Package 'shinyglass'

August 21, 2026

Type Package

Title Liquid Glass Design Themes for 'shiny' Applications

Version 0.2.0

Description Provides drop-in Liquid Glass themes for 'shiny'. Call glass_theme() and pass the result as theme = to fluidPage(), navbarPage(), or any 'bslib'-aware page function to get translucent surfaces, backdrop blur, and system typography on 'Bootstrap' components. Includes light and dark presets with runtime switching and an OS-following 'auto' mode, an iOS-style intensity control from Ultra Clear to Tinted (glass_intensity_slider()), and options for accent color, blur, corner radius, and motion or tint behavior.

License GPL-3

URL https://ericrayanderson.github.io/shinyglass/, https://github.com/ericrayanderson/shinyglass

BugReports https://github.com/ericrayanderson/shinyglass/issues

Encoding UTF-8

Language en-US

VignetteBuilder knitr

Imports bslib (>= 0.5.0), htmltools (>= 0.5.0), sass (>= 0.4.0), shiny (>= 1.5.0)

Suggests DT, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website pkgdown

Config/roxygen2/markdown TRUE

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Eric Anderson [aut, cre, cph]

Maintainer Eric Anderson <eric.ray.anderson@gmail.com>

Repository CRAN

Date/Publication 2026-08-21 05:40:42 UTC

Contents

`glass_intensity_slider` *Liquid Glass intensity slider (iOS 27-style)*

Description

UI control matching iOS 27 **Settings -> Appearance -> Liquid Glass**: a continuous slider from **Ultra Clear** (`0`) to **Tinted** (1) that live-updates the glass material without a page reload.

Usage

```
glass_intensity_slider(
  inputId = "glass_intensity",
  label = "Liquid Glass",
  value = NULL,
  min = 0,
  max = 1,
  step = 0.01,
  min_label = "Ultra Clear",
  max_label = "Tinted",
  preview = TRUE,
  width = NULL
)
```

Arguments

`inputId`	The `input` slot that will be used to access the value.
`label`	Display label for the control (or `NULL` for none).
`value`	Initial intensity in $[0, 1]$. If `NULL`, uses the theme default from `glass_theme()` / the client current intensity.
`min`, `max`, `step`	Range for the underlying range input. Defaults cover the full Ultra Clear -> Tinted spectrum.

`min_label`, `max_label`
: End-cap captions (iOS uses "Ultra Clear" / "Tinted").

`preview`
: Show three mini glass chips as a live material sample.

`width`
: CSS width (passed to `shiny::validateCssUnit()`).

Details

The client applies changes immediately via `window.shinyglass.setIntensity()`. The value is also a normal Shiny input (`input[[inputId]]`) so the server can react or persist it. Pair with `glass_theme()` `intensity=` for the starting value, and optionally `observe_glass_intensity()` to push server-driven updates through `update_glass_theme()`.

Value

A Shiny UI tag hierarchy.

Examples

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

  ui <- fluidPage(
    theme = glass_theme(intensity = 0.35),
    glass_theme_toggle(),
    glass_intensity_slider("glass_intensity"),
    plotOutput("p")
  )

  server <- function(input, output, session) {
    observe_glass_theme_toggle(input, session)
    # optional: mirror slider -> server message (client already updates live)
    observe_glass_intensity(input, session, "glass_intensity")
    output$p <- renderPlot(plot(rnorm(100), rnorm(100), pch = 16, col = "#007AFF"))
  }

  shinyApp(ui, server)
}
```

---

glass_preset_input *Light / dark / auto preset select input*

Description

Drop-in `shiny::selectInput()` for the glass theme preset. The client applies the change **immediately** via `window.shinyglass.setPreset()` (marked with `data-glass-preset-input`), so Light/Dark/Auto works even when custom messages are delayed or rewritten (e.g. on some shinyapps.io hosts). Pair with `observe_glass_preset_input()` so the server stays in sync.

Usage

