Package: luwitemplate (via r-universe)

June 7, 2026

Title Unified Brand Styling for Shiny and ggplot2

Version 0.0.0.9000

Description Provides a complete styling system driven by a single '_brand.yml' file. Create consistent Bootstrap themes for Shiny applications, ggplot2 visualizations, and interactive plotly charts with unified colors, fonts, and styling. All components automatically share your brand identity.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

LazyData true

Imports bslib, shiny, ggplot2, showtext, sysfonts, plotly, yaml

Suggests knitr, rmarkdown, sysfonts

URL https://github.com/LukasWillocx/luwitemplate

BugReports https://github.com/LukasWillocx/luwitemplate/issues

VignetteBuilder knitr

Config/pak/sysreqs
cmake libfreetype6-dev make libicu-dev libpng-dev libuv1-dev libssl-dev zlib1g-dev

Repository https://lukaswillocx.r-universe.dev

Date/Publication 2026-02-23 21:55:59 UTC

RemoteUrl https://github.com/LukasWillocx/luwitemplate

RemoteRef HEAD

RemoteSha a702a618b679ef85e007909f16bc6eb4646005b4

Contents

`.onLoad` *Package Load Hook*

Description

Automatically loads and registers Google Fonts specified in the brand YAML when the package is loaded. This ensures fonts are available for ggplot2 themes without requiring manual setup.

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	Character string giving the library directory where the package was installed.
`pkgname`	Character string giving the name of the package.

Value

NULL (invisibly). Called for side effects (font registration).

dark_mode_css *Dark Mode CSS for Third-Party Widgets*

Description

Generates a `tags$style()` block that overrides compiled colors in third-party widget stylesheets (bootstrap-datepicker, Shiny input containers) that load separately from the bslib theme. Place this in your UI to ensure these widgets respect dark mode.

Usage

```
dark_mode_css()
```

Details

Some Shiny widgets (datepicker, checkboxes, radios) load their own CSS files independently of bslib. These files contain compiled hex values that cannot be overridden from within the bslib theme. This function injects a `<style>` tag into the page head that loads after all widget CSS, ensuring the dark palette takes effect.

Value

A `shiny::tags$head()` element to include in your UI

Examples

```
## Not run:
ui <- page_sidebar(
  theme = my_theme(),
  dark_mode_css(),
  input_dark_mode(id = "dark_mode"),
  sidebar = sidebar(
    dateRangeInput("dates", "Date Range:"),
    checkboxInput("flag", "Show details")
  )
)

## End(Not run)
```

`get_theme_colors` *Extract Colors from bslib Theme*

Description

Retrieves all theme color variables as an R-friendly named list. Useful for programmatically accessing your brand colors or building custom palettes.

Usage

```
get_theme_colors(theme = my_theme())
```

Arguments

`theme`	A bslib theme object (defaults to my_theme())

Details

Color names are converted from CSS variable format (hyphenated) to R-friendly format (underscored). For example, "body-bg" becomes "body_bg".

Value

Named list of hex color values with underscore-separated names: `primary`, `secondary`, `success`, `danger`, `warning`, `info`, `light`, `dark`, `body_bg`, `body_color`, `input_border_color`

See Also

`scale_color_luwi_discrete()`, `scale_color_luwi_sequential()`, `scale_color_luwi_diverging()` for ready-to-use palettes

Examples

```
# Get all theme colors
colors <- get_theme_colors()
colors$primary  # Your brand primary color

# Use in custom visualizations
plot(1:10, col = colors$primary, pch = 19)

# Build a custom gradient
my_gradient <- colorRampPalette(c(colors$light, colors$primary))
```

`get_theme_fonts` *Extract Fonts from Brand Configuration*

Description

Retrieves font families defined in your _brand.yml file. Returns a structured list with primary and secondary fonts for easy programmatic access in plots and custom themes.

Usage

```
get_theme_fonts(theme = my_theme())
```

Arguments

`theme`	A bslib theme object (defaults to my_theme()).

Details

This function reads the `typography.fonts` section from your package's _brand.yml file directly. The first font listed becomes `primary`, the second becomes `secondary`.

Expected _brand.yml structure:

```
typography:
  fonts:
    - family: "Open Sans"
      source: google
    - family: "Merriweather"
      source: google
```

Value

Named list with three elements:

`primary` Main font family (used for body text, labels, etc.)

