Package 'brand.yml'

November 6, 2025

Title Unified Branding with a Simple YAML File

Version 0.1.0

Description Read and process 'brand.yml' 'YAML' files. 'brand.yml' is a simple, portable 'YAML' file that codifies your company's brand guidelines into a format that can be used by 'Quarto', 'Shiny' and 'R' tooling to create branded outputs. Maintain unified, branded theming for web applications to printed reports to dashboards and presentations with a consistent look and feel.

License MIT + file LICENSE

URL https://posit-dev.github.io/brand-yml/pkg/r/, https://github.com/posit-dev/brand-yml

BugReports https://github.com/posit-dev/brand-yml/issues

Depends R (>= 4.1.0)

Imports cli, rlang (>= 1.1.0), utils, yaml

Suggests base64enc, dplyr, flextable, ggiraph, ggplot2 (>= 4.0.0), gt, htmltools, knitr, mime, palmerpenguins, patchwork, plotly, prismatic, rmarkdown, sass, stringr, testthat (>= 3.0.0), thematic (>= 0.1.7.9000), tidyr, tidyverse, withr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Author Garrick Aden-Buie [aut, cre] (ORCID: <https://orcid.org/0000-0002-7111-0077>), Posit Software, PBC [cph, fnd] (ROR: <https://ror.org/03wc8by49>)

Maintainer Garrick Aden-Buie <garrick@posit.co>

Repository CRAN

Date/Publication 2025-11-06 10:40:02 UTC

Contents

`as_brand_yml` *Create a Brand instance from a list or character vector.*

Description

Create a Brand instance from a list or character vector.

Usage

```
as_brand_yml(brand)
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.

Value

A normalized `brand_yml` list.

Examples

```
as_brand_yml("
meta:
  name: Example Brand

color:
  primary: '#FF5733'
```

```
  secondary: '#33FF57'
")

as_brand_yml(list(
  meta = list(name = "Example Brand"),
  color = list(primary = "#FF5733", secondary = "#33FF57")
))
```

`brand_color_pluck` *Extract a color value from a brand object*

Description

Safely extracts a color value from a `brand` object based on the provided key. This function handles color references and resolves them, including references to palette colors and other theme colors. It detects and prevents cyclic references.

Usage

```
brand_color_pluck(brand, key)
```

Arguments

`brand`	A brand object created by `read_brand_yml()` or `as_brand_yml()`.
`key`	A character string representing the color key to extract.

Details

The function checks for the color key in both the main color theme and the color palette. It can resolve references between colors (e.g., if "primary" references "palette.blue"). If a cyclic reference is detected (e.g., A references B which references A), the function will throw an error.

Value

The resolved color value (typically a hex color code) if the key exists, otherwise returns the key itself.

See Also

Other brand.yml helpers: `brand_has()`, `brand_pluck()`, `with_brand_yml_path()`

Examples

```
brand <- as_brand_yml(list(
  color = list(
    primary = "blue",
    secondary = "info",
    info = "light-blue",
    palette = list(
      blue = "#004488",
      light_blue = "#c3ddff"
    )
  )
))

# Extract the primary color
brand_color_pluck(brand, "primary") # "#004488"

# Extract a color that references another color
brand_color_pluck(brand, "info") # "#c3ddff"

# Extract a color that references another color
# which in turn references the palette
brand_color_pluck(brand, "secondary") # "#c3ddff"

# Extract a color that isn't defined
brand_color_pluck(brand, "green") # "green"

# Use brand_pluck() if you need direct (resolved) values
brand_pluck(brand, "color", "primary") # "#004488"
brand_pluck(brand, "color", "info") # "#c3ddff"
brand_pluck(brand, "color", "green") # NULL
```

---

brand_has *Check if a brand has a specific nested element*

Description

Checks if a given brand object has a specific nested element accessible via the additional arguments provided as key paths.

Usage

