(Not run)
```

update_bs_nav	<i>Set the active link of a nav from the server</i>
---------------	---

Description

Activates the `bs_nav_link()` carrying selected as its value inside the `bs_nav()` registered under id, and reports the change back as `input$id`.

Usage

```
update_bs_nav(id, selected = NULL, session = shiny::getDefaultReactiveDomain())
```

Arguments

id	Nav id, as passed to <code>bs_nav()</code> .
selected	value of the link to activate.
session	The Shiny session.

Value

Nothing, called for its side effect.

See Also

[bs_nav\(\)](#)

Examples

```
if (interactive()) update_bs_nav("menu", selected = "profile")
```

update_bs_pagination	<i>Set the active page of a pager from the server</i>
----------------------	---

Description

Activates the `bs_page_item()` carrying selected as its value inside the `bs_pagination()` registered under id, and reports the change back as `input$id`.

Usage

```
update_bs_pagination(  
  id,  
  selected = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id	Pager id, as passed to bs_pagination() or bs_pagination_numbered() .
selected	value of the item to activate. For a numbered pager this is the page number as a string.
session	The Shiny session.

Value

Nothing, called for its side effect.

See Also

[bs_pagination\(\)](#)

Examples

```
if (interactive()) update_bs_pagination("pager", selected = "3")
```

update_bs_progress	<i>Update a progress bar from the server</i>
--------------------	--

Description

Update a progress bar from the server

Usage

```
update_bs_progress(
  id,
  value = NULL,
  label = NULL,
  color = NULL,
  min = NULL,
  max = NULL,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

id	Id of the bs_progress_bar() track to update (namespaced automatically inside modules).
value	New value.
label	New text shown inside the bar.
color	New theme colour (<code>.bg-*</code>).
min, max	New lower / upper bounds of the scale.
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
## Not run:
update_bs_progress("load", value = 80)

## End(Not run)
```

update_bs_range	<i>Update a range input from the server</i>
-----------------	---

Description

Routes through [shiny::session](#)'s `sendInputMessage()` so the range binding's `receiveMessage` updates the slider (the "shiny-like" update path).

Usage

```
update_bs_range(id, value, session = shiny::getDefaultReactiveDomain())
```

Arguments

<code>id</code>	Range input id (namespaced automatically inside modules).
<code>value</code>	New value.
<code>session</code>	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
if (interactive()) update_bs_range("vol", 75)
```

update_bs_tabset	<i>Control a tabset from the server</i>
------------------	---

Description

Control a tabset from the server

Usage

```
update_bs_tabset(id, selected, session = shiny::getDefaultReactiveDomain())
```

Arguments

id	Tabset id (namespaced automatically inside modules).
selected	Value of the panel to show.
session	The Shiny session.

Value

Invisibly NULL, called for its side effect.

Examples

```
## Not run:  
update_bs_tabset("tabs", selected = "profile")  
  
## End(Not run)
```

update_bs_toggle_buttons	<i>Set the selection of a toggle button group from the server</i>
--------------------------	---

Description

Drives a `bs_radio_button_input()` or `bs_checkbox_button_input()`. Pass `character(0)` to clear a checkbox group.

Usage

```
update_bs_toggle_buttons(  
  id,  
  selected = NULL,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

id	Input id.
selected	Value(s) to select. A radio group keeps only the first.
session	The Shiny session.

Value

Nothing, called for its side effect.

See Also

[bs_radio_button_input\(\)](#)

Examples

```
if (interactive()) update_bs_toggle_buttons("size", selected = "1")
```

use_bootstrap	<i>Activate bootstrap inside a UI</i>
---------------	---------------------------------------

Description

Returns the bootstrap HTML dependency (Shiny input bindings + supporting CSS) so it can be dropped anywhere in a UI. Page constructors such as [bs_page\(\)](#) call this for you; use it directly when embedding bootstrap widgets into a UI you build by hand.

Usage

```
use_bootstrap()
```

Value

An [htmltools::htmlDependency](#).

Examples

```
shiny::fluidPage(use_bootstrap(), bs_card(bs_card_body("Hello")))
```

use_bootstrap_golem *Scaffold a bootstrap app as a golem project hook*

Description

A `golem::create_golem()` project_hook that turns a freshly created golem skeleton into a minimal bootstrap application:

Usage

```
use_bootstrap_golem(path, package_name, ...)
```

Arguments

path	Path of the newly created golem project. Unused, kept for compatibility with the golem hook interface.
package_name	Name of the package / application being created.
...	Reserved for compatibility with the golem hook interface.

Details

- `R/app_ui.R` is rewritten to use `bs_page()` and shows a "Hello world" inside a `bs_container()`.
- An (empty by default) `inst/app/www/_variables.scss` designer variable sheet is created and wired into `bootstrap_theme()`, so a designer can drop SCSS variables (`$name: value;`) in and have them picked up automatically.

`golem` sets the working directory to the new project before calling the hook, so every path below is relative to the application root.

Value

Invisibly NULL. Called for its side effects on the project files.