`secondary` Secondary font family if defined (often used for headings), or `NULL`

`all_families` Character vector of all font families defined in brand file

If _brand.yml is not found, returns fallback values with "sans" as primary font.

See Also

`get_theme_colors()` for extracting theme colors, `theme_luwi()` to use these fonts in ggplot2 automatically

Examples

```
# Get font configuration
fonts <- get_theme_fonts()
fonts$primary    # "Open Sans"
fonts$secondary  # "Merriweather" or NULL

# Use in ggplot2
library(ggplot2)
fonts <- get_theme_fonts()

ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(title = "My Plot") +
  theme_minimal(base_family = fonts$primary) +
  theme(
    plot.title = element_text(family = fonts$secondary %||% fonts$primary)
  )

# Check what fonts are available
fonts <- get_theme_fonts()
cat("Available fonts:", paste(fonts$all_families, collapse = ", "))
```

`luwi_ggplotly` *Interactive Plotly with Brand Styling for Shiny*

Description

Converts a ggplot object to an interactive plotly visualization with brand styling automatically applied. Designed specifically for Shiny dashboards with transparent backgrounds, custom fonts, and branded grid lines.

Usage

```
luwi_ggplotly(p, theme = my_theme(), base_size = 14, tooltip = "y")
```

Arguments

`p`	A ggplot2 object to convert
`theme`	A bslib theme object (defaults to my_theme())
`base_size`	Base font size in points (default: 14)
`tooltip`	Which aesthetics to display on hover. Can be a character vector of aesthetic names (e.g., `c("x", "y")`) or "all" for all aesthetics. Default is `"y"` to show only y-values

Value

A plotly htmlwidget ready for Shiny rendering

See Also

theme_luwi() for static ggplot version, plotly::ggplotly() for underlying conversion function, plotly::config() for additional plotly configuration

Other ggplot-themes: theme_luwi()

Examples

```
library(ggplot2)
library(plotly)

# Basic interactive plot
p <- ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  labs(title = "Fuel Economy vs Weight")

luwi_ggplotly(p)

# Custom tooltips showing multiple values
p <- ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
  geom_point(size = 3)

luwi_ggplotly(p, tooltip = c("x", "y", "color"))

# In a Shiny app
## Not run:
library(shiny)

ui <- page_fluid(
  theme = my_theme(),
  plotlyOutput("plot")
)

server <- function(input, output) {
  output$plot <- renderPlotly({
    p <- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
      geom_point()
    luwi_ggplotly(p, tooltip = c("x", "y", "color"))
  })
}

shinyApp(ui, server)

## End(Not run)

# Custom hover text
p <- ggplot(mtcars, aes(mpg, wt, text = paste("Car:", rownames(mtcars)))) +
  geom_point()

luwi_ggplotly(p, tooltip = "text")
```

my_theme *Create Luwi Branded bslib Theme*

Description

Generates a Bootstrap 5 theme from your _brand.yml file with optional custom CSS. This is the central theming function that powers all other styling in the package - use it in your Shiny UI to ensure consistent branding across your application.

Usage

```
my_theme(mode = c("light", "dark"))
```

Arguments

mode	Character. Either "light" (default) or "dark". Controls which color palette is used from the color and color-dark sections of _brand.yml.

Details

This function:

1. Reads your package's _brand.yml file (colors, fonts, spacing)
2. Creates a Bootstrap 5 theme using bs_theme
3. If mode = "dark", overrides color variables with the color-dark palette from the same YAML
4. Applies custom CSS from inst/css/custom.css if available

The returned theme object is used as the default in other package functions like theme_luwi, get_theme_colors, and luwi_ggplotly, ensuring consistent styling across Shiny UI, static plots, and interactive visualizations.

Value

A bslib::bs_theme() object ready to use in Shiny applications

Dark Mode

Dark mode uses the color-dark section of _brand.yml to override Bootstrap Sass variables. This ensures the full cascade of derived variables (RGB decompositions, emphasis colors, subtle backgrounds) is recomputed by the Sass compiler. Use use_dark_mode in your Shiny server to enable reactive toggling.

