

glasstabs

tabs and layout

v0.4.1

page 1 of 3

Setup

Load the dependency once in the UI:

```
library(shiny)
library(glasstabs)

ui <- fluidPage(
  useGlassTabs(),
  # glasstabs widgets
)
```

Explore package examples and release notes:

```
runClassExample() # list apps
runClassExample("light-mode")
runClassExample("native-comparison")
runClassExample("connect-workflow")
runClassExample("stress-test") # dropdown limits: 100k choices,
  hostile labels
runClassExample("theme-switcher")
glasstabs_news()
```

Experimental page wrapper

`glassPage()` is a thin optional wrapper around `bslib::page_fillable()` and loads `useGlassTabs()`.

```
ui <- glassPage(
  title = "Review",
  lang = "en",
  theme = bslib::bs_theme(version = 5),
  padding = "1rem",
  gap = "1rem",
  fillable_mobile = FALSE,
  h2("Review queue")
)
```

Requires the suggested `bslib` package.

Tab widget – UI

```
glassTabsUI(
  id, ...,
  selected = NULL,
  wrap = TRUE,
  compact = FALSE,
  content_min_height = NULL,
  style = c("floating", "attached"),
  transition = c("fade", "slide"),
  shape = c("rounded", "square"),
  indicator = c("glass", "solid",
    "underline"),
  orientation = c("horizontal",
    "vertical"),
  tab_align = c("center", "left", "right"),
  text_align = c("center", "left", "right"),
  overflow = c("scroll", "multiline", "menu"),
  swipe = FALSE,
  extra_ui = NULL,
  theme = NULL,
  dark_selector = NULL
)

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

Small complete pattern:

```
glassTabsUI(
  "main",
  glassTabPanel("queue", "Queue",
    selected = TRUE, p("Open work")),
  glassTabPanel("done", "Done",
    icon = icon("check"), p("Finished")),
  selected = "queue",
  overflow = "scroll",
  theme = "auto"
)
```

Layout and responsive choices

- **orientation**: horizontal row or vertical rail.
- **tab_align**: places the tab group.
- **text_align**: aligns labels inside buttons.
- **indicator**: glass halo, solid fill, or underline.
- **shape**: rounded or square tabs.
- **overflow = "scroll"**: one horizontal row; selected tab stays visible.
- **overflow = "multiline"**: tabs wrap onto rows.
- **overflow = "menu"**: native compact menu; horizontal only.
- **swipe = TRUE**: guarded horizontal pane gestures.
- **compact = TRUE**: tighter dashboard-card spacing.

```
glassTabsUI(
  "mobile", ...,
  overflow = "menu",
  swipe = TRUE,
  text_align = "left"
)
```

Keyboard and accessibility

Built in:

- Arrow keys move through tabs; orientation chooses the axis.
- Home and End move to the first and last available tab.
- Enter or Space activates the focused tab.
- Disabled tabs remain discoverable but cannot activate.
- Inactive panes are inert and outside the focus order.
- Reduced-motion and forced-colour preferences are respected.

Read the active tab

```
server <- function(input, output, session) {
  active <- glassTabsServer(
    "main",
    bookmark = TRUE
  )
  observe(print(active()))

  # Same underlying value:
  input[["main-active_tab"]]
}
```

Modules: use `ns("tabs")` in the UI and bare "tabs" in `glassTabsServer()`.

Tab server actions

```
updateGlassTabsUI(session, "main", "done")

showGlassTab(session, "main", "admin")
hideGlassTab(session, "main", "admin")

disableGlassTab(session, "main", "done")
enableGlassTab(session, "main", "done")

updateGlassTabBadge(
  session, "main", "queue", count = 5L
) # count 0 hides the badge

appendGlassTab(
  session, "main",
  glassTabPanel("new", "New", p("...")),
  select = TRUE
)

removeGlassTab(session, "main", "new")
```

Dynamic tabs

```
# UI
glassTabsOutput("dynamic")

