RCytoscape

April 20, 2011

`clearMsg` *clearMsg*

Description

Clears any current message in the Cytoscape Desktop status bar.

Usage

```
clearMsg(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

Nothing.

Author(s)

Paul Shannon

See Also

msg

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
clearMsg (cw)
```

clearSelection *clearSelection*

Description

If any nodes are selected in the current Cytocape window, they will be unselected.

Usage

```
clearSelection(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
selectNodes (cw, 'A')
print (getSelectedNodeCount (cw))   # should be 1
clearSelection (cw)
print (getSelectedNodeCount (cw))   # should be 0
```

createWindow *createWindow*

Description

Request that Cytoscape create a new window for the supplied CytoscapeWindowClass object. It will hold a new network, using the title supplied when the object's constructor was called.

This method will probably not often be useful: it is called behind the scenes by the CytoscapeWindow constructor unless you specify (in calling the constructor) 'create.window=FALSE'. In that case, or if you interactively delete the window in Cytoscape, or if you call the 'destroyWindow' or 'destroyAllWindows' methods, you can create a new window by calling this method.

Usage

```
createWindow(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Paul Shannon

---

cy2.edge.names *cy2.edge.names*

Description

Bioconductor graph edges are named, i.e., A~B. The same edge in the Cytoscape domain would be 'A (<edgeType>) B', where '<edgeType>' might be 'phosphorylates' or 'represses'.

Usage

```
cy2.edge.names(graph)
```

Arguments

graph	

Value

A named list, in with Cytoscape edges names are the content, and bioc graph edge names are their names.

Author(s)

Paul Shannon

Examples

```
g <- makeSimpleGraph ()
cy2.edge.names (g)
#                     A~B                         B~C                         C~A
#  "A (phosphorylates) B" "B (synthetic lethal) C"        "C (undefined) A"
```

CytoscapeWindowClass-class
Class "CytoscapeWindowClass"

Description

A class providing access to the Cytoscape application.

Slots

`title`: An `attrData` the name of the window.

`window.id`: An `attrData` Cytoscape's identifier.

`graph`: An `attrData` a graph instance.

`uri`: An `attrData` the address of the Cytoscape XMLRPC server.

Methods

createWindow

destroyWindow

destroyAllWindows

displayGraph

firstNeighbors

getArrowShapes

getLayoutNames

getLineStyles

getNodeShapes

getWindowCount

Author(s)

Paul Shannon

Examples

```
   # create a CytoscapeWindowClass object by calling the constructor
c2 <- CytoscapeWindow ('demo', makeSimpleGraph ())
```

CytoscapeWindow *CytoscapeWindow*

Description

The constructor for the CytoscapeWindowClass

Usage

```
CytoscapeWindow(title = "default", graph = new("graphNEL", edgemode='directed'),
```

Arguments

title	A character string, this is the name you will see on the Cytoscape network window. Multiple windows with the same name are currently allowed.
graph	A Bioconductor graph.
host	Defaults to 'localhost', this is the domain name of a machine which is running Cytoscape with the appropriate XMLRPC server plugin.
rpcPort	Defaults to 9000, this may be any port to which the CytoscapeRPC server is listening.
create.window	Defaults to TRUE, but if you want a CytoscapeWindow just to call what in Java we would call 'class methods' – getWindowList () for instance, a CytoscapeWindow without an actual window can be useful.

Value

An object of the CytoscapeWindow Class.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('demo', new ('graphNEL'))
```

destroyAllWindows *destroyAllWindows*

Description

Remove and delete all the network windows currently held by Cytoscape. Note that though this is a CytoscapeWindowClass method, its affects ALL currently active CytoscapeWindowClass windows, not just its own window.