```
glass_preset_input(
  inputId = "glass_preset",
  label = "Theme preset",
  selected = c("auto", "light", "dark"),
  choices = c(Light = "light", Dark = "dark", `Auto (OS)` = "auto"),
  width = NULL
)
```

Arguments

inputId	The input slot that will be used to access the value.
label	Display label for the control (or NULL for none).
selected	Initially selected mode ("light", "dark", or "auto").
choices	Named character vector of choices. Defaults to Light / Dark / Auto (OS). Names are labels; values must be light, dark, and/or auto.
width	The width of the input (e.g. "100%").

Value

A shiny::selectInput() tag (with glass client bindings).

See Also

observe_glass_preset_input(), glass_theme_toggle(), update_glass_theme()

---

glass_resolved_preset *Resolved light or dark appearance*

Description

When the user picks **Auto**, input$preset stays "auto" so checks like identical(input$preset, "dark") stay FALSE and plots keep light ink on a dark page. The client publishes the *resolved* appearance as input$glass_resolved_preset ("light" or "dark").

Usage

```
glass_resolved_preset(input, default = c("light", "dark"))
```

Arguments

input	The server input object.
default	Fallback if the client has not reported yet ("light" or "dark").

Value

"light" or "dark".

glass_theme *Liquid Glass theme for 'shiny'*

Description

Create a `bslib::bs_theme()` styled with a Liquid Glass look: translucent surfaces, backdrop blur, soft depth, and system typography. Pass the result to `theme =` on `fluidPage()`, `navbarPage()`, `bslib::page_sidebar()`, or any other page function that accepts a bslib theme.

Usage

```
glass_theme(
  preset = c("light", "dark", "auto"),
  primary = "#007AFF",
  blur = 36,
  saturation = 200,
  radius = "1.5rem",
  material = c("regular", "clear"),
  intensity = 0.45,
  tint = TRUE,
  specular = TRUE,
  nav_morph = TRUE,
  ...
)
```

Arguments

`preset`	`"light"`, `"dark"`, or `"auto"`. `"auto"` follows `prefers-color-scheme` and updates when the OS theme changes.
`primary`	Accent color for buttons, links, and focus rings. Defaults to system blue (`#007AFF`).
`blur`	Backdrop blur radius in pixels. Default `36` matches the iOS 27 diffusion-first material.
`saturation`	Backdrop saturation percentage.
`radius`	Default border radius for glass surfaces (CSS length). Prefer larger concentric radii (default `1.5rem`).
`material`	`"regular"` (adaptive, most UI) or `"clear"` (more transparent; best over media-rich content with bold labels).
`intensity`	Liquid Glass intensity from `0` (Ultra Clear) to `1` (Tinted), matching iOS 27 Appearance -> Liquid Glass. Default `0.45`. Use `glass_intensity_slider()` for a live control.
`tint`	Content-aware ambient tint from plots/images (JS).
`specular`	Pointer-driven specular highlight on glass surfaces (JS).
`nav_morph`	Compact navbar on scroll down; expand on scroll up (JS).
`...`	Additional arguments forwarded to `bslib::bs_theme()`.

Details

Light and dark surface tokens are compiled into dual CSS custom-property packs. Switching `preset` at runtime (via `update_glass_theme()` or `preset = "auto"`) updates `document.documentElement.dataset.glas` without recompiling Sass or reloading the page. Accent color can also be updated live with `update_glass_theme()` `primary=` (CSS variables).

Value

A `bslib::bs_theme()` object suitable for 'shiny' page functions.

Examples

```
theme <- glass_theme()
dark <- glass_theme(preset = "dark", primary = "#BF5AF2")
auto <- glass_theme(preset = "auto", tint = FALSE)
clear <- glass_theme(material = "clear")

if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    theme = glass_theme(preset = "auto"),
    titlePanel("Liquid Glass"),
    glass_theme_toggle(),
    selectInput("color", "Color", c("Blue", "Purple", "Orange")),
    plotOutput("plot")
  )

  server <- function(input, output, session) {
    # Client onclick already switches; keep session in sync:
    observe_glass_theme_toggle(input, session)
  }

  shinyApp(ui, server)
}
```

`glass_theme_toggle` *Light / dark / auto theme toggle buttons*

Description

Drop-in button group for switching the glass preset. Each button sets the preset on the client immediately (`window.shinyglass.setPreset`) and also has a 'shiny' input id so `observe_glass_theme_toggle()` can keep the server in sync (important on hosts that rewrite custom messages).

Usage

