Package: hexmakR (via r-universe)

November 3, 2026

Title Editable Hex Stickers and Project Logos

Version 0.4.1

Description Create publication-ready hexagon stickers for R packages interactively via a 'shiny' app or programmatically via R functions. Complements the 'hexSticker' package (which embeds R plots into stickers) by providing an interactive visual designer with 28 curated colour themes (dark and light variants, led by the nine families of the Computational Trauma and Tissue Injury Research (CTTIR) design, version 5 (CTTIR v5)), 46 built-in scientific and technical icons, custom image upload support, font selection, and export to transparent PNG. Designed for the R package ecosystem with presets for CRAN, Bioconductor, Tidyverse, and domain-specific colour palettes. Includes the CTTIR v5 library of 45 native R vector identities, editable shapes, reproducible JSON projects, and transparent PNG, SVG and PDF export with package names inside the hexagon. Project logos also support pentagonal and square frames.

License MIT + file LICENSE

Copyright see file inst/COPYRIGHTS

URL https://github.com/CTTIR/hexmakR, https://cttir.github.io/hexmakR/

BugReports https://github.com/CTTIR/hexmakR/issues

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports ggplot2 (>= 3.4.0), grid, grDevices, shiny (>= 1.7.0), png, jpeg, xml2, utils, tools, stats, jsonlite, systemfonts (>= 1.3.0)

Suggests testthat (>= 3.3.2), knitr, rmarkdown, svglite, rsvg, colourpicker, withr

Config/testthat/edition 3

Config/Needs/website CTTIR/themakR, pkgdown

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev make libjpeg-dev libpng-dev libuv1-dev libxml2-dev zlib1g-dev

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

Date/Publication 2026-11-02 17:37:32 UTC

RemoteUrl https://github.com/CTTIR/hexmakR

RemoteRef HEAD

RemoteSha 0f4f251590d31ebfd9a07cbee71ad673bb09dbe5

Contents

hex_add_part . . . 3
hex_asset . . . 4
hex_catalog . . . 5
hex_design . . . 6
hex_duplicate_part . . . 7
hex_edit_layer . . . 8
hex_export . . . 8
hex_grob . . . 9
hex_groups . . . 10
hex_import_art . . . 11
hex_label_layer . . . 12
hex_layers . . . 12
hex_move_layer . . . 13
hex_palette . . . 14
hex_recipe . . . 14
hex_render_catalog . . . 15
hex_reuse_part . . . 16
hex_save_design . . . 16
hex_sticker . . . 17
hexmakr_app . . . 19
hexmakr_icons . . . 20
hexmakr_preview_theme . . . 21
hexmakr_themes . . . 22
logo_design . . . 23
save_hex . . . 24
save_hex_svg . . . 25

Index **26**

hex_add_part *Add an editable primitive to a logo*

Description

Add an editable primitive to a logo

Usage

```
hex_add_part(
  design,
  type = c("circle", "rectangle", "line", "polygon"),
  ...,
  fill = "#009E73",
  col = "#FFFFFF",
  lwd = 1,
  alpha = 1
)
```

Arguments

design	A hex_design() object.
type	circle, rectangle, line, or polygon.
...	Primitive geometry in normalised canvas coordinates: x, y, and r for circles; x, y, width, height for rectangles; numeric x and y vectors for lines and polygons. Sensible centred defaults apply.
fill, col	Fill and outline colours, including "transparent".
lwd	Outline width in points on the master canvas.
alpha	Opacity between zero and one.

Value

An updated design with a new stable part_ ID. The input is unchanged.