Examples

```
if (interactive()) {
  golem::create_golem(
    "my.app",
    project_hook = bootstrap::use_bootstrap_golem
  )
}
```

Index

bootstrap_dep, 4
bootstrap_dep(), 6
bootstrap_version, 5
bootstrict_dep, 5
bootstrict_theme, 6
bootstrict_theme(), 4, 52, 77, 92
bs_accordion, 7
bs_accordion_panel(bs_accordion), 7
bs_accordion_panel(), 7
bs_alert, 8
bs_alert(), 76, 77
bs_alert_heading(bs_alert), 8
bs_alert_link(bs_alert), 8
bs_badge, 9
bs_blockquote, 10
bs_breadcrumb, 10
bs_breadcrumb_item(bs_breadcrumb), 10
bs_breadcrumb_item(), 10, 11
bs_button, 11
bs_button(), 12, 26, 27, 38, 43, 51
bs_button_group, 12
bs_button_group(), 61
bs_button_toolbar(bs_button_group), 12
bs_card, 13
bs_card_body(bs_card), 13
bs_card_body(), 13
bs_card_footer(bs_card), 13
bs_card_footer(), 13
bs_card_group(bs_card), 13
bs_card_header(bs_card), 13
bs_card_header(), 13
bs_card_img(bs_card), 13
bs_card_img(), 13
bs_card_img_overlay(bs_card), 13
bs_card_img_overlay(), 14
bs_card_link(bs_card), 13
bs_card_link(), 14
bs_card_subtitle(bs_card), 13
bs_card_text(bs_card), 13
bs_card_title(bs_card), 13
bs_carousel, 14
bs_carousel(), 16
bs_carousel_item(bs_carousel), 14
bs_carousel_item(), 14, 15
bs_checkbox_button_input
 (bs_radio_button_input), 60
bs_checkbox_button_input(), 90
bs_checkbox_group_input, 16
bs_checkbox_group_input(), 38
bs_checkbox_input, 17
bs_close_button, 18
bs_col, 19
bs_col(), 22, 64
bs_collapse, 20
bs_collapse(), 21, 84
bs_collapse_trigger, 21
bs_collapse_trigger(), 20, 57, 74
bs_color_input, 21
bs_container, 22
bs_container(), 92
bs_date_input, 23
bs_date_input(), 24, 25, 38, 86
bs_date_range_input, 24
bs_date_range_input(), 24, 38
bs_display_heading, 25
bs_download_button, 26
bs_download_button(), 27
bs_download_link, 27
bs_download_link(), 27
bs_dropdown, 28
bs_dropdown(), 45, 46, 79, 80
bs_dropdown_divider(bs_dropdown), 28
bs_dropdown_divider(), 28, 46
bs_dropdown_header(bs_dropdown), 28
bs_dropdown_header(), 28, 46
bs_dropdown_item(bs_dropdown), 28
bs_dropdown_item(), 28, 46
bs_dropdown_text(bs_dropdown), 28

bs_dropdown_text(), 28, 46
 bs_feedback, 30
 bs_feedback(), 78, 79
 bs_figure, 31
 bs_figure(), 31
 bs_figure_caption(bs_figure), 31
 bs_figure_caption(), 31
 bs_figure_img(bs_figure), 31
 bs_figure_img(), 31
 bs_file_input, 32
 bs_file_input(), 38
 bs_floating_label, 33
 bs_form, 34
 bs_form_label, 34
 bs_form_text, 35
 bs_hstack, 35
 bs_hstack(), 35
 bs_icon_link, 36
 bs_img, 37
 bs_input_group, 38
 bs_input_group_text(bs_input_group), 38
 bs_input_group_text(), 38
 bs_invalid_feedback
 (bs_valid_feedback), 74
 bs_lead, 39
 bs_list_group, 39
 bs_list_group(), 40, 86
 bs_list_group_item(bs_list_group), 39
 bs_list_group_item(), 39, 40
 bs_list_inline(bs_list_unstyled), 41
 bs_list_unstyled, 41
 bs_modal, 41
 bs_modal(), 43, 80
 bs_modal_body(bs_modal), 41
 bs_modal_body(), 42
 bs_modal_footer(bs_modal), 41
 bs_modal_footer(), 42
 bs_modal_header(bs_modal), 41
 bs_modal_header(), 42
 bs_modal_title(bs_modal), 41
 bs_modal_trigger, 43
 bs_modal_trigger(), 57, 74
 bs_nav, 44
 bs_nav(), 45–47, 87
 bs_nav_dropdown, 45
 bs_nav_dropdown(), 29, 46, 79
 bs_nav_item(bs_nav), 44
 bs_nav_item(), 44
 bs_nav_link(bs_nav), 44
 bs_nav_link(), 44, 87
 bs_navbar, 46
 bs_navbar_brand(bs_navbar), 46
 bs_navbar_brand(), 46, 47
 bs_navbar_nav(bs_navbar), 46
 bs_navbar_nav(), 45, 46
 bs_navbar_text(bs_navbar), 46
 bs_navbar_text(), 46
 bs_notify_toast, 48
 bs_notify_toast(), 72
 bs_numeric_input, 49
 bs_offcanvas, 50
 bs_offcanvas(), 51, 81
 bs_offcanvas_trigger, 51
 bs_offcanvas_trigger(), 50, 57, 74
 bs_page, 52
 bs_page(), 4, 78, 91, 92
 bs_page_fillable(bs_page), 52
 bs_page_fluid(bs_page), 52
 bs_page_item, 53
 bs_page_item(), 54, 87
 bs_pagination, 54
 bs_pagination(), 53, 55, 87, 88
 bs_pagination_numbered, 55
 bs_pagination_numbered(), 54, 88
 bs_password_input, 56
 bs_placeholder, 57
 bs_placeholder(), 57
 bs_placeholder_glow(bs_placeholder), 57
 bs_placeholder_glow(), 57
 bs_placeholder_wave(bs_placeholder), 57
 bs_placeholder_wave(), 57
 bs_popover, 57
 bs_progress, 58
 bs_progress(), 58
 bs_progress_bar(bs_progress), 58
 bs_progress_bar(), 58, 59, 88
 bs_radio_button_input, 60
 bs_radio_button_input(), 90, 91
 bs_radio_input, 61
 bs_radio_input(), 38
 bs_range_input, 62
 bs_ratio, 63
 bs_row, 64
 bs_row(), 22
 bs_scrollspy, 65
 bs_select_input, 66