Usage

```
destroyAllWindows(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Paul Shannon

Examples

```
cw1 <- CytoscapeWindow ('cw1', makeSimpleGraph ())
cw2 <- CytoscapeWindow ('cw2', makeSimpleGraph ())
destroyAllWindows (cw1)
```

---

destroyWindow *destroyWindow*

Description

Remove and delete Cytoscape's (copy of the) network and window for the specified CytoscapeWindowClass object.

Usage

```
destroyWindow(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
destroyWindow (cw)
```

displayGraph *displayGraph*

Description

This method transmits the CytoscapeWindowClass's graph data, from R to Cytoscape: nodes, edges, node and edge attributes, and displays it in a window titled as specified by the objects 'title' slot. With large graphs, this transmission may take a while. (todo: provide a few timing examples.) The resulting view, in Cytoscape, of the network will need layout and vizmap rendering; layout so that all the nodes and edges can be seen; rendering so that data attributes can control the appearance of the the nodes and edges.

Usage

```
displayGraph(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

Nothing.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
```

---

dockPanel *dockPanel*

Description

The specified panel is returned to its 'home' position in the Cytoscape Desktop if it had been previously floating or hidden. The panelName parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'.

Usage

```
dockPanel(obj, panelName)
```

Arguments

obj	a CytoscapeWindowClass object.
panelName	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Paul Shannon

See Also

floatPanel hidePanel

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
 # or
cw <- CytoscapeWindow ('headless', create.window=FALSE)
dockPanel (cw, 'Control Panel')
 # or
dockPanel (cw, 'c')
```

---

eda.names *eda.names*

Description

Retrieve the names of the edge attributes in the specified graph. These are typically strings like 'score', 'weight', 'link', and (strongly recommended when you create a graph) 'edgeType'. Once you are reminded of the names of the edge attributes, you can use the method 'eda' to get all the values of this attribute for the edges in the graph.

Usage

```
eda.names (graph)
```

Arguments

graph	typically, a bioc graphNEL)

Value

A list, the contents of which are the attribute values, the names of which are the names of the edges.

Author(s)

Paul Shannon

See Also

eda

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
eda.names (cw@graph)
 # "edgeType" "score"    "misc"
```

`eda` *eda*

Description

Obtain the value of the specified edge attribute for every edge in the graph.

Usage

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

Arguments

`graph`	typically, a bioc graphNEL object
`edge.attribute.name`	a character string

Details

The edge.attribute.name may be obtained from the function, eda.names.

Value

A list, the contents of which are the attribute values, the names of which are the names of the edges.

Author(s)

Paul Shannon

See Also

eda.names

Examples

```
cw <- CytoscapeWindow ('demo', graph=makeSimpleGraph(), create.window=FALSE)
eda (cw@graph, 'edgeType')

## The function is currently defined as
function (graph, edge.attribute.name)
{
  unlist (sapply (names (edgeData (graph)), function (n) edgeData (graph)[[n]][[edge.attr

  } # eda
```

firstNeighbors *firstNeighbors*

Description

Returns the first neighbors of the specified node, using the names with which the original R graph was built. Nodes are often displayed in Cytoscape using a label which is different than the node's true name; be careful of this distinction.

Usage

```
firstNeighbors(obj, nodeName)
```

Arguments

obj	a CytoscapeWindowClass object.
nodeName	a character string

Value

A list of the names of nodes which share an edge with the supplied nodeName.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
firstNeighbors (cw, 'A')
  # $right
  #  [1] "B"
  # $left
  #  [1] "C"
```

floatPanel *floatPanel*

Description

The specified panel will 'float' detached from its 'home' position in the Cytoscape Desktop. As of this writing (10 aug 2010) the panel will tenaciously claim the topmost (visual) position on the screen... The panelName parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'.

Usage

```
floatPanel(obj, panelName)
```

Arguments

obj	a CytoscapeWindowClass object.
panelName	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Paul Shannon

See Also

hidePanel dockPanel

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
  # or, if you only want a CytoscapeWindow for control purposes, not
  # for display:
cw <- CytoscapeWindow ('headless', create.window=FALSE)
floatPanel (cw, 'Control Panel')
  # or with less typing
floatPanel (cw, 'c')
```

---

getAllEdges *getAllEdges*

Description

Retrieve all edges in the current graph, expressed in the standard Cytoscape notation.

Usage