```
brand_has(brand, ...)
```

Arguments

brand	A brand object created by read_brand_yml() or as_brand_yml().
...	One or more character strings or symbols representing the path to the nested element.

Value

TRUE if the nested element exists in the brand object, FALSE otherwise.

See Also

Other brand.yml helpers: brand_color_pluck(), brand_pluck(), with_brand_yml_path()

Examples

```
brand <- as_brand_yml(list(
  meta = list(name = "Example Brand"),
  color = list(primary = "#FF5733")
))

# Check if brand has a primary color
brand_has(brand, "color", "primary") # TRUE

# Check if brand has a secondary color
brand_has(brand, "color", "secondary") # FALSE
```

---

brand_pluck *Extract a nested element from a brand object*

Description

Safely extracts a nested element from a brand object using the provided key path. Returns NULL if the element doesn't exist.

Usage

```
brand_pluck(brand, ...)
```

Arguments

brand	A brand object created by read_brand_yml() or as_brand_yml().
...	One or more character strings or symbols representing the path to the nested element.

Value

The value of the nested element if it exists, NULL otherwise.

See Also

Other brand.yml helpers: brand_color_pluck(), brand_has(), with_brand_yml_path()

Examples

```
brand <- as_brand_yml(list(
  meta = list(name = "Example Brand"),
  color = list(primary = "#FF5733")
))

# Extract the primary color
brand_pluck(brand, "color", "primary") # "#FF5733"

# Try to extract a non-existent element
brand_pluck(brand, "color", "secondary") # NULL
```

---

brand_sass_color *Generate Sass variables for brand colors*

Description

Creates Sass variables for brand colors with the `brand_color_` prefix. Excludes the color palette which is handled by `brand_sass_color_palette()`.

Usage

```
brand_sass_color(brand)
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.

Value

A list with one component:

- `defaults`: Sass variable definitions with `!default` flag

See Also

Other brand.yml Sass helpers: `brand_sass_color_palette()`, `brand_sass_defaults_bootstrap()`, `brand_sass_fonts()`, `brand_sass_typography()`

Examples

```
brand <- list(
  color = list(
    primary = "#007bff",
    danger = "#dc3545"
  )
)

brand_sass_color(brand)
```

`brand_sass_color_palette` *Generate Sass variables and CSS custom properties for brand color palette*

Description

Converts color palette entries from a brand object to Sass variables with `brand-` prefix and CSS custom properties with `--brand-` prefix.

Usage

```
brand_sass_color_palette(brand)
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.

Value

A list with two components:

- `defaults`: Sass variable definitions with `!default` flag
- `rules`: CSS rules that define custom properties in `:root`

See Also

Other brand.yml Sass helpers: `brand_sass_color()`, `brand_sass_defaults_bootstrap()`, `brand_sass_fonts()`, `brand_sass_typography()`

Examples

```
brand <- list(
  color = list(
    palette = list(
      primary = "#007bff",
      secondary = "#6c757d"
    )
  )
)

brand_sass_color_palette(brand)
```

`brand_sass_defaults_bootstrap` *Generate Sass variables and layer for Bootstrap defaults*

Description

Creates Sass variables and a sass layer from Bootstrap defaults defined in the brand object. Allows overriding defaults from other sources like Shiny themes.

Usage

```
brand_sass_defaults_bootstrap(brand, overrides = "shiny.theme")
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.
`overrides`	Path to override defaults, e.g., "shiny.theme"

Value

A list with two components:

- `defaults`: Sass variable definitions with `!default` flag
- `layer`: A sass_layer object with functions, mixins, and rules

See Also

Other brand.yml Sass helpers: `brand_sass_color()`, `brand_sass_color_palette()`, `brand_sass_fonts()`, `brand_sass_typography()`

Examples

```
brand <- list(
  defaults = list(
    bootstrap = list(
      defaults = list(
        primary = "#007bff",
        enable_rounded = TRUE
      ),
      functions = "@function brand-function() { @return true; }"
    ),
    shiny = list(
      theme = list(
        defaults = list(
          primary = "#428bca"  # Override bootstrap primary
        )
      )
    )
  )
```

```
)

brand_sass_defaults_bootstrap(brand)
```

`brand_sass_fonts` *Generate Sass variables and CSS rules for brand fonts*

Description

Creates Sass variables and CSS rules for fonts defined in the brand object. Supports Google fonts, Bunny fonts, and file-based fonts.