Examples

```
d <- logo_design("New project")
d <- hex_add_part(d, "circle", x = .5, y = .6, r = .15)
hex_layers(d)
```

hex_asset *Locate an original approved v5 asset*

Description

The 270 SVG and 270 PDF files are the unchanged supplied masters. They are installed in one compressed archive, which is unpacked once per session, and an SVG or PDF request returns a copy with the master's exact bytes. PNG versions are rendered from the SVG master on demand with the 'rsvg' package, at the size of the supplied masters (1600 x 1848 pixels). Icons are drawn by the package renderer, which needs R with Cairo support. All files keep the master's file name and are written to `dir/<asset_dir>/`, where `asset_dir` comes from `hex_catalog()`, and are reused while the hexmakR version and build stay the same. A stamp file, `dir/.hexmakR-version`, names the version and build that wrote them; after an upgrade or a reinstall, each file is copied or rendered again on its next request. Use `hex_export()` to regenerate artwork or export an edited design.

Usage

```
hex_asset(
  package,
  mode = c("dark", "light", "mono"),
  layout = c("lockup", "hex", "icon"),
  format = c("svg", "png", "pdf"),
  size = 128L,
  dir = NULL
)
```

Arguments

package	Exact identity from `hex_catalog()`.
mode	One of `dark`, `light`, `mono`.
layout	`lockup` (name inside), `hex` (artwork only), or `icon`.
format	File format: `svg`, `png`, or `pdf`.
size	Icon width, one of 32, 48, 64, 128, 256. Icons use dark mode.
dir	Directory for the SVG and PDF copies and the rendered PNG and icon files. The default is a folder in `tempdir()`, so nothing is kept after the session. To keep files between sessions, pass a directory of your choice, such as a cache directory returned by `tools::R_user_dir()`.

Value

An absolute file path in `dir`: the copy of the SVG or PDF master, or the rendered PNG or icon.

Examples

```
hex_asset("phenoscapR")
# For a persistent cache, set dir = tools::R_user_dir("hexmakR", "cache").
if (requireNamespace("rsvg", quietly = TRUE)) {
  png <- hex_asset("phenoscapR", format = "png", dir = tempfile())
  dim(png::readPNG(png))
}
if (capabilities("cairo")) {
  hex_asset("phenoscapR", layout = "icon", format = "png", size = 64,
            dir = tempfile())
}
```

`hex_catalog` *Browse the CTTIR v5 identities*

Description

The registry includes 37 R packages, five companion apps, two suites and the CTTIR organisation identity. Exact spelling and case are preserved.

Usage

```
hex_catalog(family = NULL)
```

Arguments

`family`	Optional family key used to filter the registry.

Value

A data frame with identity, repository, family, symbol, kind and source.

Examples

```
hex_catalog()
hex_catalog("spectral")
```

hex_design *Create an editable CTTIR v5 design*

Description

A call without a package starts a fresh composition with a frame and name. Supplying a package explicitly loads its editable archive motif. Archive artwork can also be reused one part at a time with `hex_reuse_part()`. All geometry is drawn in R. Name glyphs use the bundled JetBrains Mono font.

Usage

```
hex_design(
  package = "hexmakR",
  mode = c("dark", "light", "mono"),
  layout = c("lockup", "hex", "motif"),
  label = package,
  colors = list(),
  motif_x = 0.5,
  motif_y = NULL,
  motif_scale = 1,
  label_y = 0.215,
  label_size = 32,
  border_width = 5.5,
  shape = c("hex", "pentagon", "square"),
  icon = NULL,
  archive = !missing(package)
)
```

Arguments

package	Exact identity from `hex_catalog()`.
mode	`dark`, `light`, or `mono`.
layout	`lockup` (name inside), `hex` (unnamed), or `motif` (standalone).
label	Displayed name, defaulting to the exact identity. Names must use characters supported by the bundled JetBrains Mono font and fit inside the frame at a minimum size of six points; otherwise rendering reports an error instead of clipping or substituting characters.
colors	Named list of colour-role overrides, for example `list(face = "#193E36", rim = "#009E73", ink = "white")`. Use `hex_palette()` to see available roles.
motif_x, motif_y	Motif centre in normalised canvas coordinates.
motif_scale	Motif size multiplier.
label_y	Name centre in normalised canvas coordinates.
label_size	Maximum name size in points on the six-inch master.

border_width	Border width on the six-inch master.
shape	Frame shape: hex, pentagon, or square.
icon	Optional generic icon key from hexmakr_icons(), replacing the CTTIR motif.
archive	Include the package's archive motif. By default this is true only when package is supplied explicitly.

