# Package 'inshiny'

March 31, 2026

**Type** Package

**Title** Compact Inline Widgets for 'shiny' Apps

**Version** 0.1.4

**Description** Provides a basic set of compact widgets for 'shiny' apps which occupy less space and can appear inline with surrounding text.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**URL** https://github.com/nicholasdavies/inshiny, https://nicholasdavies.github.io/inshiny/

**BugReports** https://github.com/nicholasdavies/inshiny/issues

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0), withr

**Config/testthat/edition** 3

**Imports** bslib, htmltools, rlang, shiny, stringr

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Nick Davies [aut, cre]

**Maintainer** Nick Davies <nicholas.davies@lshtm.ac.uk>

**Repository** CRAN

**Date/Publication** 2026-03-31 18:00:02 UTC

## Contents

---

`inline` *Container for inline widgets*

---

## Description

Wrapper for a line (or paragraph) containing a mix of explanatory text and inshiny inline widgets.

## Usage

```
inline(..., class = "mb-1")
```

## Arguments

| | |
|---|---|
| `...` | Unnamed arguments: Inline widgets (such as `inline_text()`), character strings, or HTML tags that will appear next to each other in a line or paragraph. These are pasted together with no spaces between them, so add extra spaces to your character strings if needed. Named arguments (e.g. `style`) are additional attributes for the HTML div tag wrapping the line. |
| `class` | Extra classes to apply to the line. The default, `"mb-1"`, is a Bootstrap 5 class that adds a small amount of margin to the bottom of the line. You can use `"mb-0"` through `"mb-5"`, other Bootstrap 5 spacing classes, or anything else. For multiple classes, provide one space-separated string. |

## Value

An HTML element to be included in your Shiny UI definition.

## Note on rendering

Inline widgets from inshiny are typically displayed with borders around them to show that they are editable. For the borders to display correctly, inshiny includes some custom CSS to make sure the borders work inside common bslib containers such as bslib::card and bslib::accordion_panel. However, if you place inline widgets inside a custom container that has its own opaque background (e.g. a `div` with `background-color` set), the border around widgets may disappear. To fix this, add the CSS class `"inshiny-bg"` to the container:

```
tags$div(class = "inshiny-bg", style = "background-color: white;",
    inline("Enter a value: ", inline_text("val", "hello"))
)
```

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Hello!"),
    inline("My name is ", inline_text("myname", "Sisyphus"), "."),
    inline("Please enter your name carefully.", style = "font-weight:bold")
)
```

---

`inline_button` *Inline action button*

---

## Description

A button widget similar to `shiny::actionButton()` that can be included in an `inline()` wrapper.

## Usage

```
inline_button(id, label, icon = NULL, meaning = label, accent = NULL)
```

## Arguments

| | |
|---|---|
| `id` | The `input` slot that will be used to access the value. |
| `label` | The text appearing within the button. This can be a character string or any other HTML, or `NULL` for no text (but then you will probably at least want an `icon`). |
| `icon` | An optional `shiny::icon()` which will appear to the left of the button. |
| `meaning` | A descriptive label, for people using assistive technology such as screen readers. |
| `accent` | A Bootstrap "accent" (such as `"primary"`, `"danger"`, etc.) that will be used to set the class of the button (such as `"btn-primary"`, etc.), or `NULL` for the default (`"btn-default"`). See Bootstrap 5 buttons for all the options. If you provide multiple accents in a character vector, each one will be appended to `"btn-"` and added to the button. |

## Value

An inline widget to be included in an `inline()` wrapper.

## See Also

shiny::actionButton for how the button works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("A wonderful button"),
    inline("To update, please feel free to press the ",
        inline_button("mybutton",
            label = shiny::span(style = "font-style:italic", "button"),
            icon = shiny::icon("play"),
```

```
            meaning = "Update button", accent = "success"),
        "."
    )
)
```

---

`inline_date` *Inline date input with calendar*

---

## Description

A date input with a calendar pop-up similar to `shiny::dateInput()` that can be included in an `inline()` wrapper.