Usage

```
brand_sass_fonts(brand)
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.

Value

A list with two components:

- `defaults`: Sass variables for font definitions
- `rules`: CSS rules for applying fonts via classes

See Also

Other brand.yml Sass helpers: `brand_sass_color()`, `brand_sass_color_palette()`, `brand_sass_defaults_bootstrap`
`brand_sass_typography()`

Examples

```
brand <- list(
  typography = list(
    fonts = list(
      list(
        family = "Roboto",
        source = "google",
        weight = c(400, 700),
        style = "normal"
      )
    )
  )
)

brand_sass_fonts(brand)
```

`brand_sass_typography` *Generate Sass variables for brand typography*

Description

Creates Sass variables for typography settings with the `brand_typography_` prefix. Font size values in pixels are converted to rem units, and color references are resolved.

Usage

```
brand_sass_typography(brand)
```

Arguments

`brand`	A list or string of YAML representing the brand, or a path to a brand.yml file.

Value

A list with one component:

- `defaults`: Sass variable definitions with `!default` flag

See Also

Other brand.yml Sass helpers: `brand_sass_color()`, `brand_sass_color_palette()`, `brand_sass_defaults_bootstrap`
`brand_sass_fonts()`

Examples

```
brand <- list(
  typography = list(
    base = list(
      size = "16px",
      "line-height" = 1.5
    ),
    headings = list(
      weight = "bold",
      style = "normal"
    )
  )
)

brand_sass_typography(brand)
```

brand_use_logo *Extract a logo resource from a brand*

Description

Returns a brand logo resource specified by name and variant from a brand object. The image paths in the returned object are adjusted to be absolute, relative to the location of the brand YAML file, if `brand` was read from a file, or the local working directory otherwise.

Usage

```
brand_use_logo(
  brand,
  name,
  variant = c("auto", "light", "dark", "light-dark"),
  ...,
  .required = !name %in% c("small", "medium", "large", "smallest", "largest"),
  .allow_fallback = TRUE
)
```

Arguments

`brand`	A brand object from `read_brand_yml()` or `as_brand_yml()`.
`name`	The name of the logo to use. Either a size (`"small"`, `"medium"`, `"large"`) or an image name from `brand.logo.images`. Alternatively, you can also use `"smallest"` or `"largest"` to select the smallest or largest available logo size, respectively.
`variant`	Which variant to use, only used when `name` is one of the brand.yml fixed logo sizes (`"small"`, `"medium"`, or `"large"`). Can be one of: • `"auto"`: Auto-detect, returns a light/dark logo resource if both variants are present, otherwise it returns a single logo resource, either the value for `brand.logo.{name}` or the single light or dark variant if only one is present. • `"light"`: Returns only the light variant. If no light variant is present, but `brand.logo.{name}` is a single logo resource and `allow_fallback` is `TRUE`, `brand_use_logo()` falls back to the single logo resource. • `"dark"`: Returns only the dark variant, or, as above, falls back to the single logo resource if no dark variant is present and `allow_fallback` is `TRUE`. • `"light-dark"`: Returns a light/dark object with both variants. If a single logo resource is present for `brand.logo.{name}` and `allow_fallback` is `TRUE`, the single logo resource is promoted to a light/dark logo resource with identical light and dark variants.
`...`	Additional named attributes to be added to the image HTML or markdown when created via `format()`, `knitr::knit_print()`, or `htmltools::as.tags()`.

`.required` Logical or character string. If `TRUE`, an error is thrown if the requested logo is not found. If a string, it is used to describe why the logo is required in the error message and completes the phrase `"is required ____"`. Defaults to `FALSE` when `name` is one of the fixed sizes – `"small"`, `"medium"`, `"large"` or `"smallest"` or `"largest"`. Otherwise, an error is thrown by default if the requested logo is not found.

`.allow_fallback` If `TRUE` (the default), allows falling back to a non-variant-specific logo when a specific variant is requested. Only used when `name` is one of the fixed logo sizes (`"small"`, `"medium"`, or `"large"`).

Value

A `brand_logo_resource` object, a `brand_logo_resource_light_dark` object, or `NULL` if the requested logo doesn't exist and `.required` is `FALSE`.

Shiny apps and HTML documents

You can use `brand_use_logo()` to include logos in Shiny apps or in HTML documents produced by Quarto or R Markdown.

In Shiny apps, logos returned by `brand_use_logo()` will automatically be converted into HTML tags using `htmltools::as.tags()`, so you can include them directly in your UI code.