Value

A serialisable hex_design object.

Examples

```
design <- hex_design("phenoscapR")
design
```

---

hex_duplicate_part *Duplicate or remove an editable part*

Description

Duplicate or remove an editable part

Usage

```
hex_duplicate_part(design, layer)

hex_remove_part(design, layer)
```

Arguments

design	A hex_design() object.
layer	Stable native or added part ID from hex_layers().

Value

An updated design. Duplication preserves the current appearance. Removal hides native geometry or removes an added part; retaining the old design object permits undo.

Examples

```
d <- hex_add_part(logo_design("New project"), "circle")
d <- hex_duplicate_part(d, "part_0001")
d <- hex_remove_part(d, "part_0001")
hex_layers(d)
```

hex_edit_layer *Edit one vector layer*

Description

Edit one vector layer

Usage

```
hex_edit_layer(design, layer, ..., reset = FALSE)
```

Arguments

design	A hex_design() object.
layer	A stable layer ID from hex_layers().
...	Edits: fill, col, lwd, alpha, dx, dy, scale, angle, visible, or explicit numeric x/y coordinates in the layer's local viewport. angle rotates a part around its centre in degrees, from -180 to 180.
reset	Remove earlier edits to this layer before applying new values.

Value

An updated design. The input is unchanged.

Examples

```
d <- hex_design("phenoscapR")
hex_edit_layer(d, "layer_0001", fill = "#193E36")
```

hex_export *Export an editable v5 design*

Description

The same native grid geometry powers previews and all three export formats. SVG and PDF retain vector paths, including outlined package-name glyphs; uploaded PNG/JPEG figures remain embedded raster images. Rendering finishes in a temporary file beside the destination before replacing it, so a failed export preserves any previous file. Export restores the previously selected graphics device.

Usage

```
hex_export(design, filename, width = 1600L)
```

Arguments

design	A hex_design() object.
filename	Destination ending in .png, .svg or .pdf.
width	PNG width in pixels; vector masters use a six-inch canvas.

Value

The destination path, invisibly.

Examples

```
if (capabilities("cairo")) {
  f <- tempfile(fileext = ".svg")
  hex_export(hex_design("phenoscapR"), f)
}
```

---

hex_grob *Render a design to native grid geometry*

Description

Render a design to native grid geometry

Usage

```
hex_grob(design)
```

Arguments

design	A hex_design() object.

Value

A grid gTree with stable, editable child layer names.

Examples

```
g <- hex_grob(logo_design("My project"))
names(g$children)
```

hex_groups *Inspect and organise groups of logo items*

Description

Native designs begin with Frame, Icon and Name groups when those items are present. Custom groups have stable IDs and one membership per item. Grouping changes metadata only: appearance and stacking order are retained.

Usage

```
hex_groups(design)

hex_group_layers(design, layers, label = "Group", group = NULL)

hex_ungroup_layers(design, layers)

hex_rename_group(design, group, label)
```

Arguments

design	A hex_design() object.
layers	Character vector of IDs from hex_layers().
label	Short item or group name, at most 80 characters.
group	Existing group ID to update; NULL creates a new group.

Value

hex_groups() returns a named list of groups, each with a label and members. The other functions return an updated design.

Examples

```
d <- hex_design("phenoscapR")
hex_groups(d)
d <- hex_group_layers(d, hex_groups(d)$icon$members, "Package icon")
```

`hex_import_art` *Import a figure or vector artwork as editable logo parts*

Description

PNG and JPEG are embedded in the project as portable raster parts. SVG shapes become separate vector parts that can be recoloured, moved, rotated, resized and edited with `hex_edit_layer()`. SVG exports retain vector paths; curves are approximated with bounded polylines (32 segments per Bezier curve, about 32 segments per radian for arcs, and 128 points per ellipse).