See Also

bs_theme for the underlying theme constructor, get_theme_colors to extract colors from the theme, get_theme_fonts to extract fonts from the theme, theme_luwi for ggplot2 theme using these colors/fonts, use_dark_mode for Shiny dark mode integration

Other theme-generators: use_dark_mode()

Examples

```
# Light theme (default)
theme <- my_theme()

# Dark theme
dark_theme <- my_theme("dark")

# Use in Shiny app
library(shiny)
library(bslib)

ui <- page_fluid(
  theme = my_theme(),
  h1("My Branded App"),
  p("All Bootstrap components use your brand colors automatically")
)

## Not run:
# App with dark mode toggle
ui <- page_fluid(
  theme = my_theme(),
  input_dark_mode(id = "dark_mode"),
  plotOutput("my_plot")
)

server <- function(input, output, session) {
  dm <- use_dark_mode(input, session)

  output$my_plot <- renderPlot({
    ggplot(mtcars, aes(mpg, wt)) +
      geom_point() +
      theme_luwi(theme = dm$theme())
  })
}

shinyApp(ui, server)

## End(Not run)
```

---

scale_color_luwi_c *Continuous Color Scale (Luwi Brand)*

Description

Apply a sequential gradient color scale for continuous data with brand colors. Convenience wrapper around `scale_color_gradientn()`.

Usage

```
scale_color_luwi_c(type = "warm", theme = my_theme(), ...)
```

Arguments

type	"warm" , "cool" or "green". Default is "warm"
...	Additional arguments passed to ggplot2::scale_color_gradientn() (e.g., limits, breaks, na.value)

Value

A ggplot2 scale object

See Also

scale_fill_luwi_c() for fill aesthetic, scale_color_luwi_sequential() for the underlying palette, scale_color_luwi_div() for diverging data

Other ggplot-scales: scale_color_luwi_d(), scale_color_luwi_div(), scale_fill_luwi_c(), scale_fill_luwi_d(), scale_fill_luwi_div()

Examples

```
library(ggplot2)

# Continuous color scale
ggplot(faithfuld, aes(waiting, eruptions, color = density)) +
  geom_point() +
  scale_color_luwi_c(type = "warm") +
  theme_luwi()

# Cool palette with custom limits
ggplot(diamonds, aes(carat, price, color = depth)) +
  geom_point(alpha = 0.3) +
  scale_color_luwi_c(type = "cool", limits = c(55, 70)) +
  theme_luwi()
```

---

scale_color_luwi_d *Discrete Color Scale (Luwi Brand)*

Description

Apply brand colors for categorical/discrete data. Convenience wrapper around scale_color_manual() with up to 15 distinct brand colors.

Usage

```
scale_color_luwi_d(theme = my_theme(), ...)
```

Arguments

...	Additional arguments passed to ggplot2::scale_color_manual() (e.g., breaks, labels, na.value)

Details

Supports up to 15 categories with carefully selected high-contrast colors. For more than 8 categories, consider grouping into "Other" or using a continuous scale instead.

Value

A ggplot2 scale object

See Also

`scale_fill_luwi_d()` for fill aesthetic, `scale_color_luwi_discrete()` for the underlying palette

Other ggplot-scales: `scale_color_luwi_c()`, `scale_color_luwi_div()`, `scale_fill_luwi_c()`, `scale_fill_luwi_d()`, `scale_fill_luwi_div()`

Examples

```
library(ggplot2)

# Categorical scatter plot
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  scale_color_luwi_d() +
  theme_luwi()

# Line plot with multiple groups
ggplot(economics_long, aes(date, value01, color = variable)) +
  geom_line() +
  scale_color_luwi_d() +
  theme_luwi()
```

---

scale_color_luwi_discrete

Discrete Color Palette for Categorical Data

Description

Provides up to 15 distinct, high-contrast colors for categorical variables like groups, categories, or factors. Colors are carefully selected to be visually distinguishable while maintaining your brand aesthetic.

Usage

```
scale_color_luwi_discrete(theme = my_theme(), n = NULL)
```

Arguments

theme	A bslib theme object (defaults to my_theme())
n	Number of colors needed. If NULL, returns all 15 colors. If n > 15, returns 15 colors with a warning

Details

The palette prioritizes your brand colors (primary, secondary, success, etc.) for the first few categories, then adds complementary colors for additional categories.

