Package ‘shinyCyJS’

September 26, 2023

Title  Create Interactive Network Visualizations in R and ‘shiny’
Version  1.0.0
Description  Create Interactive Graph (Network) Visualizations.
’shinyCyJS’ can be used in ‘Shiny’ apps or viewed from ‘Rstudio’ Viewer.
’shinyCyJS’ includes API to build Graph model like node or edge with customized attributes for R.
’shinyCyJS’ is built with ‘cytoscape.js’ and ‘htmlwidgets’ R package.
License  MIT + file LICENSE
URL  https://github.com/jhk0530/shinyCyJS
BugReports  https://github.com/jhk0530/shinyCyJS/issues
Encoding  UTF-8
RoxygenNote  7.2.3
Imports  htmlwidgets
Suggests  testthat (>= 2.1.0), rmarkdown
NeedsCompilation  no
Author  Jinhwan Kim [aut, cre, cph]
Maintainer  Jinhwan Kim <hwanistic@gmail.com>
Repository  CRAN
Date/Publication  2023-09-26 02:30:02 UTC

R topics documented:

  buildEdge .......................................................... 2
  buildElems .......................................................... 3
  buildIOptions ...................................................... 4
  buildNode .......................................................... 5
  buildROptions ...................................................... 7
  renderShinyCyJS ...................................................... 9
  shinyCyJS .......................................................... 9
  ShinyCyJSOutput ................................................... 10

Index  11
### Description

build single Edge element

### Usage

```python
buildEdge(
    source = NULL,
    target,
    width = 3,
    curveStyle = "haystack",
    label = "",
    fontSize = 16,
    lineColor = "#FECA57",
    lineStyle = "solid",
    sourceArrowColor = "#feca57",
    targetArrowColor = "#feca57",
    sourceArrowShape = "none",
    targetArrowShape = "none",
    opacity = 1,
    tooltip = ""
)
```

### Arguments

- **source**: edge linked node’s id. [string]
- **target**: edge linked target node’s id. [string]
- **width**: The width of an edge’s line. [numeric]
- **curveStyle**: The curving method used to separate two or more edges between two nodes. [string]
- **label**: edge’s label [string]
- **fontSize**: edge labels font size [numeric]
- **lineColor**: The colour of the edge’s line. [string]
- **lineStyle**: The style of the edge’s line; may be solid, dotted, or dashed. [string]
- **sourceArrowColor**: The colour of the edge’s source arrow. [string]
- **targetArrowColor**: The colour of the edge’s target arrow. [string]
- **sourceArrowShape**: The shape of the edge’s source arrow. [string]
The shape of the edge's target arrow. [string]
Opacity of edge itself. [numeric between 0 ~ 1]
Text for tooltip. [string]

Value
List typed Edge element, consisted with data options (source, target, data) and style options (width, curvestyle...)

See Also
https://js.cytoscape.org/#style

buildElems  
build multiple network elements by dataframe

Description
call buildNode or buildEdge function, note that only one function can be called

Usage
buildElems(elems, type)

Arguments
elems  
value of elements consisted in dataframe
type  
Either 'Node' or 'Edge' [string]

Value
List typed multiple 'Node' or 'Edge' elements. It consisted with repeated buildNode or buildEdge function results with given parameter.

See Also
buildNode(), buildEdge()

Examples
nodes <- buildElems(  
  # will generate 5 nodes
  elems = data.frame(    
    id = paste0("node", 1:5),
    bgColor = "#FFFFFF",
    borderColor = "#48DBFB",
    borderWidth = 2,
    fontSize = 10,
  )
)
buildIOptions

width = 60, height = 20, opacity = 1, stringsAsFactors = FALSE, type = "Node"