## Usage

```
inline_date(
  id,
  value = NULL,
  min = NULL,
  max = NULL,
  placeholder = "Enter date",
  meaning = NULL,
  max_width = "10em",
  format = "yyyy-mm-dd",
  startview = "month",
  weekstart = 0,
  language = "en",
  autoclose = TRUE,
  datesdisabled = NULL,
  daysofweekdisabled = NULL
)
```

## Arguments

| | |
|---|---|
| `id` | The `input` slot that will be used to access the value. |
| `value` | The initially selected date. Either a Date object; a character string in `"yyyy-mm-dd"` format (*not* in the calendar's display format); or `NULL` to use the current date in the client's time zone. |
| `min, max` | The minimum and maximum allowed date. Either a Date object, a character string in `"yyyy-mm-dd"` format, or `NULL` for no limit. |
| `placeholder` | The character string or HTML element that will appear in the textbox when it is empty, as a prompt. |
| `meaning` | A descriptive label, for people using assistive technology such as screen readers. |
| `max_width` | The maximum width of the text input as a CSS length (e.g. `"10em"`, `"200px"`). When the text is longer than this, the input becomes horizontally scrollable. The default is `"10em"`. Use `NULL` for no limit. |

| | |
|---|---|
| format | The format of the date to display in the browser; defaults to "yyyy-mm-dd". Note that this is only for display purposes. Changing the display format does not allow you to specify value, min, max, or datesdisabled in that format; those have to stay formatted as "yyyy-mm-dd" or as Date objects. See shiny::dateInput for format details. |
| startview | The view shown when the textbox is first clicked. Can be "month", the default, for the usual monthly calendar view, "year", or "decade". |
| weekstart | Which day is the start of the week; an integer from 0 (Sunday) to 6 (Saturday). |
| language | The language used for month and day names, with "en" (English) as the default. See shiny::dateInput for options. |
| autoclose | Whether to close the calendar once a date has been selected. |
| datesdisabled | Dates that should be disabled (a character or Date vector). Strings should be in the "yyyy-mm-dd" format. |
| daysofweekdisabled | Days of the week that should be disabled; an integer vector in which 0 is Sunday, and 6 is Saturday. |

## Value

An inline widget to be included in an `inline()` wrapper.

## See Also

shiny::dateInput for how the date input works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Select a date"),
    inline("Run simulation starting on ",
        inline_date("start_date", NULL, meaning = "Simulation start date",
            format = "dd/mm/yyyy", daysofweekdisabled = c(0, 6)),
        " (weekdays only)."
    )
)
```

---

# inline_link *Inline action link*

---

## Description

A link widget similar to `shiny::actionLink()` that can be included in an `inline()` wrapper.

## Usage

```
inline_link(id, label, icon = NULL, meaning = label, accent = NULL)
```

### Arguments

| | |
|---|---|
| id | The input slot that will be used to access the value. |
| label | The text appearing within the link. This can be a character string or any other HTML, or NULL for no text (but then you will probably at least want an icon). |
| icon | An optional shiny::icon() which will appear to the left of the link. |
| meaning | A descriptive label, for people using assistive technology such as screen readers. |
| accent | A Bootstrap "accent" (such as "primary", "danger", etc.) that will be used to set the class of the link (such as "link-primary", etc.), or NULL for no special styling. See Bootstrap 5 link utilities for all the options. If you provide multiple accents in a character vector, each one will be appended to "link-" and added to the link. |

### Value

An inline widget to be included in an inline() wrapper.

### See Also

shiny::actionLink for how the link works with your Shiny server.

### Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Link examples"),
    inline("These are ", inline_link("link1", "some"), " ",
        inline_link("link2", "increasingly", accent = "danger"), " ",
     inline_link("link3", "fancy", accent = c("success", "underline-warning", "offset-2")), " ",
        inline_link("link4", "links", icon = shiny::icon("link"), accent = "info"), "!")
)
```

---

## inline_number *Inline number input*

---

### Description

A single line numeric input similar to shiny::numericInput() that can be included in an inline() wrapper.