Value

Character vector of hex color codes (length = n or 15 if n is NULL)

See Also

ggplot2::scale_color_manual() and ggplot2::scale_fill_manual() to use with ggplot2, scale_color_luwi_sequential() for many categories (converts to gradient)

Other color-palettes: scale_color_luwi_diverging(), scale_color_luwi_sequential()

Examples

```
library(ggplot2)

# Basic categorical plot
ggplot(mpg, aes(class, hwy, fill = class)) +
  geom_boxplot() +
  scale_fill_manual(values = scale_color_luwi_discrete())

# Specify exact number needed
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  scale_color_manual(values = scale_color_luwi_discrete(n = 3))

# Preview all 15 colors
scales::show_col(scale_color_luwi_discrete())

# Warning example: too many categories
scale_color_luwi_discrete(n = 20)  # Returns 15 with warning
```

---

scale_color_luwi_div *Diverging Color Scale (Luwi Brand)*

Description

Apply a two-directional gradient color scale for data with a meaningful midpoint. Convenience wrapper around scale_color_gradientn() using the cool-to-warm diverging palette.

Usage

```
scale_color_luwi_div(theme = my_theme(), ...)
```

Arguments

...	Additional arguments passed to ggplot2::scale_color_gradientn() (e.g., limits, midpoint, breaks)

Details

Use this for data where deviations from a central value matter:

- Positive/negative change
- Above/below average
- Correlation coefficients
- Temperature anomalies

Consider setting midpoint to center the gradient on your neutral value.

Value

A ggplot2 scale object

See Also

scale_fill_luwi_div() for fill aesthetic, scale_color_luwi_diverging() for the underlying palette, ggplot2::scale_color_gradient2() for manual midpoint control

Other ggplot-scales: scale_color_luwi_c(), scale_color_luwi_d(), scale_fill_luwi_c(), scale_fill_luwi_d(), scale_fill_luwi_div()

Examples

```
library(ggplot2)

# Correlation matrix
cor_data <- reshape2::melt(cor(mtcars))
ggplot(cor_data, aes(Var1, Var2, color = value)) +
  geom_point(size = 5) +
  scale_color_luwi_div(limits = c(-1, 1)) +
  theme_luwi()

# With custom midpoint
ggplot(economics, aes(date, unemploy - mean(unemploy))) +
  geom_line() +
  scale_color_luwi_div(midpoint = 0) +
  theme_luwi()
```

`scale_color_luwi_diverging` *Diverging Color Palette for Data with Midpoint*

Description

Generates a two-directional gradient palette for data where deviations from a central value are meaningful (e.g., positive/negative change, hot/cold, above/below average).

Usage

```
scale_color_luwi_diverging(theme = my_theme(), n = 11, reverse = FALSE)
```

Arguments

`theme`	A bslib theme object (defaults to my_theme())
`n`	Number of colors to generate (default: 11). Use odd numbers to ensure a neutral midpoint color
`reverse`	Logical. Flip the palette direction (warm becomes cool and vice versa)

Details

The palette transitions: Teal (cold) → Light blue → Neutral warm → Coral → Red-orange (hot). The neutral midpoint uses your theme's `light` color to maintain brand consistency.

When to use diverging palettes:

- Temperature data (cold to hot)
- Financial change (loss to gain)
- Survey responses (disagree to agree)
- Correlation coefficients (-1 to +1)

Value

Character vector of `n` hex color codes transitioning from cool (teal) through neutral (light) to warm (coral/red)

See Also

`ggplot2::scale_fill_gradient2()` to use with ggplot2, `scale_color_luwi_sequential()` for one-directional data

Other color-palettes: `scale_color_luwi_discrete()`, `scale_color_luwi_sequential()`

Examples

```
library(ggplot2)

# Correlation matrix
ggplot(reshape2::melt(cor(mtcars)), aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_gradientn(
    colors = scale_color_luwi_diverging(n = 11),
    limits = c(-1, 1)
  )

# Temperature anomalies (reversed: red = hot)
temp_palette <- scale_color_luwi_diverging(n = 9, reverse = TRUE)

# Preview the palette
scales::show_col(scale_color_luwi_diverging())
```

---

scale_color_luwi_sequential

Sequential Color Palette for Continuous Data

Description

Generates a gradient palette suitable for continuous/ordered data. Choose from warm (coral-to-red), cool (teal-to-blue), or green (olive-to-dark) gradients.