```
getAllEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
  # [1] "C (undefined) A"  "B (synthetic lethal) C" "A (phosphorylates) B"
```

getAllNodes *getAllNodes*

Description

Retrieve the identifiers of all the nodes in the current graph - a list of strings.

Usage

```
getAllNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings. Note that node names are returned – their original and primary identifiers – and that these may be different from the node labels that you see when you look at the graph in Cytoscape.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getAllNodes (cw)
  #  [1] "C" "B" "A"
```

getArrowShapes *getArrowShapes*

Description

Retrieve the names of the currently supported 'arrows' – the decorations can (optionally) appear at the ends of edges, adjacent to the nodes they connect, and conveying information about the nature of the nodes' relationship. of strings.

Usage

```
getArrowShapes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings, e.g., 'WHITE_DIAMOND', 'BLACK_T'

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getArrowShapes (cw)
  # [1] "No Arrow" "Diamond" "Delta" "Arrow" "T" "Circle" "Half Arrow Top"  "Half Arrow
```

---

getAttributeClassNames
getAttributeClassNames

Description

Retrieve the names of the recognized and supported names for the class of any node or edge attribute. Two or three options are provided for each of the basic types, with the intention that you can use names that seem natural to you, and RCytoscape will recognize them.

Usage

```
getAttributeClassNames(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

A list of character strings group, e.g., "floating|numeric|double", "integer|int", "string|char|character"

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getAttributeClassNames (cw)
 # [1] "floating|numeric|double" "integer|int"             "string|char|character"
```

getGraph *getGraph*

Description

Returns the bioconductor graph object which belongs to the specified CytoscapeWindow object

Usage

```
getGraph(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A graph object.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
print (getGraph (cw))
```

getLayoutNames *getLayoutNames*

Description

Retrieve the names of the currently supported layout algorithms. These may be used in subsequent calls to the 'layout' function. Note that some of the more attractive layout options, from yFiles, cannot be run except from the user interface; their names do not appear here.

Usage

```
getLayoutNames(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings, e.g., "jgraph-circle" "attribute-circle" "jgraph-annealing"

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getLayoutNames (cw)
 #  [1] "jgraph-circle" "attribute-circle" "jgraph-annealing"  ...
```

`getLineStyles` *getLineStyles*

Description

Retrieve the names of the currently supported line types – values which can be used to render edges, and thus can be used in calls to 'setEdgeLineStyleRule'

Usage

```
getLineStyles(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

A list of character strings, e.g., 'SOLID', 'DOT'

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getLineStyles (cw)
#  [1] "SOLID" "LONG_DASH" "EQUAL_DASH" ...
```

getNodeShapes *getNodeShapes*

Description

Retrieve the names of the currently supported node shapes, which can then be used in calls to setNodeShapeRule and setDefaultVizMapValue

Usage

```
getNodeShapes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings, e.g., 'trapezoid', 'ellipse', 'rect'

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
getNodeShapes(cw)
 # "trapezoid" "round_rect" "ellipse" "triangle"   "rect_3d" "diamond" "parallelogram"
```

getSelectedNodeCount *getSelectedNodeCount*

Description

Returns the number of node currently selected.

Usage

```
getSelectedNodeCount(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

An integer.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
  # in Cytoscape, interactively select two nodes, or
selectNodes (cw, c ('A','B'))
getSelectedNodeCount (cw)
# [1] 2
```

---

getSelectedNodes *getSelectedNodes*

Description

Retrieve the identifiers of all the nodes selected in the current graph.

Usage

```
getSelectedNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A list of character strings.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
  # in Cytoscape, interactively select two nodes, or
selectNodes (cw, c ('A','B'))
getSelectedNodes (cw)
# [1] "A" "B"
```

getWindowCount *getWindowCount*

Description

Returns the number of windows which currently exist in the Cytoscape Desktop.

Usage

```
getWindowCount(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

An integer.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
count.at.start = getWindowCount (cw)
cw2 <- CytoscapeWindow ('test1', graph=makeSimpleGraph())
cw3 <- CytoscapeWindow ('test2', graph=makeSimpleGraph())
getWindowCount (cw)
  # should be two greater than 'count.at.start'
```

getWindowList *getWindowList*

Description

Returns a named list of windows in the current Cytoscape Desktop.

Usage

