Package: jointjs (via r-universe)

November 14, 2026

Title Interactive JointJS Diagrams

Version 0.0.0.9002

Description Create interactive 'JointJS' diagrams. Wraps the JavaScript library 'JointJS' to create interactive html widgets. It will let the user draw and interact with diagrams.

Depends R (>= 4.1.0)

Imports htmlwidgets, R6, rlang

Suggests shiny

License GPL (>= 3)

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.0

Collate 'helpers.R' 'cells.R' 'elements.R' 'graphs.R' 'jointjs-package.R' 'jointjs.R' 'links.R' 'paper.R'

URL https://pepijn-devries.github.io/jointjs/

Config/pak/sysreqs cmake make libuv1-dev

Repository https://pepijn-devries.r-universe.dev

Date/Publication 2026-11-14 16:20:22 UTC

RemoteUrl https://github.com/pepijn-devries/jointjs

RemoteRef main

RemoteSha 6768b6e2aad6b473982813f4a21d393dd76b1173

Contents

jointjs 2
jointjs-shiny 3
jointjs_borderedImage 4
jointjs_cell 5

jointjs_circle 6
jointjs_cylinder 7
jointjs_doubleLink 8
jointjs_element 9
jointjs_ellipse 13
jointjs_embeddedImage 14
jointjs_graph 14
jointjs_headeredRectangle 16
jointjs_image 17
jointjs_inscribedImage 18
jointjs_link 19
jointjs_paper 20
jointjs_path 21
jointjs_polygon 22
jointjs_polyline 23
jointjs_rectangle 24
jointjs_shadowLink 25
jointjs_textBlock 26

Index **28**

`jointjs` *Create an Interactive JointJS Diagram*

Description

A wrapper around the JavaScript JointJS library to create an interactive html widget. It can be used to draw and edit diagrams.

Usage

```
jointjs(
  paper = jointjs_paper$new(),
  graph = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`paper`	Either a named list with paper properties, or a jointjs_paper class object.
`graph`	Either a named list of cells to draw, or a jointjs_graph class object
`width`	Fixed width for widget (in css units). The default is `NULL`, which results in intelligent automatic sizing based on the widget's container.
`height`	Fixed height for widget (in css units). The default is `NULL`, which results in intelligent automatic sizing based on the widget's container.

`elementId` Use an explicit element ID for the widget (rather than an automatically generated one). Useful if you have other JavaScript that needs to explicitly discover and interact with a specific widget instance.

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

See Also

Other widgets: `jointjs-shiny`

Examples

```
if (interactive()) {
  graph <-
    jointjs_graph$new()$
      add_elements(
        jointjs_rectangle$new(id = "hello", label = list(text = "Hello"))$
          set_position(5, 5),
        jointjs_circle$new(id = "world", label = list(text = "World"))$
          set_position(70, 5),
        jointjs_doubleLink$new(source = list(id = "hello"),
                               target = list(id = "world"))
      )
  jointjs(graph = graph)
}
```

---

jointjs-shiny *Shiny bindings for jointjs*

Description

Output and render functions for using jointjs within Shiny applications and interactive Rmd documents.

Usage

```
jointjsOutput(outputId, width = "100%", height = "400px")

renderJointjs(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

`outputId` output variable to read from

`width, height` Must be a valid CSS unit (like `"100\%"`, `"400px"`, `"auto"`) or a number, which will be coerced to a string and have `"px"` appended.

expr	An expression that generates a jointjs
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

See Also

Other widgets: jointjs()

---

jointjs_borderedImage *A Bordered Image Element*

Description

An image with a border and label.

Super classes

jointjs_cell -> jointjs_element -> jointjs_image -> jointjs_borderedImage

Methods

Public methods:

- jointjs_borderedImage$new()
- jointjs_borderedImage$clone()

jointjs_borderedImage$new(): Initialise the object

Usage:

```
jointjs_borderedImage$new(href, ...)
```

Arguments:

href A Uniform Resource Identifier (URI) to locate the image

... Arguments passed to parent class initialiser

jointjs_borderedImage$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_borderedImage$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

jointjs_cell *The Backbone of JointJS Elements*

Description

It is a backbone structure used by jointjs to construct elements for a diagram. Usually you don't initiate cells directly. Rather, it is more efficient to create its children and grandchildren, e.g.: jointjs_circle, jointjs_image, jointjs_rectangle, etc.