Usage

```
scale_color_luwi_sequential(
  theme = my_theme(),
  type = "warm",
  n = 9,
  reverse = FALSE
)
```

Arguments

theme	A bslib theme object (defaults to my_theme())
type	Palette type: • "warm" • "cool" • "green"
n	Number of colors to generate (default: 9). More colors = smoother gradient
reverse	Logical. Reverse the palette direction (dark to light instead of light to dark)

Details

These palettes are perceptually uniform and designed to work well in both digital and print contexts. They automatically use your theme's semantic colors as anchor points to ensure brand consistency.

Value

Character vector of n hex color codes

See Also

ggplot2::scale_fill_gradient() and ggplot2::scale_color_gradient() to use with ggplot2, scale_color_luwi_diverging() for data with a meaningful midpoint

Other color-palettes: scale_color_luwi_discrete(), scale_color_luwi_diverging()

Examples

```
library(ggplot2)

# Warm palette for a heatmap
ggplot(faithfuld, aes(waiting, eruptions, fill = density)) +
  geom_tile() +
  scale_fill_gradientn(colors = scale_color_luwi_sequential(type = "warm"))

# Cool palette reversed (dark = high values)
ggplot(economics, aes(date, unemploy)) +
  geom_area(fill = scale_color_luwi_sequential(type = "cool", n = 1))

# Preview the palette
scales::show_col(scale_color_luwi_sequential(type = "green", n = 9))
```

---

scale_fill_luwi_c *Continuous Fill Scale (Luwi Brand)*

Description

Apply a sequential gradient fill scale for continuous data with brand colors. Convenience wrapper around scale_fill_gradientn().

Usage

```
scale_fill_luwi_c(type = "warm", theme = my_theme(), ...)
```

Arguments

type	Palette type: "warm" (coral to red), "cool" (teal to blue), or "green" (olive to dark). Default is "warm"
...	Additional arguments passed to ggplot2::scale_fill_gradientn() (e.g., limits, breaks, na.value)

Value

A ggplot2 scale object

See Also

`scale_color_luwi_c()` for color aesthetic, `scale_color_luwi_sequential()` for the underlying palette, `scale_fill_luwi_div()` for diverging data

Other ggplot-scales: `scale_color_luwi_c()`, `scale_color_luwi_d()`, `scale_color_luwi_div()`, `scale_fill_luwi_d()`, `scale_fill_luwi_div()`

Examples

```
library(ggplot2)

# Heatmap with warm colors
ggplot(faithfuld, aes(waiting, eruptions, fill = density)) +
  geom_tile() +
  scale_fill_luwi_c(type = "warm") +
  theme_luwi()

# Area chart with green gradient
ggplot(economics, aes(date, unemploy)) +
  geom_area(fill = scale_color_luwi_sequential(type = "green", n = 1)) +
  theme_luwi()
```

---

scale_fill_luwi_d *Discrete Fill Scale (Luwi Brand)*

Description

Apply brand colors for categorical/discrete fill aesthetic. Convenience wrapper around `scale_fill_manual()` with up to 15 distinct brand colors.

Usage

```
scale_fill_luwi_d(theme = my_theme(), ...)
```

Arguments

`...`	Additional arguments passed to `ggplot2::scale_fill_manual()` (e.g., `breaks`, `labels`, `na.value`)

Details

Perfect for bar charts, boxplots, and violin plots with categorical groups. Colors are ordered by brand importance: primary, secondary, success, warning, danger.

Value

A ggplot2 scale object

See Also

scale_color_luwi_d() for color aesthetic, scale_color_luwi_discrete() for the underlying palette

Other ggplot-scales: scale_color_luwi_c(), scale_color_luwi_d(), scale_color_luwi_div(), scale_fill_luwi_c(), scale_fill_luwi_div()

Examples

```
library(ggplot2)

# Bar chart
ggplot(mpg, aes(class, fill = class)) +
  geom_bar() +
  scale_fill_luwi_d() +
  theme_luwi()

# Boxplot by category
ggplot(mpg, aes(class, hwy, fill = class)) +
  geom_boxplot() +
  scale_fill_luwi_d() +
  theme_luwi() +
  theme(legend.position = "none")  # Class already on x-axis
```

---

scale_fill_luwi_div *Diverging Fill Scale (Luwi Brand)*

Description

Apply a two-directional gradient fill scale for data with a meaningful midpoint. Convenience wrapper around scale_fill_gradientn() using the cool-to-warm diverging palette.