Description
build Interact Option

Usage
buildIOptions(
  minZoom = 1e-50,
  maxZoom = 1e+50,
  zoomingEnabled = TRUE,
  userZoomingEnabled = TRUE,
  panningEnabled = TRUE,
  userPanningEnabled = TRUE,
  boxSelectionEnabled = TRUE,
  selectionType = "single",
  autolock = FALSE,
  autoungrabify = FALSE,
  autounselectify = FALSE
)

Arguments
minZoom Minimal zoom level of canvas. [numeric]
maxZoom Maximal zoom level of canvas. [numeric]
zoomingEnabled Whether canvas can zoom or not. by both user event and programmatically. [logical]
userZoomingEnabled Whether canvas can zoom or not. by user event. [logical]
panningEnabled Whether canvas can move or not. by both user event and programmatically. [logical]
userPanningEnabled Whether canvas can move or not. by user event. [logical]
boxSelectionEnabled Whether box selection by drag available [logical]
selectionType Indicate selection by user input is additive or single(default). [‘single’ or ‘additive’]
autolock Whether nodes should be locked (not draggable at all) by default (if true, overrides individual node state). [logical]
buildNode

autoungrabify  Whether nodes should be ungrabified (not grabbable by user) by default (if true, overrides individual node state). [logical]

autounselectify  Whether nodes should be unselectified (immutable selection state) by default (if true, overrides individual element state). [logical]

Details

unordered parameter will set as default. note that touchTapThreshold & desktopTapThreshold were not used.

Value

List typed Interact Option for Cytoscape.js canvas object.

See Also

https://js.cytoscape.org/#core/initialisation

Examples

iopt <- buildIOptions(  
  minZoom = 0.001, maxZoom = 3, zoomingEnabled = TRUE,  
  userZoomingEnabled = FALSE, panningEnabled = TRUE, userPanningEnabled = TRUE,  
  boxSelectionEnabled = FALSE, selectionType = "single", autolock = FALSE,  
  autoungrabify = TRUE, autounselectify = FALSE  
)

---

buildNode  

build single node element.

Description

build single node element.

Usage

buildNode(  
  id = NULL,  
  width = 15,  
  height = 15,  
  shape = "ellipse",  
  bgColor = "#48DBFB",  
  bgOpacity = 1,  
  bgFill = "solid",  
  bgBlacken = 0,  
  borderWidth = 0,
borderStyle = "solid",
borderColor = "#8395a7",
borderOpacity = 1,
isParent = FALSE,
label = NULL,
labelColor = "#8395a7",
textOpacity = 1,
fontSize = 16,
textOutlineColor = "#222f3e",
textOutlineOpacity = 1,
textOutlineWidth = 0,
textbgColor = "#FFF",
textbgOpacity = 0,
textBorderColor = "#222f3e",
textBorderOpacity = 0,
textBorderWidth = 0,
parent = NULL,
opacity = 1,
pieSize = rep("0%", 16),
pieColor = rep("#000", 16),
tooltip = "",
position.x = 0,
position.y = 0
)

Arguments

id id of node element. Also it will used as label. [string]
width Width. [numeric]
height Height. [numeric]
shape Shape of node body. polygon not accepted. [string]
bgColor Background color of node body. [string]
bgOpacity Opacity of backgroundColor. [numeric between 0 ~ 1]
bgFill The filling style of the node’s body; may be solid (default), linear-gradient, or radial-gradient. [string]
bgBlacken Blackens the node’s body for values from 0 to 1; whitens the node’s body for values from 0 to -1. [numeric between -1 ~ 1]
borderWidth The size of the node’s border. [numeric]
borderStyle The style of the node’s border; may be solid, dotted, dashed, or double. [string]
borderColor The colour of the node’s border. [string]
borderOpacity The opacity of the node’s border. [numeric between 0 ~ 1]
isParent whether this node is parent node or not [logical]
label node’s label, default is node’s id [string]
labelColor The color of node’s label
textOpacity  The opacity of the label text, including its outline. [numeric between 0 ~ 1]
fontSize    The size of the label text. [numeric]
textOutlineColor The colour of the outline around the element’s label text. [string]
textOutlineOpacity The opacity of the outline on label text. [numeric between 0 ~ 1]
textOutlineWidth The size of the outline on label text. [numeric]
textbgColor colour to apply on the text background. [string]
textbgOpacity The opacity of the label background; the background is disabled for 0 (default value). [numeric between 0 ~ 1]
textBorderColor The colour of the border around the label. [string]
textBorderOpacity The width of the border around the label; the border is disabled for 0 (default value) [numeric between 0 ~ 1]
textBorderWidth The width of the border around the label. [numeric]
parent      Indicate which node is parent of this node [string]
opacity     Opacity of node itself. [numeric between 0 ~ 1]
pieSize     Implement for pie node, consisted with 16 pie size [string]
pieColor    Color for each pie part. [string]
tooltip     Text for tooltip. [string]
position.x  Location value (specify the location of of Node)
position.y  Location value (specify the location of of Node)