```
library(shiny)
library(bslib)

brand <- read_brand_yml()

ui <- page_navbar(
  title = tagList(
    brand_use_logo(brand, "small"),
    "Brand Use Logo Test"
  ),
  nav_panel(
    "Page 1",
    card(
      card_header("My Brand"),
      brand_use_logo(brand, "medium", variant = "dark")
    )
  )
  # ... The rest of your app
)
```

If your brand includes a light/dark variant for a specific size, both images will be included in the app, but only the appropriate image will be shown based on the user's system preference of the app's current theme mode if you are using `bslib::input_dark_mode()`.

To include additional classes or attributes in the `<img>` tag, you can call `htmltools::as.tags()` directly and provide those attributes:

```
brand <- as_brand_yml(list(
  logo = list(
    images = list(
      "cat-light" = list(
        path = "https://placecats.com/louie/300/300",
        alt = "A light-colored tabby cat on a purple rug"
      ),
      "cat-dark" = list(
        path = "https://placecats.com/millie/300/300",
        alt = "A dark-colored cat looking into the camera"
      ),
      "cat-med" = "https://placecats.com/600/600"
    ),
    small = list(light = "cat-light", dark = "cat-dark"),
    medium = "cat-med"
  )
))

brand_use_logo(brand, "small") |>
  htmltools::as.tags(class = "my-logo", height = 32)
```

The same applies to HTML documents produced by Quarto or R Markdown, where images can be used in-line:

```
```{r}
brand_use_logo(brand, "small")
```

This is my brand's medium sized logo: `r brand_use_logo(brand, "medium")`
```

Finally, you can use `format()` to convert the logo to raw HTML or markdown:

```
cat(format(brand_use_logo(brand, "small", variant = "light")))
#> <img src="https://placecats.com/louie/300/300" alt="A light-colored tabby cat on a purple rug" class

cat(format(
  brand_use_logo(brand, "medium"),
  .format = "markdown",
  class = "my-logo",
  height = 500
))
#> ![](https://placecats.com/600/600){.brand-logo .my-logo alt="" height="500"}
```

Examples

```
brand <- as_brand_yml(list(
  logo = list(
    images = list(
      small = "logos/small.png",
```

```
      huge = list(path = "logos/huge.png", alt = "Huge Logo")
    ),
    small = "small",
    medium = list(
      light = list(
        path = "logos/medium-light.png",
        alt = "Medium Light Logo"
      ),
      dark = list(path = "logos/medium-dark.png")
    )
  )
))

brand_use_logo(brand, "small")
brand_use_logo(brand, "medium")
brand_use_logo(brand, "large")
brand_use_logo(brand, "huge")

brand_use_logo(brand, "small", variant = "light")
brand_use_logo(brand, "small", variant = "light", allow_fallback = FALSE)
brand_use_logo(brand, "small", variant = "light-dark")
brand_use_logo(
  brand,
  "small",
  variant = "light-dark",
  allow_fallback = FALSE
)

brand_use_logo(brand, "medium", variant = "light")
brand_use_logo(brand, "medium", variant = "dark")
brand_use_logo(brand, "medium", variant = "light-dark")
```

`read_brand_yml` *Create a Brand instance from a Brand YAML file.*

Description

Reads a Brand YAML file or finds and reads a `_brand.yml` file and returns a validated `Brand` instance.

Usage