Usage

```
scale_fill_luwi_div(theme = my_theme(), ...)
```

Arguments

...	Additional arguments passed to ggplot2::scale_fill_gradientn() (e.g., limits, midpoint, breaks)

Details

Ideal for heatmaps and tiles where deviations from center are meaningful.

Value

A ggplot2 scale object

See Also

`scale_color_luwi_div()` for color aesthetic, `scale_color_luwi_diverging()` for the underlying palette

Other ggplot-scales: `scale_color_luwi_c()`, `scale_color_luwi_d()`, `scale_color_luwi_div()`, `scale_fill_luwi_c()`, `scale_fill_luwi_d()`

Examples

```
library(ggplot2)

# Correlation heatmap
cor_data <- reshape2::melt(cor(mtcars))
ggplot(cor_data, aes(Var1, Var2, fill = value)) +
  geom_tile() +
  scale_fill_luwi_div(limits = c(-1, 1)) +
  theme_luwi()
```

---

theme_luwi *Luwi ggplot2 Theme with Brand Styling*

Description

A clean, publication-ready ggplot2 theme that automatically matches your bslib/Shiny application styling. Built on `theme_minimal()` with transparent backgrounds for seamless integration into dashboards.

Usage

```
theme_luwi(theme = my_theme(), base_size = 14)
```

Arguments

`theme`	A bslib theme object (defaults to my_theme())
`base_size`	Base font size in points (default: 14). All text elements scale proportionally from this value

Value

A ggplot2 theme object that can be added to plots with +

See Also

luwi_ggplotly() for interactive plotly version, scale_color_luwi_d(), scale_fill_luwi_c() for matching color scales, ggplot2::theme_set() to apply globally to all plots

Other ggplot-themes: luwi_ggplotly()

Examples

```
library(ggplot2)

# Basic usage
ggplot(mtcars, aes(mpg, wt)) +
  geom_point() +
  theme_luwi()

# With larger text for presentations
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point(size = 3) +
  labs(title = "Iris Measurements") +
  theme_luwi(base_size = 16)

# Set as default theme for all plots
theme_set(theme_luwi())

# Customize further
ggplot(economics, aes(date, unemploy)) +
  geom_line() +
  theme_luwi() +
  theme(panel.grid.major.x = element_blank())  # Remove vertical grid
```

---

use_dark_mode *Enable Dark Mode Toggling in Shiny*

Description

Wires up reactive theme switching for dark mode. Call this once in your Shiny server to get a reactive theme that tracks the toggle state. Bootstrap components are styled via CSS custom property overrides injected by my_theme — no full theme swap occurs on toggle.

Usage

```
use_dark_mode(input, session, input_id = "dark_mode")
```

Arguments

input	The Shiny input object
session	The Shiny session object (kept for API compatibility, not actively used)
input_id	Character. The ID of the bslib::input_dark_mode() widget in your UI. Default is "dark_mode".

Details

The light theme is compiled at initialization. The dark theme is compiled lazily on first toggle to avoid unnecessary startup cost. Bootstrap components switch instantly via CSS custom property overrides (no `session$setCurrentTheme()` call). Pass `dm$theme()` to plot functions to make them reactive to the toggle.

Value

A list with two reactive elements:

`mode` Reactive returning `"dark"` or `"light"`

`theme` Reactive returning the corresponding `bslib::bs_theme()` object from `my_theme`

UI Setup

Your UI must include `bslib::input_dark_mode(id = "dark_mode")` (or a matching `input_id`). Place it in your navbar, sidebar, or wherever makes sense for your layout.

See Also

`my_theme` for the underlying theme constructor, `input_dark_mode` for the toggle widget

Other theme-generators: `my_theme()`

Examples

```
## Not run:
library(shiny)
library(bslib)
library(ggplot2)

ui <- page_sidebar(
  theme = my_theme(),
  title = "Dark Mode Demo",
  sidebar = sidebar(
    input_dark_mode(id = "dark_mode")
  ),
  card(
    card_header("Branded Plot"),
    plotOutput("plot")
  )
)

server <- function(input, output, session) {
  dm <- use_dark_mode(input, session)

  output$plot <- renderPlot({
    ggplot(mtcars, aes(mpg, wt, color = factor(cyl))) +
      geom_point(size = 3) +
      scale_color_luwi_d() +
      theme_luwi(theme = dm$theme())
  })
```

```
}

shinyApp(ui, server)

## End(Not run)
```

Index