

# Package ‘RCyjs’

November 14, 2025

**Type** Package

**Title** Display and manipulate graphs in cytoscape.js

**Version** 2.33.0

**Date** 2024-04-30

**Author** Paul Shannon

**Maintainer** Paul Shannon <paul.thurmond.shannon@gmail.com>

**Depends** R (>= 3.5.0), BrowserViz (>= 2.7.18), graph (>= 1.56.0)

**Imports** methods, httpuv (>= 1.5.0), BiocGenerics, base64enc, utils

**Suggests** RUnit, BiocStyle, knitr, rmarkdown

## Description

Interactive viewing and exploration of graphs, connecting R to Cytoscape.js, using websockets.

**License** MIT + file LICENSE

**LazyLoad** yes

**biocViews** Visualization, GraphAndNetwork, ThirdPartyClient

**VignetteBuilder** knitr

**RoxygenNote** 7.2.3

**git\_url** <https://git.bioconductor.org/packages/RCyjs>

**git\_branch** devel

**git\_last\_commit** 85acbb7

**git\_last\_commit\_date** 2025-10-29

**Repository** Bioconductor 3.23

**Date/Publication** 2025-11-13

## Contents

|                                         |   |
|-----------------------------------------|---|
| addGraph,RCyjs-method . . . . .         | 3 |
| addGraphFromFile,RCyjs-method . . . . . | 4 |
| clearSelection,RCyjs-method . . . . .   | 5 |
| createTestGraph . . . . .               | 6 |

|                                                            |    |
|------------------------------------------------------------|----|
| dataFramesToJSON . . . . .                                 | 6  |
| deleteGraph,RCyjs-method . . . . .                         | 7  |
| deleteSelectedNodes,RCyjs-method . . . . .                 | 8  |
| eda . . . . .                                              | 8  |
| edaNames . . . . .                                         | 9  |
| fit,RCyjs-method . . . . .                                 | 10 |
| fitSelection,RCyjs-method . . . . .                        | 10 |
| getEdgeCount,RCyjs-method . . . . .                        | 11 |
| getJSON,RCyjs-method . . . . .                             | 12 |
| getLayoutStrategies,RCyjs-method . . . . .                 | 12 |
| getNodeCount,RCyjs-method . . . . .                        | 13 |
| getNodes,RCyjs-method . . . . .                            | 14 |
| getPosition,RCyjs-method . . . . .                         | 14 |
| getSelectedNodes,RCyjs-method . . . . .                    | 15 |
| getSupportedEdgeDecoratorShapes,RCyjs-method . . . . .     | 16 |
| getSupportedNodeShapes,RCyjs-method . . . . .              | 16 |
| getZoom,RCyjs-method . . . . .                             | 17 |
| graphNELtoJSON.string . . . . .                            | 18 |
| hAlign,RCyjs-method . . . . .                              | 18 |
| hideAllEdges,RCyjs-method . . . . .                        | 19 |
| hideEdges,RCyjs-method . . . . .                           | 20 |
| hideNodes,RCyjs-method . . . . .                           | 21 |
| hideSelectedNodes,RCyjs-method . . . . .                   | 22 |
| invertNodeSelection,RCyjs-method . . . . .                 | 23 |
| layout,RCyjs-method . . . . .                              | 23 |
| layoutSelectionInGrid,RCyjs-method . . . . .               | 24 |
| layoutSelectionInGridInferAnchor,RCyjs-method . . . . .    | 25 |
| loadStyleFile,RCyjs-method . . . . .                       | 26 |
| noa . . . . .                                              | 27 |
| noaNames . . . . .                                         | 27 |
| RCyjs-class . . . . .                                      | 28 |
| readAndStandardizeJSONNetworkFile . . . . .                | 29 |
| readAndStandardizeJSONStyleFile . . . . .                  | 29 |
| redraw,RCyjs-method . . . . .                              | 30 |
| restoreLayout,RCyjs-method . . . . .                       | 30 |
| saveJPG,RCyjs-method . . . . .                             | 31 |
| saveLayout,RCyjs-method . . . . .                          | 32 |
| savePNG,RCyjs-method . . . . .                             | 33 |
| selectFirstNeighborsOfSelectedNodes,RCyjs-method . . . . . | 34 |
| selectNodes,RCyjs-method . . . . .                         | 34 |
| setBackgroundColor,RCyjs-method . . . . .                  | 35 |
| setDefaultEdgeColor,RCyjs-method . . . . .                 | 36 |
| setDefaultEdgeLineColor,RCyjs-method . . . . .             | 36 |
| setDefaultEdgeLineStyle,RCyjs-method . . . . .             | 37 |
| setDefaultEdgeSourceArrowColor,RCyjs-method . . . . .      | 38 |
| setDefaultEdgeSourceArrowShape,RCyjs-method . . . . .      | 39 |
| setDefaultEdgeTargetArrowColor,RCyjs-method . . . . .      | 40 |
| setDefaultEdgeTargetArrowShape,RCyjs-method . . . . .      | 40 |

|                                                  |           |
|--------------------------------------------------|-----------|
| setDefaultEdgeWidth,RCyjs-method . . . . .       | 41        |
| setDefaultNodeBorderColor,RCyjs-method . . . . . | 42        |
| setDefaultNodeBorderWidth,RCyjs-method . . . . . | 43        |
| setDefaultNodeColor,RCyjs-method . . . . .       | 43        |
| setDefaultNodeFontColor,RCyjs-method . . . . .   | 44        |
| setDefaultNodeFontSize,RCyjs-method . . . . .    | 45        |
| setDefaultNodeHeight,RCyjs-method . . . . .      | 46        |
| setDefaultNodeShape,RCyjs-method . . . . .       | 46        |
| setDefaultNodeSize,RCyjs-method . . . . .        | 47        |
| setDefaultNodeWidth,RCyjs-method . . . . .       | 48        |
| setDefaultStyle,RCyjs-method . . . . .           | 49        |
| setEdgeAttributes,RCyjs-method . . . . .         | 49        |
| setEdgeStyle,RCyjs-method . . . . .              | 50        |
| setGraph,RCyjs-method . . . . .                  | 51        |
| setNodeAttributes,RCyjs-method . . . . .         | 52        |
| setNodeBorderColor,RCyjs-method . . . . .        | 53        |
| setNodeBorderWidth,RCyjs-method . . . . .        | 53        |
| setNodeColor,RCyjs-method . . . . .              | 54        |
| setNodeColorRule,RCyjs-method . . . . .          | 55        |
| setNodeFontColor,RCyjs-method . . . . .          | 56        |
| setNodeFontSize,RCyjs-method . . . . .           | 57        |
| setNodeHeight,RCyjs-method . . . . .             | 57        |
| setNodeLabelAlignment,RCyjs-method . . . . .     | 58        |
| setNodeLabelRule,RCyjs-method . . . . .          | 59        |
| setNodeShape,RCyjs-method . . . . .              | 59        |
| setNodeSize,RCyjs-method . . . . .               | 60        |
| setNodeSizeRule,RCyjs-method . . . . .           | 61        |
| setNodeWidth,RCyjs-method . . . . .              | 62        |
| setPosition,RCyjs-method . . . . .               | 62        |
| setZoom,RCyjs-method . . . . .                   | 63        |
| sfn,RCyjs-method . . . . .                       | 64        |
| showAll,RCyjs-method . . . . .                   | 65        |
| showEdges,RCyjs-method . . . . .                 | 65        |
| showNodes,RCyjs-method . . . . .                 | 66        |
| simpleDemoGraph . . . . .                        | 67        |
| vAlign,RCyjs-method . . . . .                    | 67        |
| <b>Index</b>                                     | <b>69</b> |

---

addGraph,RCyjs-method    *addGraph*

---

## Description

addGraph send these nodes and edges (with attributes) to RCyjs for display

**Usage**

```
## S4 method for signature 'RCyjs'  
addGraph(obj, graph)
```

**Arguments**

|       |                   |
|-------|-------------------|
| obj   | an RCyjs instance |
| graph | a graphNEL        |

**Details**

This version transmits a graph (nodes, edges and attributes) to the browser by writing the data to a file, and sending that filename to be read in the browser by javascript.

**Value**

nothing

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  g <- simpleDemoGraph()  
  setGraph(rcy, g)  
}
```

---

addGraphFromFile,RCyjs-method  
*addGraphFromFile*

---

**Description**

addGraphFromFile add graph from specified file, which contains a cytoscape.js JSON graph

**Usage**

```
## S4 method for signature 'RCyjs'  
addGraphFromFile(obj, jsonFileName)
```

**Arguments**

|              |                   |
|--------------|-------------------|
| obj          | an RCyjs instance |
| jsonFileName | path to the file  |

**Details**

More description

**Value**

nothin

**Examples**

```
if(interactive()){  
  rcy <- RCyjs()  
  filename <- system.file(package="RCyjs", "extdata", "sampleGraph.json")  
  addGraphFromFile(rcy, filename)  
  layout(rcy, "cose")  
  fit(rcy, 200)  
}
```

---

clearSelection,RCyjs-method

*clearSelection*

---

**Description**

clearSelection deselect all selected nodes, all selected edges, or both

**Usage**

```
## S4 method for signature 'RCyjs'  
clearSelection(obj, which = "both")
```

**Arguments**

|       |                                                              |
|-------|--------------------------------------------------------------|
| obj   | an RCyjs object                                              |
| which | a character string: "both" (the default), "nodes" or "edges" |

**Value**

no return value

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  selectNodes(rcy, c("A", "B"))  
  clearSelection(rcy)  
}
```

---

|                 |                        |
|-----------------|------------------------|
| createTestGraph | <i>createTestGraph</i> |
|-----------------|------------------------|

---

### Description

createTestGraph With as many nodes and edges as you wish, but neither edge nor node attributes.

