Package 'glasstabs'

April 11, 2026

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

Version 0.2.1

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), mockery

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 2026-04-11 18:40:07 UTC

Contents

appendGlassTab . . . 2
glassFilterTags . . . 3
glassMultiSelect . . . 4
glassMultiSelectValue . . . 5
glassSelect . . . 6
glassSelectValue . . . 8
glassTabPanel . . . 9
glassTabsServer . . . 10
glassTabsUI . . . 10
glass_select_theme . . . 11
glass_tab_theme . . . 12
showGlassTab . . . 13
updateGlassMultiSelect . . . 14
updateGlassSelect . . . 15
updateGlassTabsUI . . . 16
useGlassTabs . . . 17

Index **18**

`appendGlassTab` *Append or remove a glass tab at runtime*

Description

`appendGlassTab()` adds a new `glassTabPanel()` to an existing `glassTabsUI()` at runtime. `removeGlassTab()` removes a tab by value. If the removed tab was active, the first remaining tab is activated.

Usage

```
appendGlassTab(session, id, tab, select = FALSE)

removeGlassTab(session, id, value)
```

Arguments

`session`	Shiny session object.
`id`	Module id matching the `id` passed to `glassTabsUI()`.
`tab`	A `glassTabPanel()` object (for `appendGlassTab()` only).
`select`	Logical. If `TRUE`, the new tab is immediately activated. Defaults to `FALSE`.
`value`	Value of the tab to remove (for `removeGlassTab()` only).

Value

Called for its side effect; returns `NULL` invisibly.

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassTabsUI(
      "tabs",
      glassTabPanel("home", "Home", p("Home content"), selected = TRUE)
    ),
    actionButton("add",    "Add tab"),
    actionButton("remove", "Remove tab")
  )
  server <- function(input, output, session) {
    observeEvent(input$add, {
      appendGlassTab(session, "tabs",
        glassTabPanel("new", "New Tab", p("Dynamic content")),
        select = TRUE
      )
    })
    observeEvent(input$remove, {
      removeGlassTab(session, "tabs", "new")
    })
  }
  shinyApp(ui, server)
}
```

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

Description

Renders a 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

A stylized multi-select Shiny input with optional search, style switching, select-all behavior, and programmatic updates via `updateGlassMultiSelect()`.

Usage

```
glassMultiSelect(
  inputId,
  choices,
  selected = NULL,
  label = NULL,
  placeholder = "Filter by Category",
  all_label = "All categories",
  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.
`choices`	Named or unnamed character vector of choices.
`selected`	Initially selected values. Defaults to all choices when `NULL`.
`label`	Optional field label shown above the widget.
`placeholder`	Trigger label when nothing is selected.
`all_label`	Label shown when all choices are 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`.
`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.
`hues`	Optional named integer vector of HSL hue angles (0 to 360) for the `"filled"` style. Auto-assigned if `NULL`.

Details

The widget registers two Shiny inputs:

- `input$<inputId>` : character vector of selected values
- `input$<inputId>_style` : active style string (`"checkbox"`, `"check-only"`, or `"filled"`)

By default, when `selected = NULL`, all choices are initially selected. This preserves the existing package behavior.

Value

An `htmltools::tagList` containing the trigger button, dropdown panel, and scoped `<style>` block.

Examples

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

# Minimal
glassMultiSelect("f", fruits)

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

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

---

glassMultiSelectValue *Reactive helpers for glassMultiSelect values*

Description

Convenience helper for extracting a multi-select widget's value and style from Shiny's `input` object without using modules.

Usage

```
glassMultiSelectValue(input, inputId)
```

Arguments

input	Shiny input object.
inputId	Input id used in glassMultiSelect().