Methods

Public methods:

- `jointjs_cell$new()`
- `jointjs_cell$set_attrs()`
- `jointjs_cell$get_list()`
- `jointjs_cell$clone()`

`jointjs_cell$new()`: Initialise a jointjs_cell class object.

Usage:

```
jointjs_cell$new(id, type, z, ...)
```

Arguments:

`id` A unique identifier (`character`) used to identify the cell. It can be used to link cells together, by referring to these ids. By default a 16 character random hexadecimal code is generated as an id.

`type` Specifies the type of cell this object represents. Its advised to create cells using its or grand-children (e.g. jointjs_circle, jointjs_rectangle, etc.). They will automatically assign the correct type for you. By default it will be set to `"standard.Rectangle"` (a rectangle).

`z` A depth index. It will determine the plotting order on its parent container.

`...` Parameters that will be set as SVG attributes to this object.

`jointjs_cell$set_attrs()`: Set attributes to the cell

Usage:

```
jointjs_cell$set_attrs(...)
```

Arguments:

`...` Any valid attributes to a JointJS cell.

`jointjs_cell$get_list()`: Coerce the `jointjs_cell` object to a named list.

Usage:

```
jointjs_cell$get_list()
```

`jointjs_cell$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_cell$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other root-classes: jointjs_graph, jointjs_paper

---

jointjs_circle *A Circle Element*

Description

A circle shape.

Super classes

jointjs_cell -> jointjs_element -> jointjs_circle

Methods

Public methods:

- jointjs_circle$new()
- jointjs_circle$clone()

jointjs_circle$new(): Initialise the object

Usage:

```
jointjs_circle$new(...)
```

Arguments:

... Arguments passed to parent class initialiser

jointjs_circle$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_circle$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

jointjs_cylinder *A Cylinder Element*

Description

A cylinder with a vertical base and an elliptical top.

Super classes

jointjs_cell -> jointjs_element -> jointjs_cylinder

Methods

Public methods:

- jointjs_cylinder$new()
- jointjs_cylinder$set_body()
- jointjs_cylinder$clone()

jointjs_cylinder$new(): Initialise the object

Usage:

```
jointjs_cylinder$new(...)
```

Arguments:

... Arguments passed to parent class initialiser

jointjs_cylinder$set_body(): Set body attributes for the element.

Usage:

```
jointjs_cylinder$set_body(fill, stroke, strokeWidth, magnet, topRy, ...)
```

Arguments:

fill Fill colour

stroke Stroke colour

strokeWidth Width of the stroke line

magnet When set to TRUE, the element can be used as source or target for a jointjs_link.

topRy The vertical radius of the exposed area of the cylinder base, as an absolute value or a percentage.

... Arguments passed to parent class initialiser

jointjs_cylinder$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_cylinder$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

---

jointjs_doubleLink *Double Links Between Elements in a JointJS Diagram*

Description

Create a double line link between a source and target element or location in a JointJS diagram.

Super classes

jointjs_cell -> jointjs_link -> jointjs_doubleLink

Methods

Public methods:

- jointjs_doubleLink$new()
- jointjs_doubleLink$clone()

jointjs_doubleLink$new(): The link initialiser.

Usage:

```
jointjs_doubleLink$new(source, target, vertices, ...)
```

Arguments:

`source, target` source and target indicate the route the link should take. It can be a point (a `list` with an `x` and `y` coordinate), or a `list` containing an `id` of the element that should be linked. You can also provide a `port` value to that list if you wish to connect to a specific port on the element.

`vertices` A `list` of coordinates where the link should pass through. The list is unnamed and each element should be a list of and `x` and `y` value.