### Usage

```
createTestGraph(nodeCount, edgeCount)
```

### Arguments

|           |           |
|-----------|-----------|
| nodeCount | 1 or more |
| edgeCount | 0 or more |

### Value

a graphNEL with nodeCount nodes and edgeCount edges

### Examples

```
g <- createTestGraph(5, 3)
```

---

|                  |                                                                      |
|------------------|----------------------------------------------------------------------|
| dataFramesToJSON | <i>Create a cytoscape.js JSON graph from one or two data.frames.</i> |
|------------------|----------------------------------------------------------------------|

---

### Description

Create a cytoscape.js JSON graph from one or two data.frames.

### Usage

```
dataFramesToJSON(tbl.edges, tbl.nodes = NULL)
```

### Arguments

|           |                                                                                            |
|-----------|--------------------------------------------------------------------------------------------|
| tbl.edges | data.frame, with source, target and interaction columns, others option for edge attributes |
| tbl.nodes | data.frame, options, useful for orphan nodes, and necessary for adding node attributes     |

---

`deleteGraph,RCyjs-method`  
*deleteGraph*

---

## Description

`deleteGraph` Remove all nodes and edges, the elements of the current graph.

## Usage

```
## S4 method for signature 'RCyjs'
deleteGraph(obj)
```

## Arguments

`obj`                      `RCyjs` instance

## Details

This method will remove any previous graph in the browser

## Value

nothing

## See Also

[addGraph](#) [setGraph](#)

## Examples

```
if(interactive()){
  sampleGraph <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=sampleGraph)
  deleteGraph(rcy)
}
```

deleteSelectedNodes, RCyjs-method  
*deleteSelectedNodes*

---

**Description**

deleteSelectedNodes put somewhat more detailed description here

**Usage**

```
## S4 method for signature 'RCyjs'
deleteSelectedNodes(obj)
```

**Arguments**

obj                    an RCyjs instance

**Details**

multi-line description goes here with continuations on subsequent lines if you like

**Value**

explain what the method returns

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  target <- nodes(g)[1]
  selectNodes(rcy, target)
  deleteSelectedNodes(rcy)
}
```

---

eda                    eda

---

**Description**

eda retrieve the node/attribute-value pairs, for the specified node attribute category

**Usage**

```
eda(graph, edge.attribute.name)
```

**Arguments**

graph                    a graphNEL  
edge.attribute.name  
                         a character string

**Value**

character strings, the names of the unique edge attribute categories on the graph

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  edaNames(g) # discover the attribute category names  
  eda(g, "edgeType")  
  eda(g, "score")  
}
```

---

edaNames

*edaNames*

---

**Description**

edaNames the names of the unique edge attribute categories on the graph (not their values)

**Usage**

```
edaNames(graph)
```

**Arguments**

graph                    a graphNEL

**Value**

character strings, the names of the unique edge attribute categories on the graph

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  edaNames(g)  
}
```

---

|                  |            |
|------------------|------------|
| fit,RCyjs-method | <i>fit</i> |
|------------------|------------|

---

**Description**

fit zoom in (or out) to display all nodes in the current graph

**Usage**

```
## S4 method for signature 'RCyjs'
fit(obj, padding = 30)
```

**Arguments**

|         |                    |
|---------|--------------------|
| obj     | an RCyjs instance  |
| padding | numeric, in pixels |

**Value**

no return value

**Examples**

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  setZoom(rcy, 0.5) # zoom out
  fit(rcy)
}
```

---

|                           |                     |
|---------------------------|---------------------|
| fitSelection,RCyjs-method | <i>fitSelection</i> |
|---------------------------|---------------------|

---

**Description**

fitSelection zoom in to include only currently selected nodes

**Usage**

```
## S4 method for signature 'RCyjs'
fitSelection(obj, padding = 30)
```

**Arguments**

|         |                    |
|---------|--------------------|
| obj     | an RCyjs instance  |
| padding | numeric, in pixels |

**Value**

no return value

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  selectNodes(rcy, "A")  
  fitSelection(rcy, padding=100)  
}
```

---

*getEdgeCount,RCyjs-method*  
*getEdgeCount*

---

**Description**

`getEdgeCount` the number of edges in the current cytoscape.js graph

**Usage**

```
## S4 method for signature 'RCyjs'  
getEdgeCount(obj)
```

**Arguments**

obj                      RCyjs instance

**Value**

numeric count

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  getEdgeCount(rcy)  
}
```

---

*getJSON,RCyjs-method*    *getJSON*

---

**Description**

`getJSON` a JSON string from the browser, describing the graph in cytoscape.js terms

**Usage**

```
## S4 method for signature 'RCyjs'
getJSON(obj)
```

**Arguments**

`obj`                    an RCyjs instance

**Value**

a JSON string

**Examples**

```
if(interactive()){
  sampleGraph <- simpleDemoGraph()
  rcy <- RCyjs(title="getJSON", graph=sampleGraph)
  s <- getJSON(rcy)
  s.asList <- fromJSON(s) # easier to inspect if you wish to
}
```

---

*getLayoutStrategies,RCyjs-method*  
*getLayoutStrategies*

---

**Description**

`getLayoutStrategies` return a list of those currently offered

**Usage**

```
## S4 method for signature 'RCyjs'
getLayoutStrategies(obj)
```

**Arguments**

`obj`                    an RCyjs instance

**Value**

a list of character strings

**Examples**

```
if(interactive()){  
  g <- createTestGraph(nodeCount=20, edgeCount=20)  
  rcy <- RCyjs(title="layouts", graph=g)  
  strategies <- getLayoutStrategies(rcy)  
}
```

---

*getNodeCount,RCyjs-method*  
*getNodeCount*

---

**Description**

`getNodeCount` the number of nodes in the current cytoscape.js graph

**Usage**

```
## S4 method for signature 'RCyjs'  
getNodeCount(obj)
```

**Arguments**

`obj`                      `RCyjs` instance

**Value**

numeric count

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  getNodeCount(rcy)  
}
```

---

`getNodeNodes,RCyjs-method` *getNodeNodes*

---

### Description

`getNodeNodes` returns a data.frame, one row per node, providing id and (if present) name and label columns

### Usage

```
## S4 method for signature 'RCyjs'
getNodeNodes(obj, which = "all")
```

### Arguments

|                    |                                                         |
|--------------------|---------------------------------------------------------|
| <code>obj</code>   | an RCyjs instance                                       |
| <code>which</code> | a character string, either "all", "visible" or "hidden" |

### Details

Every node is guaranteed to have an "id" attribute. Because "name" and "label" are commonly used as well, they are returned as columns in the data.frame if present

### Value

a data.frame with at least an "id" column

### Examples

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  getNodeNodes(rcy)
}
```

---

`getPosition,RCyjs-method` *getPosition*

---

### Description

`getPosition` for all or specified nodes

### Usage

```
## S4 method for signature 'RCyjs'
getPosition(obj, nodeIDs = NA)
```

**Arguments**

obj                    an RCyjs instance  
nodeIDs                a vector of character strings, default NA

**Value**

a data.frame with "id", "x" and "y" columns

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="getPosition", graph=g)  
  layout(rcy, "cose")  
  tbl.pos <- getPosition(rcy)  
  tbl.posA <- getPosition(rcy, "A")  
}
```

---

getSelectedNodes,RCyjs-method  
*getSelectedNodes*

---

**Description**

getSelectedNodes get the selected nodes

**Usage**

```
## S4 method for signature 'RCyjs'  
getSelectedNodes(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

a data.frame with (at least) an id column

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  nodes.to.select <- getNodes(rcy)$id  
  selectNodes(rcy, nodes.to.select)  
}
```

---

`getSupportedEdgeDecoratorShapes,RCyjs-method`  
*getSupportedEdgeDecoratorShapes*

---

**Description**

`getSupportedEdgeDecoratorShapes` return a list of those currently offered

**Usage**

```
## S4 method for signature 'RCyjs'
getSupportedEdgeDecoratorShapes(obj)
```

**Arguments**

`obj`                      an RCyjs instance

**Value**

a list of character strings

**Examples**

```
if(interactive()){
  g <- createTestGraph(nodeCount=20, edgeCount=20)
  rcy <- RCyjs(title="shapes", graph=g)
  shapes <- getSupportedEdgeDecoratorShapes(rcy)
}
```

---

`getSupportedNodeShapes,RCyjs-method`  
*getSupportedNodeShapes*

---

**Description**

`getSupportedNodeShapes` return a list of those currently offered

**Usage**

```
## S4 method for signature 'RCyjs'
getSupportedNodeShapes(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

a list of character strings

**Examples**

```
if(interactive()){  
  g <- createTestGraph(nodeCount=20, edgeCount=20)  
  rcy <- RCyjs(title="shapes", graph=g)  
  shapes <- getSupportedNodeShapes(rcy)  
}
```

---

*getZoom,RCyjs-method*    *getZoom*

---

**Description**

*getZoom* learn the zoom level of the current display

**Usage**

```
## S4 method for signature 'RCyjs'  
getZoom(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

no return value

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  getZoom(rcy)  
  Sys.sleep(1)  
  setZoom(rcy, 5)  
  getZoom(rcy)  
}
```

---

`graphNELtoJSON.string` *Convert R graphNEL object to cytoscape.js JSON.*

---

**Description**

Convert R graphNEL object to cytoscape.js JSON.