### Usage

```
inline_number(
  id,
  value,
  min = NULL,
  max = NULL,
  step = NULL,
```

```
  default = value,
  placeholder = "Enter number",
  meaning = NULL,
  max_width = "10em",
  arrows = TRUE
)
```

## Arguments

| | |
|---|---|
| id | The input slot that will be used to access the value. |
| value | The initial number. |
| min, max | Minimum and maximum values. Use NULL for no limit. |
| step | A step value for incrementing and decrementing the number using the up or down arrow keys or with the clickable arrows on the widget. The Page Up and Page Down keys increment or decrement the number by 10 steps, and the Home and End keys set the number to the minimum or maximum respectively. The default step is 1. |
| default | A default value to be used if the input is invalid or empty. |
| placeholder | The character string or HTML element that will appear in the textbox when it is empty, as a prompt. |
| meaning | A descriptive label, for people using assistive technology such as screen readers. |
| max_width | The maximum width of the text input as a CSS length (e.g. "10em", "200px"). When the text is longer than this, the input becomes horizontally scrollable. The default is "10em". Use NULL for no limit. |
| arrows | Whether to show clickable arrows that can be used to adjust the number up or down. |

## Value

An inline widget to be included in an inline() wrapper.

## See Also

shiny::numericInput for how the number input works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Breakfast app (beta)"),
    inline("Make me an omelette with ",
        inline_number("eggs", 6, min = 2, max = 12, step = 1,
            placeholder = "6 (default)", meaning = "Number of eggs"),
        " eggs.")
)
```

`inline_select` *Inline select list input*

## Description

A select list input similar to `shiny::selectInput()` that can be included in an `inline()` wrapper.

## Usage

```
inline_select(
  id,
  choices,
  selected = NULL,
  multiple = FALSE,
  meaning = NULL,
  max_width = "10em"
)
```

## Arguments

| | |
|---|---|
| `id` | The `input` slot that will be used to access the value. |
| `choices` | Vector or list of values to select from. Provide one of the following:<br>• Use an unnamed character vector, such as `c("dog", "cat", "bee")`, for the most basic case, where you have a list of strings you want the user to select from.<br>• Use a named character vector, such as `c("Dog" = "dog", "Nice Kitty" = "cat", "Bee" = "bee")` if you want the options displayed to the user (the names; here, Dog, Nice Kitty, and Bee) to differ from the values passed to Shiny (the values; here, `"dog"`, `"cat"`, and `"bee"`).<br>• Use a named list, where each element is a "sub-list", to group the items under headings; the names at the top level of the list will be the heading titles and the "sub-lists" are the items appearing under that heading. For example, if you pass `list(Mammals = c("Dog" = "dog", "Nice Kitty" = "cat"), Invertebrates = c("Bee" = "bee"))` then Dog and Nice Kitty will appear under the Mammals heading, while Bee will appear under the Invertebrates heading, and the value passed to the Shiny server will be either `"dog"`, `"cat"`, or `"bee"`. |
| `selected` | The initially selected option's value. If `NULL`, use the first item in `choices`. |
| `multiple` | Whether to allow multiple selections. As of inshiny version 0.1.0, the version of inline_select with `multiple = TRUE` looks and behaves a bit differently from the version of inline_select with `multiple = FALSE`. The package authors are working on eliminating this inconsistency. |
| `meaning` | A descriptive label, for people using assistive technology such as screen readers. |
| `max_width` | The maximum width of the text input as a CSS length (e.g. `"10em"`, `"200px"`). When the text is longer than this, the input becomes horizontally scrollable. The default is `"10em"`. Use `NULL` for no limit. |

## Value

An inline widget to be included in an `inline()` wrapper.

## See Also

shiny::selectInput for how the select input works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Pet registration form"),
    inline("My ",
        inline_select("species", c("dog", "cat"), meaning = "Pet species"),
        "'s name is ",
        inline_select("name", list("Dog names" = c("Fido", "Rex"),
            "Cat names" = c("Felix", "Boots")), selected = "Rex"),
        ".")
)
```

---

# inline_slider *Inline slider input*

---

## Description

A numeric input with a slider pop-up similar to `shiny::sliderInput()` that can be included in an `inline()` wrapper.