Usage

```
hex_import_art(design, path)
```

Arguments

`design`	A `hex_design()` object.
`path`	Local PNG, JPEG or SVG file. Its contents determine its format.

Details

Supported SVG elements are `path`, `rect` (including rounded corners), `circle`, `ellipse`, `line`, `polygon`, `polyline`, and transformed groups. Paths support M/L/H/V/C/S/Q/T/A/Z commands, fill rules, solid fills/strokes, and individual opacity. Local `defs`/`use` references are expanded with cycle checks. Simple convex clip paths are accepted only when they provably do not cut the component; intersecting clips must first be flattened to paths. Root viewport dimensions, `viewBox` and `preserveAspectRatio` are respected; the viewport must provably contain each component, otherwise cropping must first be flattened. Dimensions may use absolute units (including px and pt), but not percentages or font-relative units. An omitted viewport dimension is inferred from the `viewBox` aspect ratio. Text must be outlined. Gradients, masks, filters, nested SVG viewports, CSS stylesheets and group opacity are rejected with an explanation rather than silently rasterised. Non-uniform transforms on strokes require outlined strokes. Scripts, event handlers, external resources and XML entities are always rejected.

Input files are limited to 10 MiB. Raster images are limited to 8192 pixels per side and 16 million pixels total; dimensions are checked before decoding. A project may contain at most 32 million embedded raster pixels altogether. SVG supports at most 512 components, 5000 points per component, 100000 points total per project, 32 nested groups and 8192 expanded element visits. The artwork is fitted proportionally above the project name. Image bytes or vector coordinates are stored in JSON; no original path is retained.

Value

An updated design with additional stable `part_` IDs. The input design and the source file are unchanged.

Examples

```
svg <- tempfile(fileext = ".svg")
writeLines('<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 100 100">
<circle cx="50" cy="50" r="35" fill="#009E73"/></svg>', svg)
d <- hex_import_art(logo_design("New project"), svg)
d <- hex_edit_layer(d, "part_0001", angle = 20, alpha = .8)
unlink(svg)
```

---

hex_label_layer *Label an editable logo item*

Description

Label an editable logo item

Usage

```
hex_label_layer(design, layer, label)
```

Arguments

design	A hex_design() object.
layer	Stable item ID from hex_layers().
label	Short item name, at most 80 characters.

Value

An updated design. Labels survive project saves and duplication.

Examples

```
d <- hex_label_layer(hex_design("phenoscapR"), "layer_0001", "Border")
```

---

hex_layers *Inspect editable vector layers*

Description

Inspect editable vector layers

Usage

```
hex_layers(design)
```

Arguments

design	A hex_design() object.

Value

A data frame with stable IDs, geometry type and current colour roles.

Examples

```
hex_layers(hex_design("phenoscapR"))
```

---

hex_move_layer *Move a logo item in canvas coordinates*

Description

Move a logo item in canvas coordinates

Usage

```
hex_move_layer(design, layer, dx, dy)
```

Arguments

design	A hex_design() object.
layer	Stable item ID from hex_layers().
dx, dy	Horizontal and vertical canvas offsets between -1 and 1. Positive values move right and up. Native and imported viewport scales are accounted for; existing point edits, size and rotation are retained.

Value

An updated design. The input is unchanged.

Examples

```
d <- hex_add_part(logo_design("Project"), "circle")
d <- hex_move_layer(d, "part_0001", .1, .05)
```

hex_palette *Inspect a design's palette roles*

Description

Inspect a design's palette roles

Usage

```
hex_palette(design)
```

Arguments

design	A hex_design() object.

Value

A named list of colours.