`...` Arguments passed to parent class initialiser.

jointjs_doubleLink$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_doubleLink$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other link: jointjs_link, jointjs_shadowLink

jointjs_element *Elements to Build a JointJS Diagram*

Description

A generic template to create shapes for a jointjs_graph. Shapes are visualised by https://www.w3.org/groups/wg/svg/ elements. `jointjs_element` objects are R's representation of such elements. This R class specifies the elements such that they can be used to construct a jointjs_graph. It's usually more effective to initialise a child class, such as jointjs_rectangle, jointjs_image, jointjs_circle, etc.

Parameters are all based on SVG attributes, you can use the `...` arguments to pass other valid attributes. Note that the attributes are renamed in camel-case conform the JointJS spefications. So, SVG attribute `font-family` becomes `fontFamily`, `stroke-width` becomes `strokeWidth`, etc.

Super class

jointjs_cell -> `jointjs_element`

Methods

Public methods:

- `jointjs_element$set_position()`
- `jointjs_element$set_image()`
- `jointjs_element$set_border()`
- `jointjs_element$set_background()`
- `jointjs_element$set_top()`
- `jointjs_element$set_line()`
- `jointjs_element$set_shadow()`
- `jointjs_element$set_outline()`
- `jointjs_element$set_wrapper()`
- `jointjs_element$set_header()`
- `jointjs_element$set_headerText()`
- `jointjs_element$set_bodyText()`
- `jointjs_element$set_body()`
- `jointjs_element$set_label()`
- `jointjs_element$get_list()`
- `jointjs_element$clone()`

`jointjs_element$set_position()`: Set the element's position relative to its parent container.

Usage:

```
jointjs_element$set_position(x, y)
```

Arguments:

`x, y` Coordinates relative to its parent container.

`jointjs_element$set_image()`: Set image attributes for the element.

Usage:

```
jointjs_element$set_image(href, ...)
```

Arguments:

`href` A Uniform Resource Identifier (URI) to locate a resource (e.g. image)

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_border()`: Set border attributes for the element.

Usage:

```
jointjs_element$set_border(fill, stroke, strokeWidth, ...)
```

Arguments:

`fill` Fill colour

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_background()`: Set background attributes for the element.

Usage:

```
jointjs_element$set_background(fill, stroke, strokeWidth, ...)
```

Arguments:

`fill` Fill colour

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_top()`: Set top attributes for the element (used to visualise the top of a cylinder base).

Usage:

```
jointjs_element$set_top(fill, stroke, strokeWidth, ...)
```

Arguments:

`fill` Fill colour

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_line()`: Set line attributes for the element.

Usage:

```
jointjs_element$set_line(stroke, strokeWidth, ...)
```

Arguments:

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_shadow()`: Set shadow attributes for the element.

Usage:

```
jointjs_element$set_shadow(stroke, strokeWidth, ...)
```

Arguments:

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_outline()`: Set outline attributes for the element.

Usage:

```
jointjs_element$set_outline(stroke, strokeWidth, ...)
```

Arguments:

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_wrapper()`: Set wrapper attributes for the element.

Usage:

```
jointjs_element$set_wrapper(stroke, strokeWidth, ...)
```

Arguments:

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_header()`: Set header attributes for the element.

Usage:

```
jointjs_element$set_header(fill, stroke, strokeWidth, ...)
```

Arguments:

`fill` Fill colour

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_headerText()`: Set headerText attributes for the element.

Usage:

```
jointjs_element$set_headerText(text, color, fontFamily, fontSize, ...)
```

Arguments:

`text` A text to display

`color` Colour of the element

`fontFamily` Font family for the text to display

`fontSize` Font size for the text to display

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_bodyText()`: Set bodyText attributes for the element.

Usage:

```
jointjs_element$set_bodyText(text, color, fontFamily, fontSize, ...)
```

Arguments:

`text` A text to display

`color` Colour of the element

`fontFamily` Font family for the text to display

`fontSize` Font size for the text to display

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_body()`: Set body attributes for the element.

Usage:

```
jointjs_element$set_body(fill, stroke, strokeWidth, magnet, ...)
```

Arguments:

`fill` Fill colour

`stroke` Stroke colour

`strokeWidth` Width of the stroke line

`magnet` When set to `TRUE`, the element can be used as source or target for a jointjs_link.

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$set_label()`: Set label attributes for the element.

