Package 'shinyglass'

September 9, 2026

Type Package

Title Liquid Glass Design Themes for 'shiny' Applications

Version 0.3.0

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, optional persistence of the look, named wallpaper scenes, and helpers to match 'ggplot2', 'plotly', and 'gt' output to the glass pack.

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, gt, knitr, plotly, 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 2026-09-09 21:00:02 UTC

Contents

glass_accent_input 2
glass_intensity_slider 3
glass_page 4
glass_plot_colors 6
glass_preset_input 6
glass_resolved_preset 7
glass_system_colors 8
glass_theme 8
glass_theme_toggle 10
gt_theme_glass 11
observe_glass 11
observe_glass_accent 12
observe_glass_intensity 12
observe_glass_preset_input 13
observe_glass_theme_toggle 13
plotly_glass 14
theme_glass 14
update_glass_theme 15

Index **17**

`glass_accent_input` *Accent color wells*

Description

Compact row of color wells. The client applies `window.shinyglass.setPrimary()` immediately; the value is also a Shiny input (`input[[inputId]]`) so the server can persist or restyle plots. Pair with `observe_glass_accent()` if you also want `update_glass_theme()` to echo the change.

Usage

```
glass_accent_input(
  inputId = "glass_accent",
  label = "Accent",
  selected = "blue",
  colors = glass_system_colors()
)
```

Arguments

inputId	The input slot that will be used to access the hex value.
label	Display label (or NULL for none).
selected	Initially selected well: a name in colors or a hex string.
colors	Named character vector of hex colors. Defaults to glass_system_colors().

Value

A Shiny UI tag hierarchy.

---

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. Intensity is page-wide: the first explicit value in each inserted group wins, and all controls synchronize. A restricted range displays the nearest endpoint when the current theme intensity lies outside that range.
min, max, step	Range for the underlying range input. Defaults cover the full Ultra Clear -> Tinted spectrum. Bounds must be finite, within [0, 1], and increasing; step must be finite and positive.

`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_page *One-call glass page*

Description

Wraps `shiny::fluidPage()` with `glass_theme()` and, by default, the Light / Dark / Auto toggle, intensity slider, and accent wells. Pair with `observe_glass()` in the server function.

Usage

```
glass_page(
  ...,
  title = NULL,
  preset = "auto",
  intensity = 0.45,
  persist = TRUE,
  scene = "tahoe",
  wallpaper = NULL,
  controls = TRUE,
  theme = NULL
)
```

Arguments

`...`	UI elements passed to `shiny::fluidPage()`.
`title`	Optional page title (`shiny::titlePanel()`).
`preset, intensity, persist, scene, wallpaper`	Forwarded to `glass_theme()`. Persistence defaults to `TRUE` here.
`controls`	`TRUE` for the default control row (toggle, intensity, accent), `FALSE` for none, or a character vector subset of `"toggle"`, `"intensity"`, `"accent"`.
`theme`	Optional pre-built `glass_theme()` object. When `NULL`, one is created from the arguments above.

Value

A Shiny UI tag list suitable as `shinyApp(ui = ...)`.

Examples

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

  ui <- glass_page(
    title = "Hello, glass",
    plotOutput("plot")
  )
  server <- function(input, output, session) {
    observe_glass(input, session)
    output$plot <- renderPlot(plot(1:10), bg = "transparent")
  }
  shinyApp(ui, server)
}
```

glass_plot_colors *Ink and grid colors for the current glass appearance*

Description

Ink and grid colors for the current glass appearance

Usage

```
glass_plot_colors(preset = NULL, input = NULL)
```

Arguments

preset	"light" or "dark". When NULL, uses glass_resolved_preset() on input.
input	Optional Shiny input (used when preset is NULL).

Value

A list with preset, ink, grid, fill, and paper.

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_system_colors *iOS system accent colors*

Description

Named hex colors matching iOS Settings wells: blue, purple, pink, orange, green, and teal.

Usage

```
glass_system_colors()
```

Value

A named character vector of #RRGGBB colors.

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,
  ambient_motion = TRUE,
  persist = FALSE,
  scene = c("default", "tahoe", "dusk", "mesh"),
  wallpaper = NULL,
  ...
)
```

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).
ambient_motion	Animate the decorative ambient sheen. Set FALSE to keep static glass surfaces. OS reduced-motion settings take priority.
persist	Remember preset, intensity, accent, material, and scene in localStorage for this app path. Default FALSE (opt in). glass_page() turns this on.
scene	Wallpaper scene: "default", "tahoe", "dusk", or "mesh".
wallpaper	Optional image URL painted as a frosted photo behind the glass (https, data URI, or site-relative path).
...	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")
remembered <- glass_theme(persist = TRUE, scene = "tahoe")
```

```
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()

---

gt_theme_glass *gt table options that match glass chrome*

Description

Transparent table chrome so glass cards show through. Requires gt.