Examples

```
hex_palette(hex_design("phenoscapR", "light"))
```

hex_recipe *Write a reproducible R recipe*

Description

Write a reproducible R recipe

Usage

```
hex_recipe(design, path)
```

Arguments

design	A hex_design() object.
path	Destination R file.

Value

The path, invisibly.

Examples

```
path <- tempfile(fileext = ".R")
hex_recipe(logo_design("My project"), path)
head(readLines(path))
```

`hex_render_catalog` *Regenerate the v5 catalogue*

Description

Regenerate the v5 catalogue

Usage

```
hex_render_catalog(
  directory,
  packages = hex_catalog()$name,
  modes = c("light", "dark", "mono"),
  layouts = c("lockup", "hex"),
  formats = c("svg", "png", "pdf"),
  width = 1600L,
  icons = TRUE
)
```

Arguments

`directory`	Destination directory; one subdirectory per identity.
`packages`	Exact identities to render. Defaults to the entire catalogue.
`modes`	Colour modes to export.
`layouts`	Master layouts to export.
`formats`	File formats to export.
`width`	Master PNG width in pixels.
`icons`	Also create five square dark icons per identity.

Value

A data frame of generated file paths and identities, invisibly.

Examples

```
if (capabilities("cairo")) {
  files <- hex_render_catalog(tempfile(), packages = "hexmakR",
                              formats = "svg", icons = FALSE)
  basename(files$path)
}
```

hex_reuse_part *Reuse editable archive parts in another logo*

Description

Reused parts retain a validated snapshot of their native source settings, colour overrides and edits. They remain independently editable. No code, external files or serialised grid objects are loaded from a project.

Usage

```
hex_reuse_part(design, source, layer = NULL)
```

Arguments

design	Destination hex_design() object.
source	A catalogue identity or a hex_design() object.
layer	Native layer ID to reuse. NULL reuses all motif layers, excluding the source frame and name.

Value

An updated destination design; its frame, name and palette are retained.

Examples

```
d <- hex_reuse_part(logo_design("New project"), "phenoscapR", "layer_0007")
```

hex_save_design *Save and load a v5 editing project*

Description

JSON projects store declarative settings, additional parts and layer edits; loading never evaluates R code. Files larger than 20 MiB and unknown fields are rejected.

Usage

```
hex_save_design(design, path)

hex_read_design(path)
```

Arguments

design	A hex_design() object.
path	JSON project path.

Value

`hex_save_design()` returns the path invisibly; `hex_read_design()` returns a validated design.

Examples

```
path <- tempfile(fileext = ".json")
design <- hex_add_part(logo_design("My project"), "circle")
hex_save_design(design, path)
restored <- hex_read_design(path)
hex_layers(restored)
```

---

hex_sticker *Create a Hex Sticker*

Description

Generates a publication-ready hexagon sticker for an R package. Returns a ggplot object that can be printed, further customised with +, or saved with `save_hex()`.

Usage

```
hex_sticker(
  name = "package",
  subtitle = NULL,
  url = NULL,
  icon = NULL,
  icon_image = NULL,
  icon_size = 1,
  theme = "stats",
  mode = "dark",
  bg = NULL,
  accent = NULL,
  text_color = NULL,
  border_color = NULL,
  sub_color = NULL,
  font_family = "mono",
  font_bold = TRUE,
  font_italic = FALSE,
  font_size = 12,
  border_width = 2,
  inner_border = TRUE,
  filename = NULL,
  width = 600,
  dpi = 300
)
```