```
getWindowList(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

A named list, in which the values are the titles of the windows; the names of the list are integers.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('toast', graph=makeSimpleGraph())
getWindowList (cw)   # assuming 'test' already exists, you will see:
  #    39     38
  # "test" "toast"
```

`hidePanel` *hidePanel*

Description

The specified panel will be hidden, and no longer visible in the Cytoscape Desktop of, if floating, elsewhere on the computer screen. The `panelName` parameter is very flexible: a match is defined as a case-independent match of the supplied panelName to any starting characters in the actual panelName. Thus, 'd' and 'DA' both identify 'Data Panel'.

Usage

```
hidePanel(obj, panelName)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`panelName`	a character string, providing a partial or complete case-independent match to the start of the name of an actual panel.

Value

Nothing.

Author(s)

Paul Shannon

See Also

floatPanel dockPanel

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
 # or
cw <- CytoscapeWindow ('headless', create.window=FALSE)
hidePanel (cw, 'Control Panel')
 # or
hidePanel (cw, 'c')
```

hideSelectedNodes *hideSelectedNodes*

Description

Hide (but do not delete) the currently selected nodes. 'Unhide' is supposed to return them to view, but this is broken in Cytoscape 2.7.

Usage

```
hideSelectedNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

unhideAll

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
selectNodes (cw, c ('A', 'B'))
hideSelectedNodes (cw)
unhideAll (cw)
 # alas, Cytoscape requires that you render these nodes, and redo the
 # layout, no that they are visible again
redraw (cw)
layout (cw, 'jgraph-spring')
```

initEdgeAttribute *initEdgeAttribute*

Description

Create the edge attribute slot that the Bioconductor graph class requires, including a default value, and then specifying what the base type (or 'class') is – 'char', 'integer', or 'numeric' – which is needed by RCytoscape. This method converts these standard R data type names, to the forms needed by Cytoscape.

Usage

```
initEdgeAttribute(graph, attribute.name, attribute.type, default.value)
```

Arguments

graph	a Bioconductor graph object.
attribute.name	a string, the name of the new edge attribute.
attribute.type	a string, either 'char', 'integer', or 'numeric'
default.value	something sensible, of the right type

Value

Returns the modified graph.

Author(s)

Paul Shannon

See Also

initNodeAttribute makeSimpleGraph

Examples

```
g = new ('graphNEL', edgemode='directed')
g = initEdgeAttribute (g, 'edgeType', 'char', 'associates with')
```

---

initNodeAttribute *initNodeAttribute*

Description

Create the node attribute slot that the Bioconductor graph class requires, including a default value, and then specifying what the base type (or 'class') is – 'char', 'integer', or 'numeric' – which is needed by RCytoscape. This method converts these standard R data type names, to the forms needed by Cytoscape.

Usage

```
initNodeAttribute(graph, attribute.name, attribute.type, default.value)
```

Arguments

graph	a Bioconductor graph object.
attribute.name	a string, the name of the new node attribute.
attribute.type	a string, either 'char', 'integer', or 'numeric'
default.value	something sensible, of the right type

Value

Returns the modified graph.

Author(s)

Paul Shannon

See Also

initEdgeAttribute makeSimpleGraph

Examples

```
g = new ('graphNEL', edgemode='directed')
g = initNodeAttribute (g, 'lfc', 'numeric', 1.0)
```

---

layout *layout*

Description

Layout the current graph according to the specified algorithm.

Usage

```
layout(obj, layout.name='jgraph-spring')
```

Arguments

obj	a CytoscapeWindowClass object.
layout.name	a string, one of the values returned by getLayoutNames, 'jgraph-spring' by default.

Value

Nothing.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)  # applies default vizmap (rendering) rules, plus any you
             # have specified
```

makeRandomGraph *makeRandomGraph*

Description

Create a random undirected graphNEL, useful for testing. Two default edge attributes are added, for demonstration purposes.