## Usage

```
inline_slider(
  id,
  value,
  min,
  max,
  step = NULL,
  default = value,
  placeholder = "Enter number",
  meaning = NULL,
  max_width = "10em"
)
```

## Arguments

| | |
|---|---|
| `id` | The `input` slot that will be used to access the value. |
| `value` | The initial number. |
| `min, max` | Minimum and maximum values. Both are required. |

| | |
|---|---|
| step | A step value that the slider will use to jump between values between min and max. |
| default | A default value to be used if the input is invalid or empty. |
| placeholder | The character string or HTML element that will appear in the textbox when it is empty, as a prompt. |
| meaning | A descriptive label, for people using assistive technology such as screen readers. |
| max_width | The maximum width of the text input as a CSS length (e.g. "10em", "200px"). When the text is longer than this, the input becomes horizontally scrollable. The default is "10em". Use NULL for no limit. |

## Value

An inline widget to be included in an inline() wrapper.

## See Also

shiny::sliderInput for how the slider input works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Pep talk"),
    inline("When you go out there tonight, give ",
        inline_slider("amount", 10, 0, 110, step = 1, default = 50,
            placeholder = "Enter a percentage.", meaning = "Percent to give"),
        "%.")
)
```

---

# inline_switch *Inline on/off switch*

---

## Description

An on/off switch widget similar to bslib::input_switch() that can be included in an inline() wrapper.

## Usage

```
inline_switch(id, value, on = "On", off = "Off", meaning = NULL)
```

## Arguments

| | |
|---|---|
| id | The input slot that will be used to access the value. |
| value | Whether the switch is initially off or on; FALSE for off, TRUE for on. |
| on, off | Labels that will appear to the right of the switch when the switch is on or off, respectively. These can be character strings or HTML elements. For example, you can style these with a span and apply one of the Bootstrap 5 text color classes (see examples). NULL for no labels. |
| meaning | A descriptive label, for people using assistive technology such as screen readers. |

## Value

An inline widget to be included in an `inline()` wrapper.

## See Also

bslib::input_switch for how the switch works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Switch test"),
    inline("The server is now ",
        inline_switch("myswitch", TRUE,
            on = shiny::span(class = "text-success", "powered ON"),
            off = shiny::span(class = "text-danger", "powered OFF"),
            meaning = "Server power switch"),
        "."
    )
)
```

---

# inline_text *Inline text input*

---

## Description

A single line text input similar to `shiny::textInput()` that can be included in an `inline()` wrapper.

## Usage

```
inline_text(
  id,
  value = "",
  placeholder = "Enter text",
  meaning = NULL,
  max_width = "10em"
)
```

## Arguments

| | |
|---|---|
| `id` | The `input` slot that will be used to access the value. |
| `value` | The initial text contents (a character string). |
| `placeholder` | The character string or HTML element that will appear in the textbox when it is empty, as a prompt. |
| `meaning` | A descriptive label, for people using assistive technology such as screen readers. |
| `max_width` | The maximum width of the text input as a CSS length (e.g. `"10em"`, `"200px"`). When the text is longer than this, the input becomes horizontally scrollable. The default is `"10em"`. Use `NULL` for no limit. |

## Value

An inline widget to be included in an `inline()` wrapper.

## See Also

shiny::textInput for how the text input works with your Shiny server.

## Examples

```
ui <- bslib::page_fixed(
    shiny::h1("Hello!"),
    inline("My name is ", inline_text("myname", "Sisyphus",
        placeholder = "Enter your name", meaning = "Your name"), ".")
)
```

---

# update_inline *Update an inline widget*

---

## Description

Use this in your server code to change the parameters of an existing inline widget. Most, but not all, parameters from the corresponding `inline_*` functions can be changed. Note that while Shiny has a separate update function for each type of widget (e.g. shiny::updateTextInput for shiny::textInput, etc.), the inshiny package has this single function to update all types of inline widgets. This function can only be called in a reactive context, and can only be called on `inline_*` widgets, not on "built-in" Shiny widgets.