**Usage**

```
graphNELtoJSON.string(g)
```

**Arguments**

`g` a graphNEL

**Examples**

```
## Not run:
g.json <- graphNELtoJSON.string(graphNEL())

## End(Not run)
```

---

`hAlign,RCyjs-method` *hAlign*

---

**Description**

`hAlign` horizontally align selected nodes

**Usage**

```
## S4 method for signature 'RCyjs'
hAlign(obj)
```

**Arguments**

`obj` an RCyjs instance

**Details**

The shared y coordinate will be the mean of the y coordinates of selected nodes. The x coordinates are preserved.

**Value**

no return value

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  layout(rcy, "cose")  
  selectNodes(rcy, nodes(g)[1:2])  
  hAlign(rcy)  
}
```

---

hideAllEdges,RCyjs-method  
*hideAllEdges*

---

**Description**

hideAllEdges

**Usage**

```
## S4 method for signature 'RCyjs'  
hideAllEdges(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

no return value

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  layout(rcy, "cose")  
  hideAllEdges()  
  showAll(rcy, "edges")  
}
```

---

hideEdges,RCyjs-method  
*hideEdges*

---

## Description

hideEdges hide all edges of the specified type

## Usage

```
## S4 method for signature 'RCyjs'
hideEdges(obj, edgeType)
```

## Arguments

|          |                    |
|----------|--------------------|
| obj      | an RCyjs instance  |
| edgeType | a character string |

## Details

edgeType is a crucial feature for RCyjs. We assume it is an attribute found on every edge in every graph.

## Value

no return value

## Examples

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  getNodes(rcy)
  edaNames(rcy)      # includes "edgeType"
  eda(rcy, "edgeType") # includes "phosphorylates"
  hideEdges(rcy, edgeType="phosphorylates")
  showEdges(rcy, edgeType="phosphorylates")
}
```

---

`hideNodes,RCyjs-method`*hideNodes*

---

## Description

hideNodes hide the named nodes from view

## Usage

```
## S4 method for signature 'RCyjs'
hideNodes(obj, nodeIDs)
```

## Arguments

obj                    an RCyjs instance

## Details

The hidden nodes are not deleted from the graph

## Value

no return value

## See Also

[showAll](#)

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  target <- nodes(g)[1]
  selectNodes(rcy, target)
  hideNodes(rcy)
  getNodes(rcy, "hidden")
  getNodes(rcy, "visible")
  showAll(rcy, which="nodes")
}
```

---

hideSelectedNodes,RCyjs-method  
*hideSelectedNodes*

---

## Description

hideSelectedNodes hide selected nodes from view

## Usage

```
## S4 method for signature 'RCyjs'  
hideSelectedNodes(obj)
```

## Arguments

obj                    an RCyjs instance

## Details

The hidden nodes are not deleted from the graph

## Value

no return value

## See Also

[showAll](#)

## Examples

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  target <- nodes(g)[1]  
  selectNodes(rcy, target)  
  hideSelectedNodes(rcy)  
  getNodes(rcy, "hidden")  
  getNodes(rcy, "visible")  
  showAll(rcy, which="nodes")  
}
```

---

```
invertNodeSelection,RCyjs-method
      invertNodeSelection
```

---

**Description**

invertNodeSelection deselect all selected nodes, select all previously unselected nodes

**Usage**

```
## S4 method for signature 'RCyjs'
invertNodeSelection(obj)
```

**Arguments**

obj                      an RCyjs instance

**Value**

no return value

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  target <- nodes(g)[1]
  selectNodes(rcy, target)
  invertNodeSelection(rcy)
}
```

---

```
layout,RCyjs-method      layout
```

---

**Description**

layout apply a layout algorithm to the current grap

**Usage**

```
## S4 method for signature 'RCyjs'
layout(obj, strategy = "random")
```

**Arguments**

obj                      an RCyjs instance  
 strategy                a character string, one of the supported algorithms

**Value**

explain what the method returns

**See Also**

[getLayoutStrategies](#)

**Examples**

```
if(interactive()){
  g <- createTestGraph(nodeCount=20, edgeCount=20)
  rcy <- RCyjs(title="layouts", graph=g)
  strategies <- getLayoutStrategies(rcy)
  for(strategy in strategies){
    layout(rcy, strategy)
    Sys.sleep(1)
  }
}
```

---

layoutSelectionInGrid,RCyjs-method  
*layoutSelectionInGrid*

---

**Description**

layoutSelectionInGrid arrange selected nodes in this region

**Usage**

```
## S4 method for signature 'RCyjs'
layoutSelectionInGrid(obj, x, y, w, h)
```

**Arguments**

|     |                                                            |
|-----|------------------------------------------------------------|
| obj | an RCyjs instance                                          |
| x   | numeric this will be the top left x coordinate of the grid |
| y   | numeric the top right                                      |
| w   | numeric width of the grid                                  |
| h   | numeric height of the grid                                 |

**Value**

no return value

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  loadStyleFile(rcy, system.file(package="RCyjs", "extdata", "sampleStyle2.js"));
  selectNodes(rcy, nodes(g))
  layoutSelectionInGrid(rcy, -1000, 10, 100, 400)
}

```

---

layoutSelectionInGridInferAnchor,RCyjs-method  
*layoutSelectionInGridInferAnchor*

---

**Description**

layoutSelectionInGridInferAnchor the top-most, left-most of the selected nodes is the anchor

**Usage**

```

## S4 method for signature 'RCyjs'
layoutSelectionInGridInferAnchor(obj, w, h)

```

**Arguments**

|     |                                     |
|-----|-------------------------------------|
| obj | an RCyjs instance                   |
| w   | numeric, the width of the grid box  |
| h   | numeric, the height of the grid box |

**Details**

anchor (the top left) of the grid is the location of the topmost/leftmost node, then arrange all the selected nodes in a box anchored here.

**Value**

explain what the method returns

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  loadStyleFile(rcy, system.file(package="RCyjs", "extdata", "sampleStyle2.js"));
}

```

```
selectNodes(rcy, nodes(g))
layoutSelectionInGrid(rcy, -1000, 10, 100, 400)
}
```

---

loadStyleFile,RCyjs-method  
*loadStyleFile*

---

### Description

loadStyleFile load a named JSON cytoscape.js style file into the browser

### Usage

```
## S4 method for signature 'RCyjs'
loadStyleFile(obj, filename)
```

### Arguments

|          |                                                 |
|----------|-------------------------------------------------|
| obj      | an RCyjs instance                               |
| filename | contains json in the proper cytoscape.js format |

### Value

nothing

### References

<https://js.cytoscape.org/#style>

Though we provide access to individual styling rules (see below) we often find it convenient to express all aspects of a visual style in a single JSON file

### Examples

```
if(interactive()){
  rcy <- demo()
  filename <- system.file(package="RCyjs", "extdata", "sampleStyle1.js");
  loadStyleFile(rcy, filename)
}
```

---

|     |            |
|-----|------------|
| noa | <i>noa</i> |
|-----|------------|

---

**Description**

noa retrieve the node/attribute-value pairs, for the specified node attribute category

**Usage**

```
noa(graph, node.attribute.name)
```

**Arguments**

|                     |                    |
|---------------------|--------------------|
| graph               | a graphNEL         |
| node.attribute.name | a character string |

**Value**

character strings, the names of the unique edge attribute categories on the graph

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  noa(g, "lfc")  
}
```

---

|          |                 |
|----------|-----------------|
| noaNames | <i>noaNames</i> |
|----------|-----------------|

---

**Description**

noaNames the names of the unique node attribute categories on the graph (not their values)

**Usage**

```
noaNames(graph)
```

**Arguments**

|       |            |
|-------|------------|
| graph | a graphNEL |
|-------|------------|

**Value**

character strings, the names of the unique node attribute categories on the graph

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  noaNames(g)
}
```

---

RCyjs-class

---

*Create an RCyjs object*


---

**Description**

The RCyjs class provides an R interface to cytoscape.js, a rich, interactive, full-featured, javascript network (graph) library. One constructs an RCyjs instance on a specified port (default 9000), the browser code is loaded, and a websocket connection opened.

**Usage**