Usage

```
makeRandomGraph(node.count=12, seed=123)
```

Arguments

node.count	the number of nodes you wish to see in the graph
seed	an integer which, when supplied, allows reproducibility

Value

Returns (by default) a 12-node, rather dense undirected graph, with some attributes on the nodes and edges.

Author(s)

Paul Shannon

Examples

```
g = makeRandomGraph (node.count=12, seed=123)

## The function is currently defined as
function (node.count = 12, seed = 123)
{
    set.seed(seed)
    node.names = as.character(1:node.count)
    g = randomGraph(node.names, M <- 1:2, p = 0.6)
    attr(edgeDataDefaults(g, attr = "weight"), "class") = "DOUBLE"
    edgeDataDefaults(g, "pmid") = "9988778899"
    attr(edgeDataDefaults(g, attr = "pmid"), "class") = "STRING"
    return(g)
}
```

makeSimpleGraph *makeSimpleGraph*

Description

A 3-node, 3-edge graph, with some biological trappings, useful for demonstrations.

Usage

```
makeSimpleGraph()
```

Value

Returns a 3-node, 3-edge graph, with some attributes on the nodes and edges.

Author(s)

Paul Shannon

Examples

```
g = makeSimpleGraph ()

## The function is currently defined as
function ()
{
    g = new("graphNEL", edgemode = "directed")
    nodeDataDefaults(g, attr = "type") = "undefined"
    attr(nodeDataDefaults(g, attr = "type"), "class") = "STRING"
    nodeDataDefaults(g, attr = "lfc") = 1
    attr(nodeDataDefaults(g, attr = "lfc"), "class") = "DOUBLE"
    nodeDataDefaults(g, attr = "label") = "default node label"
    attr(nodeDataDefaults(g, attr = "label"), "class") = "STRING"
    nodeDataDefaults(g, attr = "count") = "0"
    attr(nodeDataDefaults(g, attr = "count"), "class") = "INTEGER"
    edgeDataDefaults(g, attr = "edgeType") = "undefined"
    attr(edgeDataDefaults(g, attr = "edgeType"), "class") = "STRING"
    edgeDataDefaults(g, attr = "score") = 0
    attr(edgeDataDefaults(g, attr = "score"), "class") = "DOUBLE"
    edgeDataDefaults(g, attr = "misc") = ""
    attr(edgeDataDefaults(g, attr = "misc"), "class") = "STRING"
    g = graph::addNode("A", g)
    g = graph::addNode("B", g)
    g = graph::addNode("C", g)
    nodeData(g, "A", "type") = "kinase"
    nodeData(g, "B", "type") = "transcription factor"
    nodeData(g, "C", "type") = "glycoprotein"
    nodeData(g, "A", "lfc") = "-3.0"
    nodeData(g, "B", "lfc") = "0.0"
    nodeData(g, "C", "lfc") = "3.0"
    nodeData(g, "A", "count") = "2"
    nodeData(g, "B", "count") = "30"
    nodeData(g, "C", "count") = "100"
    nodeData(g, "A", "label") = "Gene A"
```

```
  nodeData(g, "B", "label") = "Gene B"
  nodeData(g, "C", "label") = "Gene C"
  g = graph::addEdge("A", "B", g)
  g = graph::addEdge("B", "C", g)
  g = graph::addEdge("C", "A", g)
  edgeData(g, "A", "B", "edgeType") = "phosphorylates"
  edgeData(g, "B", "C", "edgeType") = "synthetic lethal"
  edgeData(g, "A", "B", "score") = 35
  edgeData(g, "B", "C", "score") = -12
  return(g)
}
```

`msg` *msg*

Description

Display the supplied string in the Cytoscape Desktop status bar

Usage

```
msg(obj, string)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`string`	a `char`, an arbitrary string, which can be used to inform the user of things they may wish to know

Value

Nothing.

Author(s)

Paul Shannon

See Also

clearMsg

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
msg (cw, 'this message will appear in the Cytoscape Desktop status bar, which is found
```

noa.names *noa.names*

Description

Retrieve the names of the node attributes in the specified graph.