```
read_brand_yml(path = NULL)
```

Arguments

`path`	The path to the brand.yml file or a directory where `_brand.yml` is expected to be found.

Details

By default, `read_brand_yml()` finds a project-specific `_brand.yml` file, by looking in the current working directory directory or any parent directory for a `_brand.yml`, `brand/_brand.yml` or `_brand/_brand.yml` file (or the same variants with a `.yaml` extension). When `path` is provided, `read_brand_yml()` looks for these files in the provided directory; for automatic discovery, `read_brand_yml()` starts the search in the working directory and moves upward to find the `_brand.yml` file.

Value

A normalized `brand_yml` list from the brand.yml file.

References

https://posit-dev.github.io/brand-yml/

Examples

```
# For this example: copy a brand.yml to a temporary directory
tmp_dir <- tempfile()
dir.create(tmp_dir)
file.copy(
  system.file("examples/brand-posit.yml", package = "brand.yml"),
  file.path(tmp_dir, "_brand.yml")
)

brand <- read_brand_yml(tmp_dir)
```

---

theme_brand_flextable *Create a flextable theme using brand colors*

Description

Apply brand colors to a flextable table.

Usage

```
theme_brand_flextable(
  table,
  brand = NULL,
  background = NULL,
  foreground = NULL
)
```

Arguments

table	A flextable object to theme.
brand	One of: • NULL (default): Automatically detect and read a _brand.yml file • A path to a brand.yml file or directory containing _brand.yml • A brand object (as returned by read_brand_yml() or as_brand_yml()) • FALSE: Don't use a brand file; explicit colors must be provided
background	The background color, defaults to brand.color.background. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.
foreground	The foreground color, defaults to brand.color.foreground. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.

Value

Returns a themed flextable object.

See Also

Other branded theming functions: theme_brand_ggplot2(), theme_brand_gt(), theme_brand_plotly(), theme_brand_thematic()

Examples

```
brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')

library(flextable)
theme_brand_flextable(
  flextable(head(palmerpenguins::penguins)),
  brand
)


brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
```

```
  background: white
  primary: orange')

library(flextable)
theme_brand_flextable(
  flextable(head(mtcars)),
  brand
)
```

`theme_brand_ggplot2` *Create a ggplot2 theme using brand colors*

Description

Create a ggplot2 theme using explicit colors or by automatically extracting colors from a **brand.yml** file.

Usage

```
theme_brand_ggplot2(
  brand = NULL,
  background = NULL,
  foreground = NULL,
  accent = NULL,
  ...,
  base_size = 11,
  title_size = base_size * 1.2,
  title_color = NULL,
  line_color = NULL,
  rect_fill = NULL,
  text_color = NULL,
  plot_background_fill = NULL,
  panel_background_fill = NULL,
  panel_grid_major_color = NULL,
  panel_grid_minor_color = NULL,
  axis_text_color = NULL,
  plot_caption_color = NULL
)
```

Arguments

`brand` One of:

- `NULL` (default): Automatically detect and read a _brand.yml file
- A path to a brand.yml file or directory containing _brand.yml
- A brand object (as returned by `read_brand_yml()` or `as_brand_yml()`)
- `FALSE`: Don't use a brand file; explicit colors must be provided

background The background color, defaults to brand.color.background. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.

foreground The foreground color, defaults to brand.color.foreground. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.

accent The accent color, defaults to brand.color.primary or brand.color.palette.accent. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.

... Reserved for future use.

base_size Base font size in points. Used for the size property of ggplot2::element_text() in the text theme element.

title_size Title font size in points. Used for the size property of ggplot2::element_text() in the title theme element. Defaults to base_size * 1.2.

title_color Color for the color property of ggplot2::element_text() in the title theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to the foreground color.

line_color Color for the color property of ggplot2::element_line() in the line theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to a blend of foreground and background colors.

rect_fill Fill color for the fill property of ggplot2::element_rect() in the rect theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to the background color.

text_color Color for the color property of ggplot2::element_text() in the text theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to a blend of foreground and background colors.

plot_background_fill
Fill color for the fill property of ggplot2::element_rect() in the plot.background theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to the background color.

panel_background_fill
Fill color for the fill property of ggplot2::element_rect() in the panel.background theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to the background color.

panel_grid_major_color
Color for the color property of ggplot2::element_line() in the panel.grid.major theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to a blend of foreground and background colors.

panel_grid_minor_color
Color for the color property of ggplot2::element_line() in the panel.grid.minor theme element. Can be a valid R color or the name of a color in brand.color or brand.color.palette. If not provided, defaults to a blend of foreground and background colors.

`axis_text_color`
: Color for the `color` property of `ggplot2::element_text()` in the `axis.text` theme element. Can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`. If not provided, defaults to a blend of foreground and background colors.

`plot_caption_color`
: Color for the `color` property of `ggplot2::element_text()` in the `plot.caption` theme element. Can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`. If not provided, defaults to a blend of foreground and background colors.

Value

A `ggplot2::theme()` object.

Branded Theming

The `theme_brand_*` functions can be used in two ways:

1. **With a brand.yml file**: The `theme_brand_*` functions use `read_brand_yml()` to automatically detect and use a `_brand.yml` file in your current project. You can also explicitly pass a path to a brand.yml file or a brand object (as returned by `read_brand_yml()` or created with `as_brand_yml()`). When a `brand` is provided, the theme functions will use the colors defined in the brand file automatically.
2. **With explicit colors**: You can directly provide colors to override the default brand colors, or you can use `brand = FALSE` to ignore any project `_brand.yml` files and only use the explicitly provided colors.

See Also

Other branded theming functions: `theme_brand_flextable()`, `theme_brand_gt()`, `theme_brand_plotly()`, `theme_brand_thematic()`

Examples