```
RCyjs(
  portRange = 16000:16100,
  title = "RCyjs",
  graph = graphNEL(),
  quiet = TRUE
)
```

**Arguments**

|           |                                                                                       |
|-----------|---------------------------------------------------------------------------------------|
| portRange | The constructor looks for a free websocket port in this range. 16000:16100 by default |
| title     | Used for the web browser window, "RCyjs" by default                                   |
| graph     | a Biocondcutor graphNEL object                                                        |
| quiet     | A logical variable controlling verbosity during execution                             |

**Value**

An object of the RCyjs class

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  setNodeLabelRule(rcy, "label");
  setNodeSizeRule(rcy, "count", c(0, 30, 110), c(20, 50, 100));
  setNodeColorRule(rcy, "count", c(0, 100), c(colors$green, colors$red), mode="interpolate")
  redraw(rcy)
  layout(rcy, "cose")
}
```

---

`readAndStandardizeJSONNetworkFile`

*Read in a JSON network file, identify (or add) elements field return JSON*

---

**Description**

Read in a JSON network file, identify (or add) elements field return JSON

**Usage**

```
readAndStandardizeJSONNetworkFile(filename)
```

**Arguments**

|      |             |
|------|-------------|
| file | a json file |
|------|-------------|

---

`readAndStandardizeJSONStyleFile`

*Read in a JSON file, extract the selector elements, return JSON*

---

**Description**

Read in a JSON file, extract the selector elements, return JSON

**Usage**

```
readAndStandardizeJSONStyleFile(filename)
```

**Arguments**

|      |             |
|------|-------------|
| file | a json file |
|------|-------------|

---

redraw,RCyjs-method      *redraw*

---

### Description

redraw re-render the graph, using the latest style rules and assignments

### Usage

```
## S4 method for signature 'RCyjs'
redraw(obj)
```

### Arguments

obj                      an RCyjs instance

### Value

explain what the method returns

### Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  setNodeAttributes(rcy, "lfc", c("A", "B", "C"), c(0, 0, 0))
  redraw(rcy)
}
```

---

restoreLayout,RCyjs-method  
                            *restoreLayout*

---

### Description

restoreLayout restore a previously-saved layout

### Usage

```
## S4 method for signature 'RCyjs'
restoreLayout(obj, filename = "layout.RData")
```

**Arguments**

|          |                                            |
|----------|--------------------------------------------|
| obj      | an RCyjs instance                          |
| filename | a character string, default "layout.RData" |

**Value**

no return value

**See Also**

[saveLayout](#)

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  layout(rcy, "grid")  
  saveLayout(rcy, filename="gridLayout.RData")  
  layout(rcy, "circle")  
  restoreLayout(rcy, "gridLayout.RData")  
}
```

---

saveJPG,RCyjs-method    *saveJPG*

---

**Description**

saveJPG write current cytoscape view, at current resolution, to a JPG file.

**Usage**

```
## S4 method for signature 'RCyjs'  
saveJPG(obj, filename, resolutionFactor = 1)
```

**Arguments**

|                  |                                                                         |
|------------------|-------------------------------------------------------------------------|
| obj              | an RCyjs instance                                                       |
| filename         | a character string                                                      |
| resolutionFactor | numeric, default 1, higher values multiply resolution beyond screen dpi |

**Value**

no return value

## Examples

```
if(interactive()){
  rcy <- RCyjs(title="layouts", graph=createTestGraph(nodeCount=20, edgeCount=20))
  style.filename <- system.file(package="RCyjs", "extdata", "sampleStyle1.js");
  loadStyleFile(rcy, style.filename)
  layout(rcy, "cose")
  fit(rcy)
  filename <- tempfile(fileext=".jpg")
  saveJPG(rcy, filename, resolutionFactor)
}
```

---

saveLayout,RCyjs-method

*saveLayout*

---

## Description

saveLayout to a named file

## Usage

```
## S4 method for signature 'RCyjs'
saveLayout(obj, filename = "layout.RData")
```

## Arguments

|          |                           |
|----------|---------------------------|
| obj      | a RCyjs instance          |
| filename | "layout.RData" by default |

## Details

All node positions are saved to a functionally opaque RData object, in a file whose name you supply. These files are used by restoreLayout.

## Value

no return value

## See Also

[restoreLayout](#)

**Examples**

```

if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  layout(rcy, "grid")
  saveLayout(rcy, filename="gridLayout.RData")
  layout(rcy, "circle")
  restoreLayout(rcy, "gridLayout.RData")
}

```

---

savePNG,RCyjs-method    *savePNG*

---

**Description**

savePNG write current cytoscape view, at current resolution, to a PNG file.

**Usage**

```

## S4 method for signature 'RCyjs'
savePNG(obj, filename)

```

**Arguments**

|          |                    |
|----------|--------------------|
| obj      | an RCyjs instance  |
| filename | a character string |

**Value**

no return value

**Examples**

```

if(interactive()){
  rcy <- RCyjs(title="layouts", graph=createTestGraph(nodeCount=20, edgeCount=20))
  style.filename <- system.file(package="RCyjs", "extdata", "sampleStyle1.js");
  loadStyleFile(rcy, style.filename)
  layout(rcy, "cose")
  fit(rcy)
  filename <- tempfile(fileext=".png")
  savePNG(rcy, filename)
}

```

---

selectFirstNeighborsOfSelectedNodes,RCyjs-method  
*selectFirstNeighborsOfSelectedNodes*

---

**Description**

selectFirstNeighborsOfSelectedNodes

**Usage**

```
## S4 method for signature 'RCyjs'  
selectFirstNeighborsOfSelectedNodes(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

no return value

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  selectNodes(rcy, "A")  
  getSelectedNodes(rcy) # just one  
  selectFirstNeighborsOfSelectedNodes()  
  getSelectedNodes(rcy) # now three  
}
```

---

selectNodes,RCyjs-method  
*selectNodes*

---

**Description**

selectNodes by node id

**Usage**

```
## S4 method for signature 'RCyjs'  
selectNodes(obj, nodeIDs)
```

## Arguments

|         |                   |
|---------|-------------------|
| obj     | an RCyjs instance |
| nodeIDs | character strings |

## Value

no return value

## Examples

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  selectNodes(rcy, c("A", "B"))
}
```

---

setBackgroundColors,RCyjs-method  
*setBackgroundColors*

---

## Description

setBackgroundColors of the entire cytoscape.js div

## Usage

```
## S4 method for signature 'RCyjs'
setBackgroundColors(obj, newValue)
```

## Arguments

|          |                                         |
|----------|-----------------------------------------|
| obj      | an RCyjs instance                       |
| newValue | a character string, any valid CSS color |

## Value

no return value

## Examples

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  setBackgroundColors(rcy, "lightblue")
}
```

---

setDefaultEdgeColor,RCyjs-method  
*setDefaultEdgeColor*

---

**Description**

setDefaultEdgeColor

**Usage**

```
## S4 method for signature 'RCyjs'  
setDefaultEdgeColor(obj, newValue)
```

**Arguments**

|          |                                         |
|----------|-----------------------------------------|
| obj      | an RCyjs instance                       |
| newValue | a character string, any valid CSS color |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultNodeColor", graph=g)  
  layout(rcy, "cose")  
  setDefaultNodeFontColor(rcy, "red")  
}
```

---

setDefaultEdgeLineColor,RCyjs-method  
*setDefaultEdgeLineColor*

---

**Description**

setDefaultEdgeLineColor

**Usage**

```
## S4 method for signature 'RCyjs'  
setDefaultEdgeLineColor(obj, newValue)
```

### Arguments

|          |                                         |
|----------|-----------------------------------------|
| obj      | an RCyjs instance                       |
| newValue | a character string, and valid CSS color |

### Value

no value returned

### Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultEdgeLineColor", graph=g)
  layout(rcy, "cose")
  setDefaultEdgeLineColor(rcy, "red")
}
```

---

setDefaultEdgeLineStyle,RCyjs-method  
*setDefaultEdgeLineStyle*

---

### Description

setDefaultEdgeLineStyle put somewhat more detailed description here

### Usage

```
## S4 method for signature 'RCyjs'
setDefaultEdgeLineStyle(obj, newValue = c("solid", "dotted", "dashed"))
```

### Arguments

|          |                                                           |
|----------|-----------------------------------------------------------|
| obj      | an RCyjs instance                                         |
| newValue | a character string, one of "solid", "dotted", or "dashed" |

### Details

multi-line description goes here with continuations on subsequent lines if you like

### Value

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultEdgeLineStyle", graph=g)  
  layout(rcy, "cose")  
  setDefaultEdgeLineColor(rcy, "dashed")  
}
```

---

setDefaultEdgeSourceArrowColor, RCyjs-method  
*setDefaultEdgeSourceArrowColor*

---

**Description**

setDefaultEdgeSourceArrowColor

**Usage**

```
## S4 method for signature 'RCyjs'  
setDefaultEdgeSourceArrowColor(obj, newValue)
```

**Arguments**

|          |                                         |
|----------|-----------------------------------------|
| obj      | an RCyjs instance                       |
| newValue | a character string, and valid CSS color |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultEdgeSourceArrowColor", graph=g)  
  layout(rcy, "cose")  
  setDefaultEdgeSourceArrowColor(rcy, "red")  
}
```

