Package ‘glasstabs’

March 11, 2026

Title Animated Glass-Style Tabs and Multi-Select Filter for 'Shiny'

Version 0.1.0

Description Tools for creating animated glassmorphism-style tab navigation and multi-select dropdown filters in 'shiny' applications. The package provides a tab navigation component and a searchable multi-select widget with multiple checkbox indicator styles, select-all controls, and customizable colour themes. The widgets are compatible with standard 'shiny' layouts and 'bs4Dash' dashboards.

License MIT + file LICENSE

URL `https://github.com/PrigasG/glasstabs`, `https://prigasg.github.io/glasstabs/`

BugReports `https://github.com/PrigasG/glasstabs/issues`

Imports htmltools (>= 0.5.0), shiny (>= 1.7.0)

Suggests bs4Dash, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

Language en-US

NeedsCompilation no

Author George Arthur [aut, cre]

Maintainer George Arthur <`prigasgenthian48@gmail.com`>

Repository CRAN

Date/Publication 2026-03-11 16:50:08 UTC

Contents

`glassFilterTags` *Shiny tag helper for a filter-tags display area tied to a glassMultiSelect*

Description

Renders a `<div>` that the JS engine will populate with colored tag pills whenever the corresponding `glassMultiSelect()` selection changes.

Usage

```
glassFilterTags(inputId, class = NULL)
```

Arguments

`inputId`	The `inputId` of the `glassMultiSelect()` this display should reflect.
`class`	Additional CSS classes for the container.

Value

An `htmltools` tag.

`glassMultiSelect` *Animated glass multi-select dropdown filter*

Description

Animated glass multi-select dropdown filter

Usage

```
glassMultiSelect(
  inputId,
  choices,
  selected = NULL,
  placeholder = "Filter by Category",
  check_style = c("checkbox", "check-only", "filled"),
  show_style_switcher = TRUE,
  show_select_all = TRUE,
  show_clear_all = TRUE,
  theme = "dark",
  hues = NULL
)
```

Arguments

`inputId`	Shiny input id. Selected values available as `input$<inputId>` (character vector) and active style as `input$<inputId>_style`.
`choices`	Named or unnamed character vector of choices.
`selected`	Initially selected values. Defaults to all.
`placeholder`	Trigger label when nothing is selected.
`check_style`	One of `"checkbox"` (default), `"check-only"`, or `"filled"`.
`show_style_switcher`	Show the Check / Box / Fill switcher row inside the dropdown? Default `TRUE`. Set `FALSE` to lock the style silently.
`show_select_all`	Show the "Select all" row? Default `TRUE`.
`show_clear_all`	Show the "Clear all" footer link? Default `TRUE`.
`theme`	color theme. One of `"dark"` (default) or `"light"`, or a `glass_select_theme()` object for full custom control. You only need to supply the colors you want to change — everything else falls back to the dark preset.
`hues`	Optional named integer vector of HSL hue angles (0–360) for the `"filled"` style. Auto-assigned if `NULL`.

Value

An `htmltools::tagList` containing the trigger button, dropdown panel, and a scoped <style> block. Embed directly in any Shiny UI function. The widget registers two Shiny inputs: `input$<inputId>` (character vector of selected values) and `input$<inputId>_style` (active checkbox style string).

Examples

```
fruits <- c(Apple = "apple", Banana = "banana", Cherry = "cherry")

# Minimal
glassMultiSelect("f", fruits)
```

```
# Lock style, hide all chrome
glassMultiSelect("f", fruits,
  check_style         = "check-only",
  show_style_switcher = FALSE,
  show_select_all     = FALSE,
  show_clear_all      = FALSE
)

# Only tweak the accent color - rest stays dark
glassMultiSelect("f", fruits,
  theme = glass_select_theme(accent_color = "#f59e0b")
)

# Light panel
glassMultiSelect("f", fruits, theme = "light")

# Full custom via glass_select_theme()
glassMultiSelect("f", fruits,
  theme = glass_select_theme(
    bg_color     = "#1a0a2e",
    border_color = "#a855f7",
    text_color   = "#ede9fe",
    accent_color = "#a855f7"
  )
)
```

---

glassMultiSelectServer

Server logic for glassMultiSelect

Description

A convenience wrapper that exposes the widget's current state as typed reactives. The underlying Shiny inputs are also available directly as `input$<inputId>` and `input$<inputId>_style`.

Usage

```
glassMultiSelectServer(inputId)
```

Arguments

`inputId`	The same `inputId` passed to `glassMultiSelect`.

Value

A list with two elements:

`selected` Reactive character vector of currently selected values.