Usage:

```
jointjs_element$set_label(text, color, fontFamily, fontSize, ...)
```

Arguments:

`text` A text to display

`color` Colour of the element

`fontFamily` Font family for the text to display

`fontSize` Font size for the text to display

`...` Additional SVG attributes accepted by the specific SVG element

`jointjs_element$get_list()`: Returns a named list representation of the element.

Usage:

```
jointjs_element$get_list()
```

`jointjs_element$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_element$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other elements: `jointjs_circle`, `jointjs_cylinder`, `jointjs_ellipse`, `jointjs_headeredRectangle`, `jointjs_image`, `jointjs_inscribedImage`, `jointjs_path`, `jointjs_polygon`, `jointjs_polyline`, `jointjs_rectangle`, `jointjs_textBlock`

jointjs_ellipse *An Ellipse Element*

Description

An elliptical shape.

Super classes

jointjs_cell -> jointjs_element -> jointjs_ellipse

Methods

Public methods:

- jointjs_ellipse$new()
- jointjs_ellipse$clone()

jointjs_ellipse$new(): Initialise the object

Usage:

```
jointjs_ellipse$new(...)
```

Arguments:

... Arguments passed to parent class initialiser

jointjs_ellipse$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_ellipse$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

jointjs_embeddedImage *A Embedded Image Element*

Description

An image embedded into a rectangle with a label.

Super classes

jointjs_cell -> jointjs_element -> jointjs_image -> jointjs_embeddedImage

Methods

Public methods:

- jointjs_embeddedImage$new()
- jointjs_embeddedImage$clone()

jointjs_embeddedImage$new(): Initialise the object

Usage:

```
jointjs_embeddedImage$new(href, ...)
```

Arguments:

href A Uniform Resource Identifier (URI) to locate the image

... Arguments passed to parent class initialiser

jointjs_embeddedImage$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_embeddedImage$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

jointjs_graph *Create A JointJS Graph*

Description

Use the jointjs_graph class to define a diagram that can be rendered with jointjs().

Methods

Public methods:

- jointjs_graph$new()
- jointjs_graph$add_elements()
- jointjs_graph$get_widget()
- jointjs_graph$get_list()
- jointjs_graph$clone()

`jointjs_graph$new()`: Initialise a JointJS Graph

Usage:

```
jointjs_graph$new(x, ...)
```

Arguments:

`x` A named `list` of cell objects represented as `lists`. You can use the graph state object returned by a Shiny widget to initiate a graph class object.

`...` Ignored

`jointjs_graph$add_elements()`: Add elements to the graph. These can be any object that inherit from the jointjs_cell class. So, these can be shapes like jointjs_rectangle and jointjs_image, but also links like jointjs_doubleLink.

Usage:

```
jointjs_graph$add_elements(...)
```

Arguments:

`...` Pass any number of items to be added to the graph here.

`jointjs_graph$get_widget()`: This is just short for `jointjs(graph = x)`, where `x` is a `jointjs_graph` class object.

Usage:

```
jointjs_graph$get_widget(...)
```

Arguments:

`...` Arguments passed to `jointjs()`

`jointjs_graph$get_list()`: Returns the `list` of cells that define the graph.

Usage:

```
jointjs_graph$get_list()
```

`jointjs_graph$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_graph$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other root-classes: `jointjs_cell`, `jointjs_paper`

jointjs_headeredRectangle *A Headered Rectangle Element*

Description

A rectangle with a header.

Super classes

jointjs_cell -> jointjs_element -> jointjs_headeredRectangle

Methods

Public methods:

- jointjs_headeredRectangle$new()
- jointjs_headeredRectangle$clone()

jointjs_headeredRectangle$new(): Initialise the object

Usage:

```
jointjs_headeredRectangle$new(header, headerText, bodyText, ...)
```

Arguments:

header Either a jointjs_rectangle or a named list with SVG rectangle attributes.

headerText A named list with SVG text element properties

bodyText A named list with SVG text element properties

... Arguments passed to parent class initialiser

jointjs_headeredRectangle$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_headeredRectangle$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

jointjs_image *An Image Element*

Description

An image with a label.