---

setDefaultEdgeSourceArrowShape,RCyjs-method  
*setDefaultEdgeSourceArrowShape*

---

## Description

setDefaultEdgeSourceArrowShape put somewhat more detailed description here

## Usage

```
## S4 method for signature 'RCyjs'
setDefaultEdgeSourceArrowShape(
  obj,
  newValue = c("triangle", "triangle-tee", "triangle-cross", "triangle-backcurve", "vee",
    "tee", "square", "circle", "diamond", "none")
)
```

## Arguments

|          |                                                                                                                                                    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| obj      | an RCyjs instance                                                                                                                                  |
| newValue | a character string, one of "triangle", "triangle-tee", "triangle-cross", "triangle-backcurve", "vee", "tee", "square", "circle", "diamond", "none" |

## Details

multi-line description goes here with continuations on subsequent lines if you like

## Value

no value returned

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultEdgeSourceArrowShape", graph=g)
  layout(rcy, "cose")
  setDefaultEdgeSourceArrowShape(rcy, "tee")
}
```

---

```
setDefaultEdgeTargetArrowColor,RCyjs-method
      setDefaultEdgeTargetArrowColor
```

---

**Description**

setDefaultEdgeTargetArrowColor

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultEdgeTargetArrowColor(obj, newValue)
```

**Arguments**

|          |                                         |
|----------|-----------------------------------------|
| obj      | an RCyjs instance                       |
| newValue | a character string, and valid CSS color |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultEdgeTargetArrowColor", graph=g)
  layout(rcy, "cose")
  setDefaultEdgeTargetArrowColor(rcy, "red")
}
```

---

```
setDefaultEdgeTargetArrowShape,RCyjs-method
      setDefaultEdgeTargetArrowShape
```

---

**Description**

setDefaultEdgeTargetArrowShape put somewhat more detailed description here

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultEdgeTargetArrowShape(
  obj,
  newValue = c("triangle", "triangle-tee", "triangle-cross", "triangle-backcurve", "vee",
    "tee", "square", "circle", "diamond", "none")
)
```

## Arguments

|          |                                                                                                                                                    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| obj      | an RCyjs instance                                                                                                                                  |
| newValue | a character string, one of "triangle", "triangle-tee", "triangle-cross", "triangle-backcurve", "vee", "tee", "square", "circle", "diamond", "none" |

## Details

multi-line description goes here with continuations on subsequent lines if you like

## Value

no value returned

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultEdgeTargetArrowShape", graph=g)
  layout(rcy, "cose")
  setDefaultEdgeTargetArrowShape(rcy, "tee")
}
```

---

setDefaultEdgeWidth,RCyjs-method  
*setDefaultEdgeWidth*

---

## Description

setDefaultEdgeWidth in pixels

## Usage

```
## S4 method for signature 'RCyjs'
setDefaultEdgeWidth(obj, newValue)
```

## Arguments

|          |                   |
|----------|-------------------|
| obj      | an RCyjs instance |
| newValue | a numeric         |

## Value

no value returned

## Examples

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultEdgeWidth", graph=g)  
  layout(rcy, "cose")  
  setDefaultEdgeWidth(rcy, 1)  
}
```

---

setDefaultNodeBorderColor,RCyjs-method  
*setDefaultNodeBorderColor*

---

## Description

setDefaultNodeBorderColor put somewhat more detailed description here

## Usage

```
## S4 method for signature 'RCyjs'  
setDefaultNodeBorderColor(obj, newValue)
```

## Arguments

|          |                   |
|----------|-------------------|
| obj      | an RCyjs instance |
| newValue | any CSS color     |

## Value

no value returned

## Examples

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultNodeBorderColor", graph=g)  
  layout(rcy, "cose")  
  setDefaultNodeBorderColor(rcy, "red")  
}
```

---

setDefaultNodeBorderWidth,RCyjs-method  
*setDefaultNodeBorderWidth*

---

## Description

setDefaultNodeBorderWidth in pixels

## Usage

```
## S4 method for signature 'RCyjs'
setDefaultNodeBorderWidth(obj, newValue)
```

## Arguments

|          |                    |
|----------|--------------------|
| obj      | an RCyjs instance  |
| newValue | numeric, in pixels |

## Value

no value returned

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodeBorderWidth", graph=g)
  layout(rcy, "cose")
  setDefaultNodeBorderWidth(rcy, 2)
}
```

---

setDefaultNodeColor,RCyjs-method  
*setDefaultNodeColor*

---

## Description

setDefaultNodeColor put somewhat more detailed description here

## Usage

```
## S4 method for signature 'RCyjs'
setDefaultNodeColor(obj, newValue)
```

**Arguments**

|          |                                              |
|----------|----------------------------------------------|
| obj      | an RCyjs instance                            |
| newValue | a character string, any valid CSS color name |

**Details**

multi-line description goes here with continuations on subsequent lines if you like

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultNodeColor", graph=g)  
  layout(rcy, "cose")  
  setDefaultNodeColor(rcy, "lightblue")  
}
```

---

setDefaultNodeFontColor,RCyjs-method  
*setDefaultNodeFontColor*

---

**Description**

setDefaultNodeFontColor

**Usage**

```
## S4 method for signature 'RCyjs'  
setDefaultNodeFontColor(obj, newValue)
```

**Arguments**

|          |                   |
|----------|-------------------|
| obj      | an RCyjs instance |
| newValue | any CSS color     |

**Value**

no value returned

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodeColor", graph=g)
  layout(rcy, "cose")
  setDefaultNodeFontColor(rcy, "red")
}
```

---

```
setDefaultNodeFontSize,RCyjs-method
setDefaultNodeFontSize
```

---

## Description

setDefaultNodeFontSize put somewhat more detailed description here

## Usage

```
## S4 method for signature 'RCyjs'
setDefaultNodeFontSize(obj, newValue)
```

## Arguments

|          |                    |
|----------|--------------------|
| obj      | an RCyjs instance  |
| newValue | numeric, in points |

## Details

multi-line description goes here with continuations on subsequent lines if you like

## Value

no value returned

## Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodeFontSize", graph=g)
  layout(rcy, "cose")
  setDefaultNodeFontSize(rcy, 8)
}
```

---

setDefaultNodeHeight,RCyjs-method  
*setDefaultNodeHeight*

---

**Description**

setDefaultNodeHeight set all nodes to the same specified width, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultNodeHeight(obj, newValue)
```

**Arguments**

|          |                      |
|----------|----------------------|
| obj      | an RCyjs instance    |
| newValue | a numeric, in pixels |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodeHeight", graph=g)
  layout(rcy, "cose")
  setDefaultNodeHeight(rcy, 80)
}
```

---

setDefaultNodeShape,RCyjs-method  
*setDefaultNodeShape*

---

**Description**

setDefaultNodeShape change the shape of all nodes

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultNodeShape(
  obj,
  newValue = c("ellipse", "triangle", "rectangle", "roundrectangle",
    "bottomroundrectangle", "cutrectangle", "barrel", "rhomboid", "diamond", "pentagon",
    "hexagon", "concavehexagon", "heptagon", "octagon", "star", "tag", "vee")
)
```

**Arguments**

|          |                                                                                                                                                                                                                                                |
|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj      | an RCyjs instance                                                                                                                                                                                                                              |
| newValue | a character string, one of "ellipse", "triangle", "rectangle", "roundrectangle", "bottomroundrectangle", "cutrectangle", "barrel", "rhomboid", "diamond", "pentagon", "hexagon", "concavehexagon", "heptagon", "octagon", "star", "tag", "vee" |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodeShape", graph=g)
  layout(rcy, "cose")
  setDefaultNodeShape(rcy, "barrel")
}
```

---

setDefaultNodeSize,RCyjs-method  
*setDefaultNodeSize*

---

**Description**

setDefaultNodeSize set all nodes to the same specified size, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultNodeSize(obj, newValue)
```

**Arguments**

|          |                      |
|----------|----------------------|
| obj      | an RCyjs instance    |
| newValue | a numeric, in pixels |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultNodesSize", graph=g)  
  layout(rcy, "cose")  
  setDefaultNodeSize(rcy, 80)  
}
```

---

setDefaultNodeWidth,RCyjs-method  
*setDefaultNodeWidth*

---

**Description**

setDefaultNodeWidth set all nodes to the same specified width, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'  
setDefaultNodeWidth(obj, newValue)
```

**Arguments**

|          |                      |
|----------|----------------------|
| obj      | an RCyjs instance    |
| newValue | a numeric, in pixels |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setDefaultNodesWidth", graph=g)  
  layout(rcy, "cose")  
  setDefaultNodeWidth(rcy, 80)  
}
```

---

```
setDefaultStyle,RCyjs-method
      setDefaultStyle
```

---

**Description**

setDefaultStyle use some sensible rendering options for all elements of the graph

**Usage**

```
## S4 method for signature 'RCyjs'
setDefaultStyle(obj)
```

**Arguments**

obj                    an RCyjs instance

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setDefaultNodesSize", graph=g)
  layout(rcy, "cose")
  setDefaultStyle(rcy)
}
```

---

```
setEdgeAttributes,RCyjs-method
      setEdgeAttributes
```

---

**Description**

setEdgeAttributes on the graph in the browse

**Usage**

```
## S4 method for signature 'RCyjs'
setEdgeAttributes(obj, attribute, sourceNodes, targetNodes, edgeTypes, values)
```

**Arguments**

|             |                             |
|-------------|-----------------------------|
| obj         | an RCyjs instance           |
| attribute   | a character string          |
| sourceNodes | vector of character strings |
| targetNodes | vector of character strings |
| edgeTypes   | vector of character strings |
| values      | vector of character strings |

**Details**

Edges are specified by sourceNode/targetNode/edgeType triples.