Value
List typed Node element, consisted with data options ( id ) and style options ( width, shape... )

See Also
https://js.cytoscape.org/#style
Usage

buildROptions(
  headless = FALSE,
  styleEnabled = TRUE,
  hideEdgesOnViewport = FALSE,
  textureOnViewport = FALSE,
  motionBlur = FALSE,
  motionBlurOpacity = 0.2,
  wheelSensitivity = 1,
  pixelRatio = "auto"
)

Arguments

headless A convenience option that initialises the instance to run headlessly. [logical]
styleEnabled Whether style available or not. [logical]
hideEdgesOnViewport Whether edge will show on canvas manipulation. [logical]
textureOnViewport Whether texture used in canvas manipulation. [logical]
motionBlur Whether use motionBlur effect. [logical]
motionBlurOpacity opacity of motion blur frames [numeric between 0 ~ 1 (transparent)]
wheelSensitivity Changes the scroll wheel sensitivity when zooming. [numeric between 0 (zoom slower) ~ 1 (zoom faster)]
pixelRatio Overrides the screen pixel ratio with a manually set value [numeric]

Details

undescribed parameter will set as default.

Value

List typed Rendering Option for Cytoscape.js canvas object.

See Also

https://js.cytoscape.org/#core/initialisation

Examples

ropt <- buildROptions(wheelSensitivity = 0.5)
renderShinyCyJS

ShinyCyJS output

Description
renders a cytoscape image for output

Usage
renderShinyCyJS(expr, env = parent.frame(), quoted = FALSE)

Arguments
expr expression that returns a list
denv the environment in which to evaluate expr
quoted is expr a quoted expression (with quote())

See Also
ShinyCyJSOutput()

shinyCyJS

cytoscape.js in shiny application

Description
generate canvas with given network element and options

Usage
shinyCyJS(
  elements = list(),
options = list(),
  layout = list(name = "cose"),
  width = NULL,
  height = NULL,
  elementId = NULL,
  ...
)
)
**Arguments**

- **elements** node and edge objects, it should be list of element.
- **options** rendering / interaction options, can be created with buildIoption(), buildRoption()
- **layout** list type layout, it must be contain name and other optional values
- **width** canvas width.
- **height** canvas height.
- **elementId** id used to identify in application.
- **[...]** other parameters

---

**ShinyCyJSOutput** *create an cytoscape canvas element*

---

**Description**

render a renderShinyCyJS() within an application page.

**Usage**

```r
ShinyCyJSOutput(outputId, width = "100\%", height = "400px")
```

**Arguments**

- **outputId** output variable to read the canvas from
- **width** canvas width
- **height** canvas height

**See Also**

renderShinyCyJS()
Index

buildEdge, 2
buildElems, 3
buildIOptions, 4
buildNode, 5
buildROptions, 7

renderShinyCyJS, 9

shinyCyJS, 9
ShinyCyJSOutput, 10