```
brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')

library(ggplot2)
ggplot(diamonds, aes(carat, price)) +
  geom_point() +
  theme_brand_ggplot2(brand)
```

theme_brand_gt *Create a gt table theme using brand colors*

Description

Apply brand colors to a gt table.

Usage

```
theme_brand_gt(table, brand = NULL, background = NULL, foreground = NULL)
```

Arguments

table	A gt table object to theme.
brand	One of: • NULL (default): Automatically detect and read a _brand.yml file • A path to a brand.yml file or directory containing _brand.yml • A brand object (as returned by read_brand_yml() or as_brand_yml()) • FALSE: Don't use a brand file; explicit colors must be provided
background	The background color, defaults to brand.color.background. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.
foreground	The foreground color, defaults to brand.color.foreground. If provided directly, this value can be a valid R color or the name of a color in brand.color or brand.color.palette.

Value

Returns a themed gt table object.

See Also

Other branded theming functions: theme_brand_flextable(), theme_brand_ggplot2(), theme_brand_plotly(), theme_brand_thematic()

Examples

```
brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')
```

```
library(gt)
theme_brand_gt(
  gt(head(palmerpenguins::penguins)),
  brand
)


brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')

library(gt)
theme_brand_gt(
  gt(head(mtcars)),
  brand
)
```

`theme_brand_plotly` *Create a plotly theme using brand colors*

Description

Apply brand colors to a plotly plot.

Usage

```
theme_brand_plotly(
  plot,
  brand = NULL,
  background = NULL,
  foreground = NULL,
  accent = NULL
)
```

Arguments

`plot`	A plotly plot object to theme.
`brand`	One of: • `NULL` (default): Automatically detect and read a _brand.yml file • A path to a brand.yml file or directory containing _brand.yml • A brand object (as returned by `read_brand_yml()` or `as_brand_yml()`)

- FALSE: Don't use a brand file; explicit colors must be provided

background — The background color, defaults to `brand.color.background`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.

foreground — The foreground color, defaults to `brand.color.foreground`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.

accent — The accent color, defaults to `brand.color.primary` or `brand.color.palette.accent`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.

Value

Returns a themed plotly plot object.

See Also

Other branded theming functions: `theme_brand_flextable()`, `theme_brand_ggplot2()`, `theme_brand_gt()`, `theme_brand_thematic()`

Examples