**Value**

no return value

---

setEdgeStyle,RCyjs-method  
*setEdgeStyle*

---

**Description**

setEdgeStyle plain & fast (haystack) vs fancy & slower (bezier)

**Usage**

```
## S4 method for signature 'RCyjs'
setEdgeStyle(obj, mode = c("bezier", "haystack"))
```

**Arguments**

|      |                                                   |
|------|---------------------------------------------------|
| obj  | an RCyjs instance                                 |
| mode | a character string, either "bezier" or "haystack" |

**Details**

cytoscape.js offers two kinds of edge rendering - a tradeoff in richness and speed edge target decorations (arrows, tee, etc) are only rendered with the "bezier" style

**Value**

no return value

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  loadStyleFile(rcy, system.file(package="RCyjs", "extdata", "sampleStyle2.js"))
  setEdgeStyle(rcy, "bezier")
  redraw(rcy)
}

```

---

setGraph,RCyjs-method    *setGraph*

---

**Description**

setGraph Establish a new graph in RCyjs, removing any previous graph

**Usage**

```

## S4 method for signature 'RCyjs'
setGraph(obj, graph)

```

**Arguments**

|       |                |
|-------|----------------|
| obj   | RCyjs instance |
| graph | a graphNEL     |

**Details**

This method will remove any previous graph in the browser, adding a new one. Setting visual properties and performing layout must follow.

**Value**

nothing

**See Also**

[addGraph](#)

**Examples**

```

if(interactive()){
  sampleGraph <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo")
  setGraph(rcy, sampleGraph)
}

```

---

setNodeAttributes,RCyjs-method  
*setNodeAttributes*

---

## Description

setNodeAttributes put somewhat more detailed description here

## Usage

```
## S4 method for signature 'RCyjs'  
setNodeAttributes(obj, attribute, nodes, values)
```

## Arguments

|           |                                                                                     |
|-----------|-------------------------------------------------------------------------------------|
| obj       | an RCyjs instance                                                                   |
| attribute | a character string                                                                  |
| nodes     | character strings - node ids                                                        |
| values    | scalar values, all of one type (all numeric, or all character, or all integer, ...) |

## Details

multi-line description goes here with continuations on subsequent lines if you like

## Value

explain what the method returns

## Examples

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  layout(rcy, "cose")  
  fit(rcy, 100)  
  setNodeAttributes(rcy, "lfc", c("A", "B", "C"), c(0, 0, 0))  
  redraw(rcy)  
}
```

---

setNodeBorderColor,RCyjs-method  
*setNodeBorderColor*

---

**Description**

setNodeBorderColor set the specified nodes to the specified node border color

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeBorderColor(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                     |
|-----------|-------------------------------------|
| obj       | an RCyjs instance                   |
| nodeIDs   | a character string (one or more)    |
| newValues | legal CSS color names (one or more) |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodeBorderColor", graph=g)
  layout(rcy, "cose")
  setNodeBorderColor(rcy, "green")
}
```

---

setNodeBorderWidth,RCyjs-method  
*setNodeBorderWidth*

---

**Description**

setNodeBorderWidth set the specified nodes to the same specified node border width, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeBorderWidth(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                           |
|-----------|-----------------------------------------------------------|
| obj       | an RCyjs instance                                         |
| nodeIDs   | a character string (one or more)                          |
| newValues | numeric, in pixels (one, or as many as there are nodeIDs) |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodeBorderWidth", graph=g)
  layout(rcy, "cose")
  setNodeBorderWidth(rcy, 3)
}
```

---

setNodeColor,RCyjs-method  
*setNodeColor*

---

**Description**

setNodeColor set the specified nodes to the specifed color

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeColor(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                         |
|-----------|---------------------------------------------------------|
| obj       | an RCyjs instance                                       |
| nodeIDs   | a character string (one or more)                        |
| newValues | a character string, legal CSS color names (one or more) |

**Value**

no value returned

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodeColor", graph=g)
  layout(rcy, "cose")
  setNodeColor(rcy, 80)
}

```

---

```

setNodeColorRule,RCyjs-method
      setNodeColorRule

```

---

**Description**

setNodeColorRule control node color via values of the specified attribute

**Usage**

```

## S4 method for signature 'RCyjs'
setNodeColorRule(
  obj,
  attribute,
  control.points,
  colors,
  mode = c("interpolate", "lookup")
)

```

**Arguments**

|                |                                                                            |
|----------------|----------------------------------------------------------------------------|
| obj            | an RCyjs instance                                                          |
| attribute      | a character string, the node attribute category whose value controls color |
| control.points | a list of all possible values of the attribute                             |
| colors         | the corresponding node color, one specified for each of the control.points |
| mode           | a character string, either "interpolate" or "lookup"                       |

**Details**

for interpolate mode, in which the node attribute should be a continuously varying numerical quantity in-between colors are calculated for in-between values. for lookup mode, in which the node attribute is a discrete string variable, simple color lookup is performed.

**Value**

no return value

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  setNodeColorRule(rcy, "count", c(0, 100), c("green", "red"), mode="interpolate")
  redraw(rcy)
}

```

---

```

setNodeFontColor,RCyjs-method
      setNodeFontColor

```

---

**Description**

setNodeFontColor set the specified nodes to the same specified node font color

**Usage**

```

## S4 method for signature 'RCyjs'
setNodeFontColor(obj, nodeIDs, newValues)

```

**Arguments**

|           |                                                          |
|-----------|----------------------------------------------------------|
| obj       | an RCyjs instance                                        |
| nodeIDs   | a character string (one or more)                         |
| newValues | a character string, a legal CSS color name (one or more) |

**Value**

no value returned

**Examples**

```

if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodeFontColor", graph=g)
  layout(rcy, "cose")
  setNodeFontColor(rcy, "red")
}

```

---

setNodeFontSize,RCyjs-method  
*setNodeFontSize*

---

**Description**

setNodeFontSize set the specified nodes to the same specified node font size

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeFontSize(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                             |
|-----------|-------------------------------------------------------------|
| obj       | an RCyjs instance                                           |
| nodeIDs   | a character string (one or more)                            |
| newValues | a numeric, in pixels (one, or as many as there are nodeIDs) |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodeFontSize", graph=g)
  layout(rcy, "cose")
  setNodeFontSize(rcy, 5)
}
```

---

setNodeHeight,RCyjs-method  
*setNodeHeight*

---

**Description**

setNodeHeight set the specified nodes to the specified heights, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeHeight(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                             |
|-----------|-------------------------------------------------------------|
| obj       | an RCyjs instance                                           |
| nodeIDs   | a character string (one or more)                            |
| newValues | a numeric, in pixels (one, or as many as there are nodeIDs) |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setNodesHeight", graph=g)  
  layout(rcy, "cose")  
  setNodeHeight(rcy, 80)  
}
```

---

setNodeLabelAlignment,RCyjs-method  
*setNodeLabelAlignment*

---

**Description**

setNodeLabelAlignment put somewhat more detailed description here

**Usage**

```
## S4 method for signature 'RCyjs'  
setNodeLabelAlignment(obj, horizontal, vertical)
```

**Arguments**

|            |                   |
|------------|-------------------|
| obj        | an RCyjs instance |
| horizontal | character string  |
| vertical   | character string  |

**Details**

multi-line description goes here with continuations on subsequent lines if you like

**Value**

explain what the method returns

---

setNodeLabelRule,RCyjs-method  
*setNodeLabelRule*

---

**Description**

setNodeLabelRule put somewhat more detailed description here

**Usage**

```
## S4 method for signature 'RCyjs'  
setNodeLabelRule(obj, attribute)
```

**Arguments**

|           |                                                            |
|-----------|------------------------------------------------------------|
| obj       | an RCyjs instance                                          |
| attribute | a character string, the node attribute to display as label |

**Details**

multi-line description goes here with continuations on subsequent lines if you like

**Value**

explain what the method returns

**Examples**

```
if(interactive()){  
  g <- createTestGraph(nodeCount=20, edgeCount=20)  
  rcy <- RCyjs(title="layouts", graph=g)  
  setNodeLabelRule(rcy, "label");  
}
```

---

setNodeShape,RCyjs-method  
*setNodeShape*

---

**Description**

setNodeShape set the specified nodes to specified shapes

**Usage**

```
## S4 method for signature 'RCyjs'  
setNodeShape(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                                    |
|-----------|--------------------------------------------------------------------|
| obj       | an RCyjs instance                                                  |
| nodeIDs   | a character string (one or more)                                   |
| newValues | a character string, one of the legitimate cytoscape.js node shapes |

**Value**

no value returned

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="setNodeShape", graph=g)  
  layout(rcy, "cose")  
  setNodeShape(rcy, 80)  
}
```