Usage

```
gt_theme_glass(data, preset = NULL, input = NULL)
```

Arguments

data	A gt::gt() table (or data frame, which is passed to gt()).
preset, input	See glass_plot_colors().

Value

A gt table.

---

observe_glass *Observe every built-in glass control*

Description

Registers observe_glass_theme_toggle(), observe_glass_intensity(), and observe_glass_accent() with the default input ids used by glass_page().

Usage

```
observe_glass(
  input,
  session,
  toggle_id = "glass_toggle",
  intensity_id = "glass_intensity",
  accent_id = "glass_accent"
)
```

Arguments

input, session	Shiny input and session objects.
toggle_id, intensity_id, accent_id	Input ids matching the UI controls.

Value

NULL, invisibly.

---

observe_glass_accent *Keep session accent in sync with* glass_accent_input()

Description

The wells already update glass live on the client. This observer echoes the hex through update_glass_theme() so plots and other sessions can follow.

Usage

```
observe_glass_accent(input, session, inputId = "glass_accent")
```

Arguments

input	The server input object.
session	A Shiny session object.
inputId	Input id of the accent wells.

Value

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

---

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).

---

plotly_glass *plotly layout that matches glass chrome*

Description

Transparent paper/plot backgrounds and ink that follows the resolved preset. Requires plotly. Call as plotly_glass(p) or do.call(plotly::layout, c(list(p), plotly_glass())).

Usage

```
plotly_glass(p = NULL, preset = NULL, input = NULL)
```

Arguments

p	Optional plotly object. When NULL, returns a named list of layout arguments.
preset, input	See glass_plot_colors().

Value

p with layout applied, or a list of layout arguments.

---

theme_glass *ggplot2 theme that follows glass light/dark ink*

Description

Transparent panel and plot backgrounds so the page wallpaper shows through glass cards. Requires ggplot2 (a Suggests dependency).

Usage

```
theme_glass(preset = NULL, base_size = 13, input = NULL, ...)
```

Arguments

preset	"light" or "dark". When NULL, uses glass_resolved_preset().
base_size	Base font size passed to ggplot2::theme_minimal().
input	Optional Shiny input.
...	Additional ggplot2::theme() arguments.

Value

A ggplot2 theme object.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  ggplot2::ggplot(mtcars, ggplot2::aes(wt, mpg)) +
    ggplot2::geom_point() +
    theme_glass("light")
}
```

---

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,
  material = NULL,
  ambient_motion = NULL,
  scene = 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`).
`material`	Optional `"regular"` or `"clear"`.
`ambient_motion`	Optional logical.
`scene`	Optional `"default"`, `"tahoe"`, `"dusk"`, or `"mesh"`.

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

bslib::bs_theme(), *8, 9*

ggplot2::theme(), *15*
ggplot2::theme_minimal(), *15*
glass_accent_input, 2
glass_accent_input(), *12*
glass_intensity_slider, 3
glass_intensity_slider(), *9*, *12*, *16*
glass_page, 4
glass_page(), *9*, *11*
glass_plot_colors, 6
glass_plot_colors(), *11*, *14*
glass_preset_input, 6
glass_preset_input(), *13*
glass_resolved_preset, 7
glass_resolved_preset(), *6*, *15*
glass_system_colors, 8
glass_system_colors(), *3*
glass_theme, 8
glass_theme(), *3–5*, *10*, *15*
glass_theme_toggle, 10
glass_theme_toggle(), *7*, *13, 14*
gt::gt(), *11*
gt_theme_glass, 11

htmltools::tag(), *11*

observe_glass, 11
observe_glass(), *4*
observe_glass_accent, 12
observe_glass_accent(), *2*, *11*
observe_glass_intensity, 12
observe_glass_intensity(), *4*, *11*
observe_glass_preset_input, 13
observe_glass_preset_input(), *6, 7*
observe_glass_theme_toggle, 13
observe_glass_theme_toggle(), *10, 11*

plotly_glass, 14

shiny::fluidPage(), *4, 5*
shiny::observeEvent(), *12, 13*
shiny::selectInput(), *6, 7*
shiny::titlePanel(), *5*
shiny::validateCssUnit(), *4*

theme_glass, 14

update_glass_theme, 15
update_glass_theme(), *2*, *4*, *7*, *9*, *11–13*