## Usage

```
update_inline(
  id,
  session = shiny::getDefaultReactiveDomain(),
  value,
  placeholder,
  meaning,
  label,
  icon,
  accent,
  min,
  max,
  step,
  default,
  on,
  off,
  datesdisabled,
  daysofweekdisabled,
  choices,
  selected
)
```

## Arguments

| | |
|---|---|
| id | The id of the inline widget to change. |
| session | The currently active Shiny session. In almost all cases you can leave this to its default value. |
| value | (date, number, select, slider, switch, text) The current value of the widget. |
| placeholder | (date, number, slider, text) The character string or HTML element that will appear when the widget's textbox is empty, as a prompt. |
| meaning | (all widgets) The descriptive label for people using assistive technology such as screen readers. |
| label, icon | (button, link) The label and icon that appear in the button or link. |
| accent | (button, link) The Boostrap accent to apply to the button or link. |
| min, max | (date, number, slider) The minimum and maximum allowable value. |
| step | (number, slider) The increment or decrement by which to change the value. |
| default | (number, slider) The default value to assume when the input is blank or invalid. |
| on, off | (switch) Labels to use for when the switch is on or off. |
| datesdisabled, daysofweekdisabled | (date) Dates to make unselectable. |
| choices, selected | (select) Options to choose from and current selection. |

## Details

See the documentation for each inline widget for details of how each parameter is interpreted.

When adjusting min, max, datesdisabled, or daysofweekdisabled, it is recommended that you also send an update to value with the current value of input[[id]] or any new value as applicable. This will ensure that any invalid value gets highlighted in the app as invalid after accounting for the new bounds and disallowed values.

## Value

Nothing.

## Examples

```
# Example UI setup
ui <- bslib::page_fixed(
    inline(
        inline_button("mybutton", "Button"),
        inline_date("mydate"),
        inline_link("mylink", "Link"),
        inline_number("mynumber", 42),
        inline_select("myselect", letters),
        inline_slider("myslider", 42, 0, 100),
        inline_switch("myswitch", TRUE),
        inline_text("mytext")
    )
```

```
)

# This covers all updatable attributes except `meaning` (all widgets) and
# `placeholder` (date, number, slider, text).
server <- function(input, output) {
    update_inline("mybutton", label = "Click me", icon = shiny::icon("recycle"),
        accent = "info")
    update_inline("mydate", value = "2026-01-01", min = "2025-01-01",
        max = "2026-12-31", datesdisabled = "2025-12-25",
        daysofweekdisabled = c(0, 6))
    update_inline("mylink", label = "Click me", icon = shiny::icon("recycle"),
        accent = "info")
    update_inline("mynumber", value = 25, min = 20, max = 50, step = 5,
        default = 25)
    update_inline("myselect", choices = letters[1:5], selected = "c")
    update_inline("myslider", value = 25, min = 20, max = 50, step = 5,
        default = 25)
    update_inline("myswitch", value = TRUE, on = "Present", off = "Absent")
    update_inline("mytext", value = "Howdy")
}
```

---

`use_inshiny` *Manually include inshiny scripts and stylesheet*

---

## Description

For inshiny to work, you need to link your Shiny app to inshiny's JavaScript code and CSS stylesheets. If you use inshiny's function `inline()` anywhere in your Shiny UI definition, which you probably do, this happens automatically. Otherwise, you can add a call to `use_inshiny()` to your UI.

## Usage

```
use_inshiny()
```

## Value

An `htmltools::htmlDependency()` object to include in your UI.

## Examples

```
ui <- bslib::page(
    use_inshiny(),
    shiny::h1("My slider app"),
    inline_slider("slider", 50, 0, 100)
)
```

# Index