Arguments

name	Character. The package name displayed on the sticker.
subtitle	Character or NULL. Optional subtitle shown below the package name.
url	Character or NULL. Optional URL shown near the bottom of the sticker.
icon	Character or NULL. Name of a built-in icon. See hexmakr_icons() for valid values. Ignored if icon_image is provided.
icon_image	Character or NULL. Path to a custom image file (PNG or JPEG). If provided, overrides icon and retains the image proportions. Images must be at most 10 MiB, 8192 pixels per side and 16 million pixels in total. Invalid or oversized images report an error before rendering.
icon_size	Numeric. Size multiplier for the icon (default 1.0).
theme	Character. Theme name. One of names(hexmakr_themes()). The nine CTTIR v5 families (for example "spatial" or "clinical") are listed first and are the house standard; for complete v5 artwork use hex_design(). Default "stats".
mode	Character. Colour mode, either "dark" (default) or "light".
bg	Character or NULL. Override the background colour (hex string, e.g. "#1B2838").
accent	Character or NULL. Override the accent colour.
text_color	Character or NULL. Override the text colour.
border_color	Character or NULL. Override the border colour.
sub_color	Character or NULL. Override the subtitle/URL colour.
font_family	Character. Font shorthand (e.g. "mono", "serif", "sans") or a system font name. Default "mono". "jetbrains_mono" uses the JetBrains Mono font bundled for CTTIR v5 names, drawn as glyph outlines so it needs no installed font (bold uses the semibold cut; italic is slanted). Unsupported bundled-font characters report an error.
font_bold	Logical. Use bold weight. Default TRUE.
font_italic	Logical. Use italic style. Default FALSE.
font_size	Numeric. Base font size in pt for the package name. Default 12.
border_width	Numeric. Border line width in mm. Default 2.
inner_border	Logical. Draw a thin inner accent ring. Default TRUE.
filename	Character or NULL. If provided, save the sticker as a transparent PNG at this path using save_hex().
width	Integer. Export width in pixels (height is derived from the hex aspect ratio). Default 600.
dpi	Numeric. Resolution for PNG export. Default 300.

Details

Create a hex sticker

Value

A ggplot object (invisibly if filename is not NULL).

See Also

hexmakr_themes(), hexmakr_icons(), save_hex(), hexmakr_app()

Other stickers: hexmakr_app(), save_hex(), save_hex_svg()

Examples

```
# Minimal sticker
hex_sticker("mypackage")

# With icon and theme
hex_sticker("mypackage", icon = "dna", theme = "genomics")

# CTTIR v5 family colours and the bundled v5 name font
hex_sticker("mypackage", icon = "microscope", theme = "spatial",
            font_family = "jetbrains_mono")

# Save to file
tmp <- tempfile(fileext = ".png")
hex_sticker("mypackage", icon = "atom", theme = "stats", filename = tmp)
file.exists(tmp)
```

---

hexmakr_app *Launch Interactive Hex Sticker Designer*

Description

Opens an interactive Shiny app for designing hex stickers visually. The default workspace starts a new editable logo with a hexagonal, pentagonal or square frame. Add, duplicate, remove and edit individual parts; optionally reuse artwork from the secondary CTTIR Gallery. Portable projects, R recipes and PNG/SVG/PDF export preserve the current design. The classic designer remains available.

Usage

```
hexmakr_app(workspace = c("v5", "classic"), package = "hexmakR", ...)
```

Arguments

workspace	Use the new-logo v5 editor (default) or classic designer.
package	Initially selected CTTIR Gallery identity. The editable logo still starts as a new project until a gallery action is chosen.
...	Additional arguments passed to shiny::runApp().

Details

Launch the hexmakR Shiny App

Value

Does not return; runs the Shiny app.

See Also

hex_sticker(), save_hex()

Other stickers: hex_sticker(), save_hex(), save_hex_svg()

Examples

```
if (interactive()) {
  hexmakr_app()
}
```

`hexmakr_icons` *List Available Icons*

Description

Returns the built-in icon library as a named list. Icons are organized by category. Each icon entry contains a `label` and `draw` function.

Usage

```
hexmakr_icons(category = NULL)
```

Arguments

`category`	Character or `NULL`. If `NULL` (default), returns all categories. Otherwise, one of `"science"`, `"biology"`, `"stats"`, `"medical"`, `"tech"`, `"shapes"`, `"fun"`.