---

setNodeSize,RCyjs-method

*setNodeSize*

---

**Description**

setNodeSize set the specified nodes to the specified sizes, in pixels

**Usage**

```
## S4 method for signature 'RCyjs'  
setNodeSize(obj, nodeIDs, newValues)
```

**Arguments**

|           |                                                             |
|-----------|-------------------------------------------------------------|
| obj       | an RCyjs instance                                           |
| nodeIDs   | a character string (one or more)                            |
| newValues | a numeric, in pixels (one, or as many as there are nodeIDs) |

**Value**

no value returned

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodesSize", graph=g)
  layout(rcy, "cose")
  setNodeSize(rcy, 80)
}
```

---

```
setNodeSizeRule,RCyjs-method
      setNodeSizeRule
```

---

**Description**

setNodeSizeRule control node size via values of the specified attribute

**Usage**

```
## S4 method for signature 'RCyjs'
setNodeSizeRule(obj, attribute, control.points, node.sizes)
```

**Arguments**

|                |                                                                           |
|----------------|---------------------------------------------------------------------------|
| obj            | an RCyjs instance                                                         |
| attribute      | a character string, the node attribute category whose value controls size |
| control.points | a list of values of the attribute                                         |
| node.sizes     | the corresponding node size, one specified for each of the control.points |

**Details**

actual node sizes are interpolated via the specified relationship of control.points node.sizes

**Value**

no return value

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  fit(rcy, 100)
  setNodeSizeRule(rcy, "count", c(0, 30, 110), c(20, 50, 100));
  redraw(rcy)
}
```

---

```
setNodeWidth,RCyjs-method
      setNodeWidth
```

---

### Description

setNodeWidth set the specified nodes to the specifed widths, in pixels

### Usage

```
## S4 method for signature 'RCyjs'
setNodeWidth(obj, nodeIDs, newValues)
```

### Arguments

|           |                                                             |
|-----------|-------------------------------------------------------------|
| obj       | an RCyjs instance                                           |
| nodeIDs   | a character string (one or more)                            |
| newValues | a numeric, in pixels (one, or as many as there are nodeIDs) |

### Value

no value returned

### Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="setNodesWidth", graph=g)
  layout(rcy, "cose")
  setNodeWidth(rcy, 80)
}
```

---

```
setPosition,RCyjs-method
      setPosition
```

---

### Description

setPosition of nodes by their id

### Usage

```
## S4 method for signature 'RCyjs'
setPosition(obj, tbl.pos)
```

**Arguments**

|         |                                           |
|---------|-------------------------------------------|
| obj     | an RCyjs instance                         |
| tbl.pos | a data.frame with three columns: id, x, y |

**Value**

no return value

**See Also**

[getPosition](#)

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="getPosition", graph=g)
  layout(rcy, "cose")
  tbl.pos <- getPosition(rcy)
  # shift all the nodes to the right
  tbl.pos$x <- tbl.pos$x + 50
  setPosition(rcy, tbl.pos)
}
```

---

setZoom,RCyjs-method    *setZoom*

---

**Description**

setZoom zoom in or out

**Usage**

```
## S4 method for signature 'RCyjs'
setZoom(obj, newValue)
```

**Arguments**

|          |                                                                                                     |
|----------|-----------------------------------------------------------------------------------------------------|
| obj      | an RCyjs instance                                                                                   |
| newValue | numeric, typically be 0.1 (zoomed way out, nodes are small) and 10 (zoomed way in, nodes are large) |

**Value**

no return value

**Examples**

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  setZoom(rcy, 0.2)
  Sys.sleep(1)
  setZoom(rcy, 5)
}
```

---

|                  |            |
|------------------|------------|
| sfn,RCyjs-method | <i>sfn</i> |
|------------------|------------|

---

**Description**

sfn select first neighbors of the currently selected nodes

**Usage**

```
## S4 method for signature 'RCyjs'
sfn(obj)
```

**Arguments**

obj                      an RCyjs instance

**Value**

no return value

**Examples**

```
if(interactive()){
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())
  selectNodes(rcy, "A")
  getSelectedNodes(rcy) # just one
  sfn()
  getSelectedNodes(rcy) # now three
}
```

---

showAll,RCyjs-method    *showAll*


---

**Description**

showAll show any hidden objects: nodes, edges, or both

**Usage**

```
## S4 method for signature 'RCyjs'
showAll(obj, which = c("both", "nodes", "edges"))
```

**Arguments**

obj                    an RCyjs instance

which                a character string, either "nodes", "edges" or "both"

**Value**

no return value

**Examples**

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  layout(rcy, "cose")
  selectNodes(rcy, getNodes(rcy)$id)
  hideSelectedNodes(rcy)
  showAll(rcy, "nodes")
}
```

---

showEdges,RCyjs-method  
                          *showEdges*


---

**Description**

showEdges if hidden, edges of the specified type will be made visible

**Usage**

```
## S4 method for signature 'RCyjs'
showEdges(obj, edgeType)
```

**Arguments**

|          |                    |
|----------|--------------------|
| obj      | an RCyjs instance  |
| edgeType | a character string |

**Details**

edgeType is a crucial feature for RCyjs. We assume it is an attribute found on every edge in every graph.

**Value**

no return value

**Examples**

```
if(interactive()){  
  rcy <- RCyjs(title="rcyjs demo", graph=simpleDemoGraph())  
  getNodes(rcy)  
  edaNames(rcy)      # includes "edgeType"  
  eda(rcy, "edgeType") # includes "phosphorylates"  
  hideEdges(rcy, edgeType="phosphorylates")  
  showEdges(rcy, edgeType="phosphorylates")  
}
```

---

showNodes,RCyjs-method

*showNodes*

---

**Description**

showNodes show the named nodes from view

**Usage**

```
## S4 method for signature 'RCyjs'  
showNodes(obj, nodeIDs)
```

**Arguments**

|     |                   |
|-----|-------------------|
| obj | an RCyjs instance |
|-----|-------------------|

**Value**

no return value

**See Also**

[showAll](#)

Examples

```
if(interactive()){
  g <- simpleDemoGraph()
  rcy <- RCyjs(title="rcyjs demo", graph=g)
  target <- nodes(g)[1]
  hideNodes(rcy, "A")
  getNodes(rcy, "hidden")
  getNodes(rcy, "visible")
  showNodes(rcy, "A")
  getNodes(rcy, "visible")
}
```

---

|                 |                        |
|-----------------|------------------------|
| simpleDemoGraph | <i>simpleDemoGraph</i> |
|-----------------|------------------------|

---

Description

simpleDemoGraph

Usage

simpleDemoGraph()

Value

a graphNEL with 3 nodes and 3 edges

Examples

```
g <- simpleDemoGraph()
```

---

|                     |               |
|---------------------|---------------|
| vAlign,RCyjs-method | <i>vAlign</i> |
|---------------------|---------------|

---

Description

vAlign vertically align selected nodes

Usage

```
## S4 method for signature 'RCyjs'
vAlign(obj)
```

**Arguments**

`obj`                      an RCyjs instance

**Details**

The shared x coordinate will be the mean of the x coordinates of selected nodes. The y coordinates are preserved.

**Value**

no return value

**Examples**

```
if(interactive()){  
  g <- simpleDemoGraph()  
  rcy <- RCyjs(title="rcyjs demo", graph=g)  
  layout(rcy, "cose")  
  selectNodes(rcy, nodes(g)[1:2])  
  vAlign(rcy)  
}
```