# Server
output$dynamic <- renderGlassTabs({
  glassTabsUI(
    "generated",
    glassTabPanel("a", "A", p("A")),
    glassTabPanel("b", "B", p("B"))
  )
})
```

Browser behavior reinitialises after each render.

Conditional content

```
conditionalPanel(
  condition =
    glassTabCondition("main", "done"),
  p("Visible on Done")
)

# In a module, use the UI id:
glassTabCondition(ns("tabs"), "done")
```

URL bookmarking

```
shinyApp(
  ui, server,
  enableBookmarking = "url"
)

# Opt one widget out:
glassTabsServer("main", bookmark = FALSE)
```

Tab themes

Presets are "dark", "light", and "auto". Auto follows Bootstrap 5 colour mode.

```
glassTabsUI(
  "main", ...,
  theme = glass_tab_theme(
    halo_bg = "rgba(37,99,235,.14)",
    tab_active_text = "#1d4ed8",
    focus_ring = "#1d4ed8"
  )
)
```

`glass_tab_theme()` handles:

- **tab_text**, **tab_active_text**
- **halo_bg**, **halo_border**, **focus_ring**
- **content_bg**, **content_border**
- **card_bg**, **card_text**
- **mode** records the dark or light theme mode

Multi-select – UI

```
glassMultiSelect(  
  inputId, choices,  
  selected = NULL,  
  label = NULL,  
  placeholder = "Filter by Category",  
  all_label = "All categories",  
  check_style = c("checkbox", "filled"),  
    "check-only", "filled"),  
  show_style_switcher = TRUE,  
  show_select_all = TRUE,  
  show_clear_all = TRUE,  
  theme = "dark",  
  shape = c("rounded", "square"),  
  width = NULL,  
  disabled = FALSE,  
  disabled_choices = NULL,  
  hues = NULL,  
  dark_selector = NULL,  
  server = FALSE,  
  server_limit = 50L,  
  server_min_chars = 0L,  
  searchable = TRUE,  
  search_threshold = 15L,  
  selection_display = c("auto", "count",  
    "summary", "labels"),  
  selection_max_items = 2L,  
  dropdown_max_height = "260px",  
  no_matches_text = "No matches",  
  tongue = TRUE,  
  label_bold = TRUE,  
  fill_colors = c("single", "multi"),  
  label_size = "14px",  
  search_focus_ring = FALSE,  
  overflow_tooltip = c("auto", "never")  
)  
  
metrics <- c(  
  Revenue = "rev",  
  Orders = "ord",  
  Returns = "ret"  
)  
  
glassMultiSelect(  
  "metric", metrics,  
  selected = "rev",  
  label = "Metrics",  
  searchable = "auto",  
  selection_display = "summary",  
  theme = "auto"  
)  
  
glassFilterTags("metric")
```

Single-select – UI

```
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",  
  shape = c("rounded", "square"),  
  width = NULL,  
  disabled = FALSE,  
  disabled_choices = NULL,  
  server = FALSE,  
  server_limit = 50L,  
  server_min_chars = 0L,  
  no_matches_text = "No matches",  
  tongue = TRUE,  
  label_bold = TRUE,  
  label_size = "14px",  
  search_focus_ring = FALSE,  
  overflow_tooltip = c("auto", "never"),  
  search_threshold = 15L,  
  dark_selector = NULL  
)  
  
glassSelect(  
  "region",  
  c(North = "n", South = "s"),  
  selected = "n",  
  clearable = TRUE,  
  shape = "square"  
)
```

Server-side search

Use for large choice collections. The UI and server IDs match.