Value

A named list with two reactives:

selected Reactive character vector of selected values

style Reactive string for the active style

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    useGlassTabs(),
    glassMultiSelect("cats", c(A = "a", B = "b", C = "c"))
  )

  server <- function(input, output, session) {
    ms <- glassMultiSelectValue(input, "cats")
    observe({
      message("Selected: ", paste(ms$selected(), collapse = ", "))
      message("Style: ", ms$style())
    })
  }

  shinyApp(ui, server)
}
```

---

glassSelect *Animated glass single-select dropdown*

Description

A stylized single-select Shiny input with optional search, clear control, selection-marker styling, and programmatic updates via updateGlassSelect().

Usage

```
glassSelect(
  inputId,
  choices,
  selected = NULL,
  label = NULL,
```

```
  placeholder = "Select an option",
  searchable = TRUE,
  clearable = FALSE,
  include_all = FALSE,
  all_choice_label = "All categories",
  all_choice_value = "__all__",
  check_style = c("checkbox", "check-only", "filled"),
  theme = "dark"
)
```

Arguments

inputId	Shiny input id.
choices	Named or unnamed character vector of choices.
selected	Initially selected value. Defaults to NULL.
label	Optional field label shown above the widget.
placeholder	Trigger label when nothing is selected.
searchable	Logical. Show search input inside dropdown? Default TRUE.
clearable	Logical. Show clear control for removing the current selection? Default FALSE.
include_all	Logical. Prepend an explicit "All" option. Default FALSE.
all_choice_label	Label used for the explicit "All" option.
all_choice_value	Value used for the explicit "All" option.
check_style	One of "checkbox" (default), "check-only", or "filled".
theme	Color theme. One of "dark" (default) or "light", or a glass_select_theme() object.

Details

The widget registers one Shiny input:

- input$<inputId> : selected value as a length-1 character string, or NULL when nothing is selected

Value

An htmltools::tagList containing the single-select trigger, dropdown panel, and scoped <style> block.

Examples

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

glassSelect("fruit", fruits)

glassSelect(
```

```
  "fruit",
  fruits,
  selected = "banana",
  clearable = TRUE
)

glassSelect(
  "fruit",
  fruits,
  include_all = TRUE,
  all_choice_label = "All fruits",
  all_choice_value = "__all__"
)

glassSelect(
  "fruit",
  fruits,
  check_style = "filled"
)
```

`glassSelectValue` *Reactive helper for glassSelect values*

Description

Convenience helper for extracting a single-select widget's value from Shiny's `input` object without using modules.

Usage

```
glassSelectValue(input, inputId)
```

Arguments

`input`	Shiny `input` object.
`inputId`	Input id used in `glassSelect()`.

Value

A reactive expression returning the current selected value as a character scalar, or `NULL` when nothing is selected.

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
```

```
    useGlassTabs(),
    glassSelect("fruit", c(Apple = "apple", Banana = "banana"))
  )

  server <- function(input, output, session) {
    fruit <- glassSelectValue(input, "fruit")
    observe({
      print(fruit())
    })
  }

  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 glass select widgets*

Description

Create a custom color theme for glass select widgets

Usage

```
glass_select_theme(
  mode = c("dark", "light"),
  bg_color = NULL,
  border_color = NULL,
  text_color = NULL,
  accent_color = NULL,
  label_color = NULL
)
```

Arguments

mode	Base theme preset. One of "dark" (default) or "light". Custom colors are applied on top of this base mode.
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 clear controls.
label_color	Optional label color. If NULL, the label defaults to text_color.

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".

---

showGlassTab *Show or hide a glass tab*

Description

showGlassTab() makes a hidden tab visible again. hideGlassTab() hides a tab from the navigation bar. If the hidden tab is currently active, the first remaining visible tab is activated automatically.

Usage

```
showGlassTab(session, id, value)

hideGlassTab(session, id, value)
```

Arguments

session	Shiny session object.
id	Module id matching the id passed to glassTabsUI().
value	Value of the tab to show or hide.

Value

Called for its side effect; returns NULL invisibly.