Usage

```
noa.names(graph)
```

Arguments

```
graph
```

Author(s)

Paul Shannon

See Also

noa, eda, eda.names

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
noa.names (cw@graph)
  #  [1] "type"  "lfc"   "label" "count"
```

noa *noa*

Description

Retrieve the value of the specified node attribute for every node in the graph.

Usage

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

Arguments

graph	typically, a bioc graphNEL)
node.attribute.name	a character string

Value

A list, the contents of which are the attribute values, the names of which are the names of the nodes.

Author(s)

Paul Shannon

See Also

noa.names

Examples

```
cw <- CytoscapeWindow ('demo', graph=makeSimpleGraph(), create.window=FALSE)
noa (cw@graph, 'type')
  #           A.A                   B.B                     C.C
  #      "kinase" "transcription factor"         "glycoprotein"
```

`ping` *ping*

Description

Test the connection to Cytoscape.

Usage

```
ping(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

"It works!"

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
ping (cw)
  # "It works!"
```

redraw *redraw*

Description

Asks Cytoscape to redraw all nodes and edges, applying the vizmap rules.

Usage

```
redraw(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

displayGraph layout

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
redraw (cw)
```

selectNodes *selectNodes*

Description

Select the specified nodes.

Usage

```
selectNodes(obj, node.names)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.

Value

None.

Author(s)

Paul Shannon

See Also

clearSelection getSelectedNodeCount getSelectedNodes hideSelectedNodes

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
clearSelection (cw)
selectNodes (cw, c ('A', 'B'))
getSelectedNodes (cw)
  # [1] "A" "B"
```

---

sendEdgeAttributesDirect *sendEdgeAttributesDirect*

Description

Transfer the named edge attribute to Cytoscape. This method is required, for instance, if you wish to run a 'movie.' For example, if you have a timecourse experiment, with different values at successive time points of the 'phosphorylates' or 'binds' relationship between two nodes. With an edgeColor rule already specified, you can animate the display of the edges in the graph by pumping new values of the edge attributes, and then asking for a redraw. An example of such edge-attribute-driven animation can be found here....[todo].

Usage

```
sendEdgeAttributesDirect(obj, attribute.name, attribute.type, edge.names, values
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`attribute.name`	a `string` one of the attributes defined on the edges.
`attribute.type`	a `string` from one of these four groups: (string), (floating, numeric, double), (integer, int), (string, char, character). This parameter is required because RCytoscape cannot always infer the type of an attribute.
`edge.names`	a `list` of strings, edge names
`values`	a list of objects of the type specified by 'attribute.name', one per edge

Value

None.

Author(s)

Paul Shannon

See Also

sendEdgeAttributes sendNodeAttributes sendNodeAttributesDirect

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
edge.names = as.character (cy2.edge.names (cw@graph))
stopifnot (length (edge.names) == 3)
edge.values = c ('alligator', 'hedgehog', 'anteater')
result = sendEdgeAttributesDirect (cw, 'misc', 'string', edge.names, edge.values)
```

---

sendEdgeAttributes *sendEdgeAttributes*

Description

Transfer the named edge attribute from the the R graph (found in obj@graph) to Cytoscape. This method is typically called by displayGraph, which will suffice for most users' needs. It transfers the specified edge attributes, for all edges, from the cw@graph slot to Cytoscape.

Usage

```
sendEdgeAttributes(obj, attribute.name)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`attribute.name`	a `string` one of the attributes defined on the edges.

Value

None.