```
# UI  
glassMultiSelect(  
  "people", character(),  
  server = TRUE,  
  server_limit = 40L,  
  server_min_chars = 2L  
)  
  
glassSelect(  
  "city", character(),  
  server = TRUE  
)  
  
# Server  
glassMultiSelectServer(  
  "people", people_choices,  
  session = session,  
  limit = 40L,  
  ignore_case = TRUE  
)  
  
glassSelectServer(  
  "city", city_choices,  
  session = session  
)
```

Read select values

```
multi <- glassMultiSelectValue(  
  input, "metric",  
  choices = metrics,  
  empty_behavior = "all"  
)  
  
multi$selected()  
multi$resolved()  
multi$is_empty()  
multi$style()  
  
single <- glassSelectValue(  
  input, "region"  
)  
single()
```

State	Shiny input
tab value	input[["id-active_tab"]]
multi values	input\$metric
multi style	input\$metric_style
single value	input\$region
open state	input\$metric_open / input\$region_open

Update select widgets

```
updateGlassMultiSelect(  
  session, "metric",  
  choices = metrics,  
  selected = c("rev", "ord"),  
  check_style = "filled",  
  shape = "square",  
  disabled = FALSE,  
  disabled_choices = "ret",  
  preserve_selection = TRUE,  
  drop_invalid = TRUE,  
  notify = "changed",  
  searchable = "auto",  
  search_threshold = 15L,  
  selection_display = "summary",  
  selection_max_items = 2L,  
  dropdown_max_height = "18rem"  
)  
  
updateGlassSelect(  
  session, "region",  
  choices = c(North="n", South="s"),  
  selected = "s",  
  check_style = "check-only",  
  shape = "rounded",  
  disabled = FALSE,  
  disabled_choices = "n",  
  searchable = "auto",  
  search_threshold = 15L  
)
```

Close dropdowns

```
closeGlassSelect(session, "region")
closeGlassMultiSelect(session, "metric")
closeAllGlassSelects(session)
```

All default to the current reactive domain when `session` is omitted.

Select themes

```
theme <- glass_select_theme(
  mode = "dark",
  bg_color = "rgba(9,20,42,.97)",
  border_color = "rgba(255,255,255,.1)",
  text_color = "#cfe6ff",
  accent_color = "#38bdf8",
  focus_ring = "#7dd3fc",
  label_color = "#cfe6ff"
)

glassSelect("region", choices,
  theme = theme)
```

Both select widgets accept `"dark"`, `"light"`, `"auto"`, or a custom theme object.

Function map – all exports

All 32 exported functions, grouped by job:

```
# Setup / page
useGlassTabs  glassPage
runGlassExample  glasstabs_news

# Tab construction / state
glassTabsUI  glassTabPanel
glassTabsServer  glassTabCondition
glassTabsOutput  renderGlassTabs

# Tab updates
updateGlassTabsUI  updateGlassTabBadge
showGlassTab  hideGlassTab
disableGlassTab  enableGlassTab
appendGlassTab  removeGlassTab

# Multi-select
glassMultiSelect  glassMultiSelectServer
updateGlassMultiSelect
glassMultiSelectValue  glassFilterTags
closeGlassMultiSelect

# Single-select
glassSelect  glassSelectServer
updateGlassSelect  glassSelectValue
closeGlassSelect  closeAllGlassSelects

# Themes
glass_tab_theme  glass_select_theme
```

Common gotchas

- Call `useGlassTabs()` once unless using `glassPage()`.
- Panel `values` must be unique; `selected` uses values, not labels.
- Modules use `ns("tabs")` in UI and bare `"tabs"` in server.
- `overflow = "menu"` supports horizontal tabs only.
- Swipe ignores interactive and scrollable content.
- `glassSelectValue()` returns one reactive function.
- `glassMultiSelectValue()` returns raw `selected()`, resolved values, empty state, and `style()` reactives.
- Server-side choice functions belong in the server.
- Select dropdowns open attached to the field (no gap): the panel unfurls from the trigger edge with squared adjoining corners, flipping above the field when space is tight below.
- Set `tongue = FALSE` for a detached panel with a short fade. Use `overflow_tooltip = "never"` to suppress long-label tips.

Useful links

- Documentation and articles
- Live Posit Connect app
- Source and issue tracker