`style` Reactive string — the active checkbox style (`"checkbox"`, `"check-only"`, or `"filled"`).

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassMultiSelect("cats", c(A = "a", B = "b", C = "c"))
  )
  server <- function(input, output, session) {
    ms <- glassMultiSelectServer("cats")
    observe(message("Selected: ", paste(ms$selected(), collapse = ", ")))
  }
  shinyApp(ui, server)
}
```

`glassTabPanel` *Define a single glass tab panel*

Description

Used as child arguments inside `glassTabsUI()`. Each call defines one tab button and its associated content pane.

Usage

```
glassTabPanel(value, label, ..., selected = FALSE)
```

Arguments

`value`	A unique string identifier for this tab (e.g. `"A"`).
`label`	The text shown on the tab button.
`...`	UI elements for the pane content.
`selected`	Logical. Whether this tab starts selected. Only the first `selected = TRUE` tab takes effect; defaults to `FALSE`.

Value

A list of class `"glassTabPanel"` consumed by `glassTabsUI()`.

Examples

```
glassTabPanel("overview", "Overview",
  shiny::h3("Welcome"),
  shiny::p("This is the overview tab.")
)
```

glassTabsServer *Server logic for glass tabs*

Description

Tracks the active tab and exposes it as a reactive value.

Usage

```
glassTabsServer(id)
```

Arguments

id	Module id matching the `id` passed to `glassTabsUI()`.

Value

A reactive expression returning the active tab value.

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassTabsUI(
      "tabs",
      glassTabPanel("a", "A", p("Tab A"), selected = TRUE),
      glassTabPanel("b", "B", p("Tab B"))
    )
  )
  server <- function(input, output, session) {
    active <- glassTabsServer("tabs")
    observe(print(active()))
  }
  shinyApp(ui, server)
}
```

glassTabsUI *Animated glass-style tab navigation UI*

Description

Animated glass-style tab navigation UI

Usage

```
glassTabsUI(
  id,
  ...,
  selected = NULL,
  wrap = TRUE,
  extra_ui = NULL,
  theme = NULL
)
```

Arguments

`id`	Module namespace id.
`...`	One or more `glassTabPanel()` objects.
`selected`	Value of the initially selected tab.
`wrap`	Logical. When `TRUE` wraps everything in a `div.gt-container`.
`extra_ui`	Optional additional UI placed to the right of the tab bar.
`theme`	One of `"dark"`, `"light"`, or a `glass_tab_theme()` object.

Value

An `htmltools::tagList` ready to use in a Shiny UI.

`glass_select_theme` *Create a custom color theme for glassMultiSelect*

Description

Create a custom color theme for glassMultiSelect

Usage

```
glass_select_theme(
  bg_color = NULL,
  border_color = NULL,
  text_color = NULL,
  accent_color = NULL
)
```

Arguments

`bg_color`	Background color of the trigger button and dropdown panel.
`border_color`	Border color.
`text_color`	Main text color.
`accent_color`	Accent color used for the animated tick, badge, checked-state highlights, and the "Clear all" link.

Value

A named list of class "`glass_select_theme`".

`glass_tab_theme` *Create a custom color theme for glassTabsUI*

Description

Create a custom color theme for glassTabsUI

Usage

```
glass_tab_theme(
  tab_text = NULL,
  tab_active_text = NULL,
  halo_bg = NULL,
  halo_border = NULL,
  content_bg = NULL,
  content_border = NULL,
  card_bg = NULL,
  card_text = NULL
)
```

Arguments

`tab_text`	Inactive tab text color.
`tab_active_text`	Active tab text color.
`halo_bg`	Halo background.
`halo_border`	Halo border.
`content_bg`	Tab content background.
`content_border`	Tab content border.
`card_bg`	Inner card background.
`card_text`	Inner card text color.

Value

A named list of class "`glass_tab_theme`".

`useGlassTabs` *Attach glasstabs CSS and JS dependencies*

Description

Call this once in your UI — either inside `fluidPage()`, `bs4DashPage()`, or any other Shiny page wrapper. It injects the required CSS and JS as proper `htmltools` dependencies so they are deduplicated automatically.

Usage

```
useGlassTabs()
```

Value

An `htmltools::htmlDependency` object (invisible to the user, consumed by Shiny's renderer).

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassTabsUI("demo",
      glassTabPanel("A", "Tab A", p("Content A")),
      glassTabPanel("B", "Tab B", p("Content B"))
    )
  )
  server <- function(input, output, session) {}
  shinyApp(ui, server)
}
```

Index