Super classes

jointjs_cell -> jointjs_element -> jointjs_image

Methods

Public methods:

- jointjs_image$new()
- jointjs_image$clone()

jointjs_image$new(): Initialise the object

Usage:

```
jointjs_image$new(href, ...)
```

Arguments:

href A Uniform Resource Identifier (URI) to locate the image

... Arguments passed to parent class initialiser

jointjs_image$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_image$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

jointjs_inscribedImage *An Inscribed Image Element*

Description

An image inscribed in an ellipse with a label.

Super classes

jointjs_cell -> jointjs_element -> jointjs_image -> jointjs_inscribedImage

Methods

Public methods:

- jointjs_inscribedImage$new()
- jointjs_inscribedImage$clone()

jointjs_inscribedImage$new(): Initialise the object

Usage:

```
jointjs_inscribedImage$new(href, ...)
```

Arguments:

href A Uniform Resource Identifier (URI) to locate the image

... Arguments passed to parent class initialiser

jointjs_inscribedImage$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_inscribedImage$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

jointjs_link *Links Between Elements in a JointJS Diagram*

Description

Create a link between a source and target element or location in a JointJS diagram.

Super class

jointjs_cell -> jointjs_link

Methods

Public methods:

- jointjs_link$new()
- jointjs_link$clone()

jointjs_link$new(): The link initialiser.

Usage:

```
jointjs_link$new(source, target, vertices, ...)
```

Arguments:

source, target source and target indicate the route the link should take. It can be a point (a list with an x and y coordinate), or a list containing an id of the element that should be linked. You can also provide a port value to that list if you wish to connect to a specific port on the element.

vertices A list of coordinates where the link should pass through. The list is unnamed and each element should be a list of and x and y value.

... Arguments passed to parent class initialiser.

Examples:

```
jointjs_link$new(vertices = list(
    list(x = 50, y = 50),
    list(x = 40, y = 60)
  ))
```

jointjs_link$clone(): The objects of this class are cloneable with this method.

Usage:

```
jointjs_link$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other link: jointjs_doubleLink, jointjs_shadowLink

Examples

```
## ------------------------------------------------
## Method `jointjs_link$new()`
## ------------------------------------------------

jointjs_link$new(vertices = list(
    list(x = 50, y = 50),
    list(x = 40, y = 60)
  ))
```

`jointjs_paper` *A Paper to Draw JointJS Diagrams on*

Description

Set up a sheet of paper to draw jointjs diagrams on. This class can be used to define the paper specifications. These specifications are needed when creating a widget with `jointjs()`.

Methods

Public methods:

- `jointjs_paper$new()`
- `jointjs_paper$get_list()`
- `jointjs_paper$clone()`

`jointjs_paper$new()`**:** Initiate a paper object

Usage:

```
jointjs_paper$new(
  width,
  height,
  background = list(color = "#E9ECEF"),
  gridSize = 20,
  drawGrid = list(name = "dot", args = list(color = "#808080", thickness = 1)),
  interactive = TRUE,
  snapLinks = list(radius = 20),
  defaultLink = jointjs::jointjs_cell$new(NULL, "standard.Link", .connection =
   list(stroke = "#2C3E50", strokeWidth = 2), `.marker-target` = list(fill = "#2C3E50",
    d = "M 10 0 L 0 5 L 10 10 z")),
  validateConnection =
   htmlwidgets::JS("\n      function(cellViewS, magnetS, cellViewT, magnetT, end, linkView) {\n
  ...
)
```

Arguments:

`width` Paper width in CSS units.

`height` Paper height in CSS units.

`background` A named list with background properties. The list can contain the elements `color`, `image` (an URL for a background image), `position` (`x` and `y` coordinate for the image), `size` of the background image (`width` and `height`), `repeat` a CSS `background-repeat` property, `quality` coefficient for image quality, `opacity` in the range of 0-1, `watermarkAngle` in degrees. If set to `FALSE` no background is drawn.

`gridSize` gridSize in CSS units

`drawGrid` A named list specifying how to draw a grid on the paper. An element `name` can use any of these predefined grid patterns: `"dot"` (default), `"fixedDot"`, `"mesh"`, or `"doubleMesh"`. Use an element named `args` to specify pattern properties. If set to `FALSE`, no grid is drawn.

`interactive` Use a `logical` value to enable or disable all interactive elements. Or use a named list of interactions with logical values to enable or disable specific interactions

`snapLinks` A `logical` value indicating if links should be snapped to magnets. Or a named `list` with element `radius` in pixels, giving the distance for snapping links.

`defaultLink` Default properties for links. A jointjs_link class object.

`validateConnection` A JavaScript function that validates if a connection between a specific source and target is allowed. It should return a logical value.

`...` Other options passed to the JointJS JavaScript library.

`jointjs_paper$get_list()`: Coerce JointJS paper options to a named list.

Usage:

```
jointjs_paper$get_list()
```

`jointjs_paper$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_paper$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other root-classes: `jointjs_cell`, `jointjs_graph`

`jointjs_path` *A Path Element*

Description

A vector of vertices connected by lines.

Super classes

`jointjs_cell` -> `jointjs_element` -> `jointjs_path`

Methods

Public methods:

- jointjs_path$new()
- jointjs_path$clone()

`jointjs_path$new()`: Initialise the object

Usage:

```
jointjs_path$new(refD, ...)
```

Arguments:

`refD` A `character` string that is used to set the `d` attribute of an SVG `path` element.

`...` Arguments passed to parent class initialiser

`jointjs_path$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_path$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_polygon, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

Examples

```
jointjs_path$new('M 0 5 10 0 C 20 0 20 20 10 20 L 0 15 Z')
```

---

jointjs_polygon *A Polygon Element*

Description

A vector of vertices connected as a polygon.

Super classes

jointjs_cell -> jointjs_element -> jointjs_polygon

Methods

Public methods:

- jointjs_polygon$new()
- jointjs_polygon$clone()

`jointjs_polygon$new()`: Initialise the object

Usage:

```
jointjs_polygon$new(refPoints, ...)
```

Arguments:

`refPoints` A `character` string of coordinates that is used to set the SVG `points` attribute of `polygon` or `polyline` elements.

`...` Arguments passed to parent class initialiser

`jointjs_polygon$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_polygon$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polyline, jointjs_rectangle, jointjs_textBlock

Examples

```
jointjs_polygon$new("10,10 30,10 30,30")
```

---

jointjs_polyline *A Polyline Element*

Description

A polyline with a label.

Super classes

jointjs_cell -> jointjs_element -> jointjs_polyline

Methods

Public methods:

- jointjs_polyline$new()
- jointjs_polyline$clone()

`jointjs_polyline$new()`: Initialise the object

Usage:

```
jointjs_polyline$new(refPoints, ...)
```

Arguments:

`refPoints` A `character` string of coordinates that is used to set the SVG `points` attribute of `polygon` or `polyline` elements.

`...` Arguments passed to parent class initialiser

`jointjs_polyline$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_polyline$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_rectangle, jointjs_textBlock

Examples

```
jointjs_polyline$new("10,10 30,10 30,30")
```

---

jointjs_rectangle *A Rectangle Element*

Description

A Rectangle shape.

Super classes

jointjs_cell -> jointjs_element -> jointjs_rectangle

Methods

Public methods:

- jointjs_rectangle$new()
- jointjs_rectangle$clone()

`jointjs_rectangle$new()`: Initialise the object

Usage:

```
jointjs_rectangle$new(...)
```

Arguments:

`...` Arguments passed to parent class initialiser

`jointjs_rectangle$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_rectangle$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_textBlock

---

jointjs_shadowLink *Shadow Links Between Elements in a JointJS Diagram*

Description

Create a thicker line link with shadow between a source and target element or location in a JointJS diagram.

Super classes

jointjs_cell -> jointjs_link -> jointjs_shadowLink

Methods

Public methods:

- jointjs_shadowLink$new()
- jointjs_shadowLink$clone()

`jointjs_shadowLink$new()`: The link initialiser.

Usage:

```
jointjs_shadowLink$new(source, target, vertices, ...)
```

Arguments:

`source, target` `source` and `target` indicate the route the link should take. It can be a point (a `list` with an `x` and `y` coordinate), or a `list` containing an `id` of the element that should be linked. You can also provide a `port` value to that list if you wish to connect to a specific port on the element.

`vertices` A `list` of coordinates where the link should pass through. The list is unnamed and each element should be a list of and `x` and `y` value.

`...` Arguments passed to parent class initialiser.

`jointjs_shadowLink$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_shadowLink$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other link: `jointjs_doubleLink`, `jointjs_link`

`jointjs_textBlock` *A Text Block Element*

Description

A rectangle shape with text.

Super classes

`jointjs_cell` -> `jointjs_element` -> `jointjs_textBlock`

Methods

Public methods:

- `jointjs_textBlock$new()`
- `jointjs_textBlock$clone()`

`jointjs_textBlock$new()`: Initialise the object

Usage:

```
jointjs_textBlock$new(text, ...)
```

Arguments:

`text` A `character` string to be displayed.

`...` Arguments passed to parent class initialiser

`jointjs_textBlock$clone()`: The objects of this class are cloneable with this method.

Usage:

```
jointjs_textBlock$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

Other elements: jointjs_circle, jointjs_cylinder, jointjs_element, jointjs_ellipse, jointjs_headeredRectangle, jointjs_image, jointjs_inscribedImage, jointjs_path, jointjs_polygon, jointjs_polyline, jointjs_rectangle

Index

∗ **elements**
  jointjs_circle, 6
  jointjs_cylinder, 7
  jointjs_element, 9
  jointjs_ellipse, 13
  jointjs_headeredRectangle, 16
  jointjs_image, 17
  jointjs_inscribedImage, 18
  jointjs_path, 21
  jointjs_polygon, 22
  jointjs_polyline, 23
  jointjs_rectangle, 24
  jointjs_textBlock, 26
∗ **link**
  jointjs_doubleLink, 8
  jointjs_link, 19
  jointjs_shadowLink, 25
∗ **root-classes**
  jointjs_cell, 5
  jointjs_graph, 14
  jointjs_paper, 20
∗ **widgets**
  jointjs, 2
  jointjs-shiny, 3

jointjs, 2
jointjs(), *4*, *14*, *15*, *20*
jointjs-shiny, 3
jointjs_borderedImage, 4
jointjs_cell, *4*, 5, *6–9*, *13–19*, *21–26*
jointjs_circle, *5*, 6, *8*, *9*, *12*, *13*, *16–18*, *22–25*, *27*
jointjs_cylinder, *6*, 7, *12*, *13*, *16–18*, *22–25*, *27*
jointjs_doubleLink, 8, *15*, *19*, *26*
jointjs_element, *4*, *6–8*, 9, *13*, *14*, *16–18*, *21–27*
jointjs_ellipse, *6*, *8*, *12*, 13, *16–18*, *22–25*, *27*
jointjs_embeddedImage, 14
jointjs_graph, *2*, *6*, *9*, 14, *21*
jointjs_headeredRectangle, *6*, *8*, *12*, *13*, 16, *17*, *18*, *22–25*, *27*
jointjs_image, *4–6*, *8*, *9*, *12–16*, 17, *18*, *22–25*, *27*
jointjs_inscribedImage, *6*, *8*, *12*, *13*, *16*, *17*, 18, *22–25*, *27*
jointjs_link, *7*, *8*, *12*, 19, *21*, *25*, *26*
jointjs_paper, *2*, *6*, *15*, 20
jointjs_path, *6*, *8*, *12*, *13*, *16–18*, 21, *23–25*, *27*
jointjs_polygon, *6*, *8*, *12*, *13*, *16–18*, 22, *22*, *24*, *25*, *27*
jointjs_polyline, *6*, *8*, *12*, *13*, *16–18*, *22*, 23, *23*, *25*, *27*
jointjs_rectangle, *5*, *6*, *8*, *9*, *12*, *13*, *15–18*, *22*, *23*, 24, *24*, *27*
jointjs_shadowLink, *8*, *19*, 25
jointjs_textBlock, *6*, *8*, *12*, *13*, *16–18*, *22–25*, 26
jointjsOutput *(*jointjs-shiny*)*, 3

renderJointjs *(*jointjs-shiny*)*, 3