Author(s)

Paul Shannon

See Also

sendEdgeAttributesDirect sendNodeAttributes sendNodeAttributesDirect

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
attribute.names = eda.names (cw@graph)

for (attribute.name in attribute.names)
  result = sendEdgeAttributes (cw, attribute.name)
```

sendEdges *sendEdges*

Description

Transfer the edges of the R graph (found in obj@graph) to Cytoscape. This method is not recommended for the average user. It is called behind the scenes by displayGraph.

Usage

```
sendEdges(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

displayGraph sendNodes

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
sendEdges (cw)
```

sendNodeAttributesDirect *sendNodeAttributesDirect*

Description

Transfer the named node attribute, for all named nodes, to Cytoscape. The attribute must be previously defined on the nodes of the graph: see nodeDataDefaults in the graph class. This method is useful if you wish to run a 'movie.' For example, if you have a timecourse experiment, with different values at successive time points of the 'lfc' (log fold change) measurements or 'pValue' of each node. With a nodeColor and nodeSize rule already specified, you can animate the display of the nodes across time in the graph by pumping new values of the attributes attributes using this method, and then asking for a redraw. An example of such node-attribute-driven animation can be found here....[todo].

Usage

```
sendNodeAttributesDirect(obj, attribute.name, attribute.type, node.names, values
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string one of the attributes defined on the nodes.
attribute.type	a string from one of these four groups: (string), (floating, numeric, double), (integer, int), (string, char, character). This parameter is required because RCytoscape cannot always infer the type of an attribute.
node.names	a list of strings, node names
values	a list of objects of the type specified by 'attribute.name', one per node

Value

None.

Author(s)

Paul Shannon

See Also

sendNodeAttributes sendEdgeAttributes sendEdgeAttributesDirect

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
stopifnot ('count' %in% noa.names (cw@graph))
result = sendNodeAttributesDirect (cw, 'count', 'int', c ('A', 'B', 'C'), c (4, 8, 12))
```

---

sendNodeAttributes *sendNodeAttributes*

Description

Transfer the named node attribute from the the R graph (found in obj@graph) to Cytoscape. This method is typically called by displayGraph, which will suffice for most users' needs. It transfers the specified node attributes, for all nodes, from the cw@graph slot to Cytoscape.

Usage

```
sendNodeAttributes(obj, attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	a string one of the attributes defined on the nodes.

Value

None.

Author(s)

Paul Shannon

See Also

sendNodeAttributesDirect sendEdgeAttributes sendEdgeAttributesDirect sendEdges sendNodes displayGraph

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
attribute.names = noa.names (cw@graph)

for (attribute.name in attribute.names)
  result = sendNodeAttributes (cw, attribute.name)
```

---

sendNodes *sendNodes*

Description

Transfer the nodes of the R graph (found in obj@graph) to Cytoscape. This method is not recommended for the average user. It is called behind the scenes by displayGraph.

Usage

```
sendNodes(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

displayGraph sendEdges

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
sendNodes (cw)
```

setDefaultEdgeColor
setDefaultEdgeColor

Description

In the specified CytoscapeWindow, stipulate the color for all edges other than those mentioned in a edge color rule.

Usage

```
setDefaultEdgeColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object, if this vizmap style needs to be distinguished from the default type.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultEdgeColor (cw, '#FFFFFF') # white edges
redraw (cw)
```

setDefaultEdgeLineWidth
setDefaultEdgeLineWidth

Description

In the specified CytoscapeWindow, stipulate the line width, in pixels for all edges.

Usage

```
setDefaultEdgeLineWidth(obj, new.width, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.width	an integer object, typically from 0 to 5.
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setEdgeColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultEdgeLineWidth (cw, 5)
redraw (cw)
```

setDefaultNodeBorderColor
setDefaultNodeBorderColor

Description

In the specified CytoscapeWindow, stipulate the color for all nodeBorders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeBorderColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeBorderColor (cw, '#FFFFFF') # white borders
redraw (cw)
```

setDefaultNodeBorderWidth
setDefaultNodeBorderWidth

Description

In the specified CytoscapeWindow, stipulate the color for all nodeBorders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeBorderWidth(obj, new.width, vizmap.style.name = "default")
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`new.width`	a `String` object, a hex string, of the form '#RRGGBB'.
`vizmap.style.name`	a `String` object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeBorderWidth (cw, 5)
redraw (cw)
```

setDefaultNodeColor *setDefaultNodeColor*

Description

In the specified CytoscapeWindow, stipulate the color for all nodes other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeColor (cw, '#8888FF') # light blue
redraw (cw)
```

setDefaultNodeFontSize
setDefaultNodeFontSize

Description

In the specified CytoscapeWindow, stipulate the color for all nodeBorders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeFontSize(obj, new.size, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.size	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeFontSize (cw, 32)
redraw (cw)
```

setDefaultNodeLabelColor
setDefaultNodeLabelColor

Description

In the specified CytoscapeWindow, stipulate the color for all node labels. There is, at present, no mapping rule for this trait.

Usage