```
glass_theme_toggle(
  inputId = "glass_toggle",
  selected = c("auto", "light", "dark"),
  labels = c(light = "Light", dark = "Dark", auto = "Auto (OS)"),
  class = "glass-theme-toggle"
)
```

Arguments

inputId	Base id. Buttons are {inputId}_light, {inputId}_dark, and {inputId}_auto.
selected	Initially highlighted mode ("light", "dark", or "auto"). Cosmetic only; the page theme still comes from glass_theme().
labels	Named character vector for button labels. Names must be light, dark, and/or auto.
class	Extra CSS classes for the wrapper.

Value

An htmltools::tag() button group.

See Also

observe_glass_theme_toggle(), update_glass_theme()

---

observe_glass_intensity

Keep session intensity in sync with glass_intensity_slider()

Description

The slider already updates glass live on the client. This observer optionally echoes the value through update_glass_theme() so other clients / server state stay aligned.

Usage

```
observe_glass_intensity(input, session, inputId = "glass_intensity")
```

Arguments

input	The server input object.
session	A Shiny session object.
inputId	Input id of the intensity slider.

Value

An shiny::observeEvent() observer (invisibly).

observe_glass_preset_input

Observe `glass_preset_input()` *on the server*

Description

Wires a preset select to `update_glass_theme()` so session state stays aligned with the client (the client already applied the preset live).

Usage

```
observe_glass_preset_input(input, session, inputId = "glass_preset")
```

Arguments

`input`	The server `input` object.
`session`	The server `session` object.
`inputId`	Same id passed to `glass_preset_input()`.

Value

An `shiny::observeEvent()` observer (invisibly).

observe_glass_theme_toggle

Observe `glass_theme_toggle()` *buttons on the server*

Description

Wires the three toggle inputs to `update_glass_theme()` so session state stays aligned with the client (needed on some hosts that rewrite 'shiny' messaging).

Usage

```
observe_glass_theme_toggle(input, session, inputId = "glass_toggle")
```

Arguments

`input`	The server `input` object.
`session`	The server `session` object.
`inputId`	Same base id passed to `glass_theme_toggle()`.

Value

`NULL`, invisibly. Called for side effects (registers observers).

update_glass_theme *Update glass theme options in a running app*

Description

Send a message to the browser to change the Liquid Glass preset, accent color, intensity, or content-tint behavior without reloading the page. Requires a page that used `glass_theme()` (so `shiny-glass.js` is loaded).

Usage

```
update_glass_theme(
  session,
  preset = NULL,
  tint = NULL,
  primary = NULL,
  intensity = NULL
)
```

Arguments

`session`	A Shiny session object (usually the `session` argument of the server function).
`preset`	Optional. `"light"`, `"dark"`, or `"auto"`.
`tint`	Optional logical. Enable or disable content-aware ambient tint.
`primary`	Optional accent color (hex like `"#AF52DE"` or `rgb()`).
`intensity`	Optional numeric in $[0, 1]$: Ultra Clear (`0`) to Tinted (1).

Details

`primary` updates CSS variables (`--bs-primary`, `--bs-primary-rgb`, `--glass-primary`) so buttons, checks, and other accent surfaces follow the new color. Sass-baked one-off colors may not all switch until a full reload. `intensity` updates fill/blur along the Ultra Clear to Tinted spectrum (see `glass_intensity_slider()`).

Value

`session`, invisibly.

Examples

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

  ui <- fluidPage(
    theme = glass_theme(),
    glass_theme_toggle(),
```

```
    selectInput("accent", "Accent", c("#007AFF", "#AF52DE", "#FF9500"))
  )

  server <- function(input, output, session) {
    observe_glass_theme_toggle(input, session)
    observeEvent(input$accent, {
      update_glass_theme(session, primary = input$accent)
    }, ignoreInit = TRUE)
  }

  shinyApp(ui, server)
}
```

Index