Details

List available icons

Value

A named list of icon entries (each with `label` and `draw`), or a named list of categories if `category = NULL`.

See Also

hex_sticker()

Other themes: hexmakr_preview_theme(), hexmakr_themes()

Examples

```
icons <- hexmakr_icons()
names(icons)
hexmakr_icons("science")
```

`hexmakr_preview_theme` *Preview a Colour Theme*

Description

Renders a sample hex sticker to preview a theme's colours.

Usage

```
hexmakr_preview_theme(theme = "stats", mode = "dark")
```

Arguments

`theme`	Character. Theme name, one of `names(hexmakr_themes())`.
`mode`	Character. Either `"dark"` (default) or `"light"`.

Details

Preview a theme

Value

A ggplot object (invisibly). Also prints the plot.

See Also

`hexmakr_themes()`, `hex_sticker()`

Other themes: `hexmakr_icons()`, `hexmakr_themes()`

Examples

```
hexmakr_preview_theme("stats")
hexmakr_preview_theme("genomics", mode = "light")
hexmakr_preview_theme("spatial")
```

hexmakr_themes *List Available Themes*

Description

Returns a named list of all 28 built-in colour themes for hex stickers. Each theme has `dark` and `light` sub-lists with colour values for `bg`, `accent`, `text`, `border`, and `sub` (subtitle).

The first nine are the CTTIR v5 families (`spectral`, `spatial`, `omics`, `clinical`, `methods`, `simulation`, `evidence`, `platform`, `brand`) and are the house standard. They are derived from the bundled v5 design tokens by the same engine as `hex_design()`: `bg` is the family's tile face, `border` its rim, `text` the v5 heading ink and `sub` the muted text colour. `accent`, used for icons and the inner ring, follows the colour v5 motifs use on each tile: the mid shade (the default motif fill) on dark tiles and the darker edge shade (strokes on a pale badge) on light tiles. Icons reach at least 3:1 and text at least 4.5:1 contrast on every tile. The 19 generic themes follow unchanged. For complete v5 artwork with motifs and outlined names, use `hex_design()`.

Usage

```
hexmakr_themes()
```

Details

List available themes

Value

A named list of length 28. Each element contains:

label Human-readable theme name (character).

group `"CTTIR v5"` for the nine v5 families; absent for the generic themes.

dark List of 5 colour hex strings for dark mode.

light List of 5 colour hex strings for light mode.

See Also

`hex_sticker()`, `hexmakr_preview_theme()`, `hex_design()`

Other themes: `hexmakr_icons()`, `hexmakr_preview_theme()`

Examples

```
themes <- hexmakr_themes()
names(themes)
themes$stats$dark
themes$spatial$light
```

logo_design *Create a project logo in three frame shapes*

Description

Start a fresh project with its name inside a hexagonal, pentagonal or square frame. Add individual parts with hex_add_part(), optionally reuse archive artwork with hex_reuse_part(), or choose a generic icon. The same editing, project and export functions work with every frame shape.

Usage

```
logo_design(
  name,
  motif = NULL,
  shape = c("hex", "pentagon", "square"),
  icon = NULL,
  ...
)
```

Arguments

name	Project name shown inside the frame.
motif	Optional CTTIR archive identity supplying a motif and palette. NULL starts a fresh composition without archive artwork.
shape	hex, pentagon or square.
icon	Optional generic icon key, replacing the CTTIR motif.
...	Other settings passed to hex_design().

Value

An editable hex_design object.

Examples

```
project <- logo_design("New project", shape = "pentagon")
project <- hex_add_part(project, "circle")
project <- hex_reuse_part(project, "phenoscapR")
logo_design("Study", shape = "square", icon = "microscope")
```

save_hex *Save a Hex Sticker*

Description

Renders and saves a hex sticker ggplot object to a PNG file with a transparent background and the correct hexb.in aspect ratio (width / height = sqrt(3) / 2). Rendering finishes in a temporary file beside the destination before replacing it, so a failed export preserves any previous file. The previously selected graphics device is restored after export.