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassTabsUI(
      "tabs",
      glassTabPanel("a", "A", p("Tab A"), selected = TRUE),
      glassTabPanel("b", "B", p("Tab B")),
      glassTabPanel("admin", "Admin", p("Admin only"))
    ),
    checkboxInput("is_admin", "Admin mode", FALSE)
  )
  server <- function(input, output, session) {
    observeEvent(input$is_admin, {
      if (input$is_admin) showGlassTab(session, "tabs", "admin")
      else                hideGlassTab(session, "tabs", "admin")
    }, ignoreInit = FALSE)
  }
  shinyApp(ui, server)
}
```

`updateGlassMultiSelect` *Update a glassMultiSelect widget*

Description

Update the available choices and/or current selection of an existing `glassMultiSelect()` input.

Usage

```
updateGlassMultiSelect(
  session,
  inputId,
  choices = NULL,
  selected = NULL,
  check_style = NULL
)
```

Arguments

`session`	Shiny session.
`inputId`	Input id of the widget.
`choices`	New choices, or `NULL` to keep current choices.
`selected`	New selected values, or `NULL` to keep current selection. Use `character(0)` to clear.
`check_style`	Optional new style string. One of `"checkbox"`, `"check-only"`, or `"filled"`. Defaults to `NULL`, which keeps the current style unchanged.

Details

This function now follows Shiny-style update semantics more closely:

- `choices = NULL` leaves choices unchanged
- `selected = NULL` leaves selection unchanged
- `selected = character(0)` clears the selection

When `choices` is supplied and `selected` is not, the browser side keeps the intersection of the current selection and the new set of choices.

`updateGlassSelect` *Update a glassSelect widget*

Description

Update the available choices, current selection, and/or selection-marker style of an existing `glassSelect()` input.

Usage

```
updateGlassSelect(
  session,
  inputId,
  choices = NULL,
  selected = NULL,
  check_style = NULL
)
```

Arguments

`session`	Shiny session.
`inputId`	Input id of the widget.
`choices`	New choices, or `NULL` to keep current choices.
`selected`	New selected value, or `NULL` to keep the current selection. Use `character(0)` to clear.
`check_style`	Optional new style string. One of `"checkbox"`, `"check-only"`, or `"filled"`. Defaults to `NULL`, which keeps the current style unchanged.

Details

This function follows Shiny-style update semantics:

- `choices` = `NULL` leaves choices unchanged
- `selected` = `NULL` leaves selection unchanged
- `selected` = `character(0)` clears the selection
- `check_style` = `NULL` leaves the current style unchanged

When `choices` is supplied and `selected` is not, the browser side keeps the current selection if it is still present in the new choices.

Value

No return value. Called for its side effect of updating the client-side widget.

`updateGlassTabsUI` *Programmatically switch the active glass tab*

Description

Server-side equivalent of Shiny's `updateTabsetPanel()`. Sends a message to the browser to animate the tab switch just as if the user had clicked the tab button.

Usage

```
updateGlassTabsUI(session, id, selected)
```

Arguments

`session`	Shiny session object.
`id`	Module id matching the `id` passed to `glassTabsUI()`.
`selected`	Value of the tab to activate.

Value

Called for its side effect; returns `NULL` invisibly.

Examples

```
if (interactive()) {
  library(shiny)
  ui <- fluidPage(
    useGlassTabs(),
    glassTabsUI(
      "tabs",
      glassTabPanel("a", "A", p("Tab A"), selected = TRUE),
      glassTabPanel("b", "B", p("Tab B"))
    ),
    actionButton("go", "Go to B")
  )
  server <- function(input, output, session) {
    observeEvent(input$go, {
      updateGlassTabsUI(session, "tabs", selected = "b")
    })
  }
  shinyApp(ui, server)
}
```

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

appendGlassTab, 2

glass_select_theme, 11
glass_select_theme(), *4*, *7*
glass_tab_theme, 12
glass_tab_theme(), *11*
glassFilterTags, 3
glassMultiSelect, 4
glassMultiSelect(), *3*, *6*, *14*
glassMultiSelectValue, 5
glassSelect, 6
glassSelect(), *8*, *15*
glassSelectValue, 8
glassTabPanel, 9
glassTabPanel(), *2*, *11*
glassTabsServer, 10
glassTabsUI, 10
glassTabsUI(), *2*, *9*, *10*, *13*, *16*

hideGlassTab *(*showGlassTab*)*, 13

removeGlassTab *(*appendGlassTab*)*, 2

showGlassTab, 13

updateGlassMultiSelect, 14
updateGlassMultiSelect(), *4*
updateGlassSelect, 15
updateGlassSelect(), *6*
updateGlassTabsUI, 16
useGlassTabs, 17