bs_select_input(), 33
 bs_spinner, 67
 bs_switch_input (bs_checkbox_input), 17
 bs_tab_panel (bs_tabset), 69
 bs_tab_panel(), 70
 bs_table, 68
 bs_tabset, 69
 bs_tabset(), 44, 45
 bs_text_input, 70
 bs_text_input(), 33, 38
 bs_textarea_input, 71
 bs_toast, 72
 bs_toast(), 73, 81
 bs_toast_container, 73
 bs_toast_container(), 49, 72
 bs_tooltip, 74
 bs_valid_feedback, 74
 bs_valid_feedback(), 30
 bs_visually_hidden, 75
 bs_vr, 76
 bs_vstack (bs_hstack), 35
 bs_vstack(), 35

 close_bs_alert, 76
 close_bs_alert(), 8

 golem::create_golem(), 92

 hide_bs_dropdown (show_bs_dropdown), 79
 hide_bs_dropdown(), 29
 hide_bs_modal (show_bs_modal), 80
 hide_bs_modal(), 41
 hide_bs_offcanvas (show_bs_offcanvas), 81
 hide_bs_offcanvas(), 50
 hide_bs_toast (show_bs_toast), 81
 hide_bs_toast(), 72
 htmltools::htmlDependency, 4, 5, 91

 parse_scss_variables, 77
 parse_scss_variables(), 6

 set_bs_color_mode, 78
 set_bs_color_mode(), 52
 set_bs_validation, 78
 set_bs_validation(), 30
 shiny::actionButton(), 11
 shiny::checkboxGroupInput(), 16, 17
 shiny::checkboxInput(), 18
 shiny::dateInput(), 23
 shiny::downloadButton(), 26
 shiny::downloadHandler(), 26, 27
 shiny::downloadLink(), 27
 shiny::fileInput(), 32, 33
 shiny::numericInput(), 50
 shiny::passwordInput(), 56
 shiny::radioButtons(), 61, 62
 shiny::selectInput(), 66
 shiny::session, 84, 89
 shiny::showNotification(), 48
 shiny::textAreaInput(), 72
 shiny::textInput(), 71
 show_bs_dropdown, 79
 show_bs_dropdown(), 29, 46
 show_bs_modal, 80
 show_bs_modal(), 41
 show_bs_offcanvas, 81
 show_bs_offcanvas(), 50
 show_bs_toast, 81
 show_bs_toast(), 72

 toggle_bs_dropdown (show_bs_dropdown), 79
 toggle_bs_dropdown(), 29
 toggle_bs_modal (show_bs_modal), 80
 toggle_bs_modal(), 41
 toggle_bs_offcanvas
 (show_bs_offcanvas), 81
 toggle_bs_offcanvas(), 50

 update_bs_accordion, 82
 update_bs_accordion(), 7
 update_bs_carousel, 83
 update_bs_carousel(), 14
 update_bs_collapse, 84
 update_bs_collapse(), 20
 update_bs_color, 84
 update_bs_color(), 21
 update_bs_date_input, 85
 update_bs_date_input(), 23, 24
 update_bs_date_range_input
 (update_bs_date_input), 85
 update_bs_date_range_input(), 25
 update_bs_list_group, 86
 update_bs_list_group(), 39
 update_bs_nav, 87
 update_bs_nav(), 45, 47
 update_bs_pagination, 87

- update_bs_pagination(), [54](#), [55](#)
- update_bs_progress, [88](#)
- update_bs_progress(), [58](#), [59](#)
- update_bs_range, [89](#)
- update_bs_range(), [62](#)
- update_bs_tabset, [90](#)
- update_bs_tabset(), [69](#)
- update_bs_toggle_buttons, [90](#)
- update_bs_toggle_buttons(), [61](#)
- use_bootstrap, [91](#)
- use_bootstrap_golem, [92](#)