Usage

```
save_hex(sticker, filename, width = 600, dpi = 300)
```

Arguments

`sticker`	A ggplot object returned by `hex_sticker()`.
`filename`	Character. Output file path. Must end in `.png`.
`width`	Integer. Width in pixels. Default `600`.
`dpi`	Numeric. Resolution. Default `300`.

Details

Save a hex sticker to a PNG file

Value

The `filename` path, invisibly.

See Also

`hex_sticker()`

Other stickers: `hex_sticker()`, `hexmakr_app()`, `save_hex_svg()`

Examples

```
s <- hex_sticker("mypackage", icon = "atom", theme = "stats")
tmp <- tempfile(fileext = ".png")
save_hex(s, tmp)
file.exists(tmp)
```

save_hex_svg *Save a Hex Sticker as SVG*

Description

Renders and saves a hex sticker to SVG. Requires the **svglite** package. Rendering finishes in a temporary file beside the destination before replacing it, so a failed export preserves any previous file. The previously selected graphics device is restored.

Usage

```
save_hex_svg(sticker, filename, width = 2)
```

Arguments

sticker	A ggplot object returned by hex_sticker().
filename	Character. Output file path. Must end in .svg.
width	Numeric. Width in inches. Default 2.

Details

Save a hex sticker to an SVG file

Value

The filename path, invisibly.

See Also

save_hex(), hex_sticker()

Other stickers: hex_sticker(), hexmakr_app(), save_hex()

Examples

```
if (requireNamespace("svglite", quietly = TRUE)) {
  s <- hex_sticker("mypackage")
  tmp <- tempfile(fileext = ".svg")
  save_hex_svg(s, tmp)
}
```

Index

∗ **stickers**
  hex_sticker, 17
  hexmakr_app, 19
  save_hex, 24
  save_hex_svg, 25
∗ **themes**
  hexmakr_icons, 20
  hexmakr_preview_theme, 21
  hexmakr_themes, 22

hex_add_part, 3
hex_add_part(), *23*
hex_asset, 4
hex_catalog, 5
hex_catalog(), *4*, *6*
hex_design, 6
hex_design(), *3*, *7–14*, *16*, *18*, *22, 23*
hex_duplicate_part, 7
hex_edit_layer, 8
hex_edit_layer(), *11*
hex_export, 8
hex_export(), *4*
hex_grob, 9
hex_group_layers *(*hex_groups*)*, 10
hex_groups, 10
hex_import_art, 11
hex_label_layer, 12
hex_layers, 12
hex_layers(), *7, 8*, *10*, *12, 13*
hex_move_layer, 13
hex_palette, 14
hex_palette(), *6*
hex_read_design *(*hex_save_design*)*, 16
hex_recipe, 14
hex_remove_part *(*hex_duplicate_part*)*, 7
hex_rename_group *(*hex_groups*)*, 10
hex_render_catalog, 15
hex_reuse_part, 16
hex_reuse_part(), *6*, *23*
hex_save_design, 16
hex_sticker, 17
hex_sticker(), *20–22*, *24, 25*
hex_ungroup_layers *(*hex_groups*)*, 10
hexmakr_app, 19
hexmakr_app(), *19*, *24, 25*
hexmakr_icons, 20
hexmakr_icons(), *7*, *18, 19*, *21, 22*
hexmakr_preview_theme, 21
hexmakr_preview_theme(), *20*, *22*
hexmakr_themes, 22
hexmakr_themes(), *19–21*

logo_design, 23

save_hex, 24
save_hex(), *17–20*, *25*
save_hex_svg, 25
save_hex_svg(), *19, 20*, *24*
shiny::runApp(), *19*

tempdir(), *4*
tools::R_user_dir(), *4*