# Index

.RCyjs (RCyjs-class), 28

addGraph, 7, 51

addGraph (addGraph, RCyjs-method), 3

addGraph, RCyjs-method, 3

addGraphFromFile

    (addGraphFromFile, RCyjs-method), 4

addGraphFromFile, RCyjs-method, 4

clearSelection

    (clearSelection, RCyjs-method), 5

clearSelection, RCyjs-method, 5

createTestGraph, 6

dataFramesToJSON, 6

deleteGraph (deleteGraph, RCyjs-method), 7

deleteGraph, RCyjs-method, 7

deleteSelectedNodes

    (deleteSelectedNodes, RCyjs-method), 8

deleteSelectedNodes, RCyjs-method, 8

eda, 8

edaNames, 9

fit (fit, RCyjs-method), 10

fit, RCyjs-method, 10

fitSelection

    (fitSelection, RCyjs-method), 10

fitSelection, RCyjs-method, 10

getEdgeCount

    (getEdgeCount, RCyjs-method), 11

getEdgeCount, RCyjs-method, 11

getJSON (getJSON, RCyjs-method), 12

getJSON, RCyjs-method, 12

getLayoutStrategies, 24

getLayoutStrategies

    (getLayoutStrategies, RCyjs-method), 12

getLayoutStrategies, RCyjs-method, 12

getNodeCount

    (getNodeCount, RCyjs-method), 13

getNodeCount, RCyjs-method, 13

getNodes (getNodes, RCyjs-method), 14

getNodes, RCyjs-method, 14

getPosition, 63

getPosition (getPosition, RCyjs-method), 14

getPosition, RCyjs-method, 14

getSelectedNodes

    (getSelectedNodes, RCyjs-method), 15

getSelectedNodes, RCyjs-method, 15

getSupportedEdgeDecoratorShapes

    (getSupportedEdgeDecoratorShapes, RCyjs-method), 16

getSupportedEdgeDecoratorShapes, RCyjs-method, 16

getSupportedNodeShapes

    (getSupportedNodeShapes, RCyjs-method), 16

getSupportedNodeShapes, RCyjs-method, 16

getZoom (getZoom, RCyjs-method), 17

getZoom, RCyjs-method, 17

graphNELtoJSON.string, 18

hAlign (hAlign, RCyjs-method), 18

hAlign, RCyjs-method, 18

hideAllEdges

    (hideAllEdges, RCyjs-method), 19

hideAllEdges, RCyjs-method, 19

hideEdges (hideEdges, RCyjs-method), 20

hideEdges, RCyjs-method, 20

hideNodes (hideNodes, RCyjs-method), 21

hideNodes, RCyjs-method, 21

- hideSelectedNodes  
    (hideSelectedNodes,RCyjs-method),  
    22
- hideSelectedNodes,RCyjs-method, 22
- invertNodeSelection  
    (invertNodeSelection,RCyjs-method),  
    23
- invertNodeSelection,RCyjs-method, 23
- layout (layout,RCyjs-method), 23
- layout,RCyjs-method, 23
- layoutSelectionInGrid  
    (layoutSelectionInGrid,RCyjs-method),  
    24
- layoutSelectionInGrid,RCyjs-method, 24
- layoutSelectionInGridInferAnchor  
    (layoutSelectionInGridInferAnchor,RCyjs-method),  
    25
- layoutSelectionInGridInferAnchor,RCyjs-method, 25
- loadStyleFile  
    (loadStyleFile,RCyjs-method),  
    26
- loadStyleFile,RCyjs-method, 26
- noa, 27
- noaNames, 27
- RCyjs (RCyjs-class), 28
- RCyjs-class, 28
- readAndStandardizeJSONNetworkFile, 29
- readAndStandardizeJSONStyleFile, 29
- redraw (redraw,RCyjs-method), 30
- redraw,RCyjs-method, 30
- restoreLayout, 32
- restoreLayout  
    (restoreLayout,RCyjs-method),  
    30
- restoreLayout,RCyjs-method, 30
- saveJPG (saveJPG,RCyjs-method), 31
- saveJPG,RCyjs-method, 31
- saveLayout, 31
- saveLayout (saveLayout,RCyjs-method), 32
- saveLayout,RCyjs-method, 32
- savePNG (savePNG,RCyjs-method), 33
- savePNG,RCyjs-method, 33
- selectFirstNeighborsOfSelectedNodes  
    (selectFirstNeighborsOfSelectedNodes,RCyjs-method),  
    34
- selectFirstNeighborsOfSelectedNodes,RCyjs-method,  
    34
- selectNodes (selectNodes,RCyjs-method),  
    34
- selectNodes,RCyjs-method, 34
- setBackgroundColor  
    (setBackgroundColor,RCyjs-method),  
    35
- setBackgroundColor,RCyjs-method, 35
- setDefaultEdgeColor  
    (setDefaultEdgeColor,RCyjs-method),  
    36
- setDefaultEdgeColor,RCyjs-method, 36
- setDefaultEdgeLineColor  
    (setDefaultEdgeLineColor,RCyjs-method),  
    36
- setDefaultEdgeLineColor,RCyjs-method,  
    36
- setDefaultEdgeLineStyle  
    (setDefaultEdgeLineStyle,RCyjs-method),  
    37
- setDefaultEdgeLineStyle,RCyjs-method,  
    37
- setDefaultEdgeSourceArrowColor  
    (setDefaultEdgeSourceArrowColor,RCyjs-method),  
    38
- setDefaultEdgeSourceArrowColor,RCyjs-method,  
    38
- setDefaultEdgeSourceArrowShape  
    (setDefaultEdgeSourceArrowShape,RCyjs-method),  
    39
- setDefaultEdgeSourceArrowShape,RCyjs-method,  
    39
- setDefaultEdgeTargetArrowColor  
    (setDefaultEdgeTargetArrowColor,RCyjs-method),  
    40
- setDefaultEdgeTargetArrowColor,RCyjs-method,  
    40
- setDefaultEdgeTargetArrowShape  
    (setDefaultEdgeTargetArrowShape,RCyjs-method),  
    40
- setDefaultEdgeTargetArrowShape,RCyjs-method,  
    40
- setDefaultEdgeWidth  
    (setDefaultEdgeWidth,RCyjs-method),

- 41
- setDefaultEdgeWidth, RCyjs-method, 41
- setDefaultNodeBorderColor
  - (setDefaultNodeBorderColor, RCyjs-method), 42
- setDefaultNodeBorderColor, RCyjs-method, 42
- setDefaultNodeBorderWidth
  - (setDefaultNodeBorderWidth, RCyjs-method), 43
- setDefaultNodeBorderWidth, RCyjs-method, 43
- setDefaultNodeColor
  - (setDefaultNodeColor, RCyjs-method), 43
- setDefaultNodeColor, RCyjs-method, 43
- setDefaultNodeFontColor
  - (setDefaultNodeFontColor, RCyjs-method), 44
- setDefaultNodeFontColor, RCyjs-method, 44
- setDefaultNodeFontSize
  - (setDefaultNodeFontSize, RCyjs-method), 45
- setDefaultNodeFontSize, RCyjs-method, 45
- setDefaultNodeHeight
  - (setDefaultNodeHeight, RCyjs-method), 46
- setDefaultNodeHeight, RCyjs-method, 46
- setDefaultNodeShape
  - (setDefaultNodeShape, RCyjs-method), 46
- setDefaultNodeShape, RCyjs-method, 46
- setDefaultNodeSize
  - (setDefaultNodeSize, RCyjs-method), 47
- setDefaultNodeSize, RCyjs-method, 47
- setDefaultNodeWidth
  - (setDefaultNodeWidth, RCyjs-method), 48
- setDefaultNodeWidth, RCyjs-method, 48
- setDefaultStyle
  - (setDefaultStyle, RCyjs-method), 49
- setDefaultStyle, RCyjs-method, 49
- setEdgeAttributes
  - (setEdgeAttributes, RCyjs-method), 49
- setEdgeAttributes, RCyjs-method, 49
- setEdgeStyle
  - (setEdgeStyle, RCyjs-method), 50
- setEdgeStyle, RCyjs-method, 50
- setGraph, 7
- setGraph (setGraph, RCyjs-method), 51
- setGraph, RCyjs-method, 51
- setNodeAttributes
  - (setNodeAttributes, RCyjs-method), 52
- setNodeAttributes, RCyjs-method, 52
- setNodeBorderColor
  - (setNodeBorderColor, RCyjs-method), 53
- setNodeBorderColor, RCyjs-method, 53
- setNodeBorderWidth
  - (setNodeBorderWidth, RCyjs-method), 53
- setNodeBorderWidth, RCyjs-method, 53
- setNodeColor
  - (setNodeColor, RCyjs-method), 54
- setNodeColor, RCyjs-method, 54
- setNodeColorRule
  - (setNodeColorRule, RCyjs-method), 55
- setNodeColorRule, RCyjs-method, 55
- setNodeFontColor
  - (setNodeFontColor, RCyjs-method), 56
- setNodeFontColor, RCyjs-method, 56
- setNodeFontSize
  - (setNodeFontSize, RCyjs-method), 57
- setNodeFontSize, RCyjs-method, 57
- setNodeHeight
  - (setNodeHeight, RCyjs-method), 57
- setNodeHeight, RCyjs-method, 57
- setNodeLabelAlignment
  - (setNodeLabelAlignment, RCyjs-method), 58
- setNodeLabelAlignment, RCyjs-method, 58
- setNodeLabelRule
  - (setNodeLabelRule, RCyjs-method), 59
- setNodeLabelRule, RCyjs-method, 59
- setNodeShape

(setNodeShape,RCyjs-method), [59](#)  
setNodeShape,RCyjs-method, [59](#)  
setNodeSize (setNodeSize,RCyjs-method),  
[60](#)  
setNodeSize,RCyjs-method, [60](#)  
setNodeSizeRule  
(setNodeSizeRule,RCyjs-method),  
[61](#)  
setNodeSizeRule,RCyjs-method, [61](#)  
setNodeWidth  
(setNodeWidth,RCyjs-method), [62](#)  
setNodeWidth,RCyjs-method, [62](#)  
setPosition (setPosition,RCyjs-method),  
[62](#)  
setPosition,RCyjs-method, [62](#)  
setZoom (setZoom,RCyjs-method), [63](#)  
setZoom,RCyjs-method, [63](#)  
sfn (sfn,RCyjs-method), [64](#)  
sfn,RCyjs-method, [64](#)  
showAll, [21](#), [22](#), [66](#)  
showAll (showAll,RCyjs-method), [65](#)  
showAll,RCyjs-method, [65](#)  
showEdges (showEdges,RCyjs-method), [65](#)  
showEdges,RCyjs-method, [65](#)  
showNodes (showNodes,RCyjs-method), [66](#)  
showNodes,RCyjs-method, [66](#)  
simpleDemoGraph, [67](#)  
  
vAlign (vAlign,RCyjs-method), [67](#)  
vAlign,RCyjs-method, [67](#)