```
brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')

library(plotly)
plot_ly(palmerpenguins::penguins, x = ~bill_length_mm, y = ~bill_depth_mm) |>
  theme_brand_plotly(brand)


brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')

library(plotly)
plot_ly(mtcars, x = ~wt, y = ~mpg) |>
  theme_brand_plotly(brand)
```

`theme_brand_thematic` *Create a thematic theme using brand colors*

Description

Apply thematic styling using explicit colors or by automatically extracting colors from a **brand.yml** file. This function sets global theming for base R graphics.

Usage

```
theme_brand_thematic(
  brand = NULL,
  background = NULL,
  foreground = NULL,
  accent = NULL,
  ...
)

theme_brand_thematic_on(
  brand = NULL,
  background = NULL,
  foreground = NULL,
  accent = NULL,
  ...
)
```

Arguments

`brand`	One of: • `NULL` (default): Automatically detect and read a _brand.yml file • A path to a brand.yml file or directory containing _brand.yml • A brand object (as returned by `read_brand_yml()` or `as_brand_yml()`) • `FALSE`: Don't use a brand file; explicit colors must be provided
`background`	The background color, defaults to `brand.color.background`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.
`foreground`	The foreground color, defaults to `brand.color.foreground`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.
`accent`	The accent color, defaults to `brand.color.primary` or `brand.color.palette.accent`. If provided directly, this value can be a valid R color or the name of a color in `brand.color` or `brand.color.palette`.
`...`	Additional arguments passed to `thematic::thematic_theme()` or `thematic::thematic_on()`.

Value

`thematic_theme()` returns a theme object as a list (which can be activated with `thematic_with_theme()` or `thematic_set_theme()`).

`thematic_on()`, `thematic_off()`, and `thematic_shiny()` all return the previous global theme.

Functions

- `theme_brand_thematic()`: brand.yml wrapper for `thematic::thematic_theme()`
- `theme_brand_thematic_on()`: brand.yml wrapper for `thematic::thematic_theme()`

See Also

See the "Branded Theming" section of `theme_brand_ggplot2()` for more details on how the `brand` argument works.

Other branded theming functions: `theme_brand_flextable()`, `theme_brand_ggplot2()`, `theme_brand_gt()`, `theme_brand_plotly()`

Examples

```
brand <- as_brand_yml('
color:
  palette:
    black: "#1A1A1A"
    white: "#F9F9F9"
    orange: "#FF6F20"
  foreground: black
  background: white
  primary: orange')


library(ggplot2)

thematic::thematic_with_theme(theme_brand_thematic(brand), {
  ggplot(diamonds, aes(carat, price)) +
    geom_point()
})
```

---

with_brand_yml_path *Temporarily set the* `BRAND_YML_PATH` *environment variable*

Description

This function sets the `BRAND_YML_PATH` environment variable to the specified path for the duration of the local environment. This ensures that, for the scope of the local environment, any calls to functions that automatically discover a `_brand.yml` file will use the path specified.

Usage

```
with_brand_yml_path(path, code)

local_brand_yml_path(path, .local_envir = parent.frame())
```

Arguments

path	The path to a brand.yml file.
code	[any] Code to execute in the temporary environment
.local_envir	[environment] The environment to use for scoping.

Value

[any]
The results of the evaluation of the code argument.

Functions

- with_brand_yml_path(): Run code in a temporary environment with the BRAND_YML_PATH environment variable set to path.
- local_brand_yml_path(): Set the BRAND_YML_PATH environment variable for the scope of the local environment (e.g. within the current function).

See Also

Other brand.yml helpers: brand_color_pluck(), brand_has(), brand_pluck()

Examples

```
# Create a temporary brand.yml file in a tempdir for this example
tmpdir <- withr::local_tempdir("brand")
path_brand <- file.path(tmpdir, "my-brand.yml")
yaml::write_yaml(
  list(color = list(primary = "#abc123")),
  path_brand
)

with_brand_yml_path(path_brand, {
  brand <- read_brand_yml()
  brand$color$primary
})
```

Index