```
setDefaultNodeLabelColor(obj, new.color, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.color	a String object, a hex string, of the form '#RRGGBB'.
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeLabelColor (cw, '#FFFFFF') # white node labels
redraw (cw)
```

setDefaultNodeShape
setDefaultNodeShape

Description

In the specified CytoscapeWindow, stipulate the color for all nodeBorders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeShape(obj, new.shape, vizmap.style.name = "default")
```

Arguments

obj	a CytoscapeWindowClass object.
new.shape	a String object, one of the permissible values (see getNodeShapes).
vizmap.style.name	a String object.

Value

None.

Author(s)

Paul Shannon

See Also

getNodeShapes setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
legal.shapes <- getNodeShapes (cw)
# stopifnot ('diamond' %in% legal.shapes)
setDefaultNodeShape (cw, 'diamond')
redraw (cw)
```

setDefaultNodeSize *setDefaultNodeSize*

Description

In the specified CytoscapeWindow, stipulate the color for all nodeBorders other than those mentioned in a node border color rule.

Usage

```
setDefaultNodeSize(obj, new.size, vizmap.style.name = "default")
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`new.size`	a `integer` object, typically 20 to 100.
`vizmap.style.name`	a `String` object.

Value

None.

Author(s)

Paul Shannon

See Also

setDefaultNodeShape setDefaultNodeColor setDefaultNodeSize setDefaultNodeColor setDefaultNodeBorderColor setDefaultNodeBorderWidth setDefaultNodeFontSize setDefaultNodeLabelColor setDefaultEdgeLineWidth setEdgeColorRule setNodeBorderColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
setDefaultNodeSize (cw, 60) # an intermediate value
redraw (cw)
```

setEdgeColorRule *setEdgeColorRule*

Description

Specify how data attributes – for the specified named attribute – is mapped to edge color.

Usage

```
setEdgeColorRule(obj, attribute.name, attribute.values, colors, default.color='#
```

Arguments

obj	a CytoscapeWindowClass object.
attribute.name	the edge attribute whose values will, when this rule is applied, determine the color of each edge.
attribute.values	a list of (currently, only 3) values. Wise choices are: the minimum, the maximum, some sensible midpoint.
colors	a list of (currently) only 3 colors, expressed as hexadecimal RGB, like this: '#FF0000' or '#FA8800'
default.color	a String object, expressed in hexadecimal RGB, like this: '#FF0000' or '#FA8800'

Value

None.

Author(s)

Paul Shannon

See Also

setEdgeLineStyleRule setNodeColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
edgeType.values = c ('phosphorylates', 'synthetic lethal', 'undefined')
colors = c ('#FF0000', '#FFFF00', '#00FF00')
setEdgeColorRule (cw, 'edgeType',  edgeType.values, colors)
redraw (cw)
```

setEdgeLineStyleRule *specify the line styles to be used in drawing edges*

Description

Specify how data attributes – for the specified named attribute – are mapped to edge line style.

Usage

```
setEdgeLineStyleRule(obj, edge.attribute.name, attribute.values, line.styles, de
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will, when this rule is applied, determine the lineStyle of each edge.
attribute.values	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
line.styles	One line style for each of the attribute.values
default.style	The style to use when an explicit mapping is not provided.

Value

None.

Author(s)

Paul Shannon

See Also

getLineStyles

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
line.styles     <- c ('SINEWAVE',       'DOT',              'PARALLEL_LINES')
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeLineStyleRule (cw, 'edgeType', edgeType.values, line.styles)
redraw (cw)
```

setEdgeSourceArrowColorRule
Specify Rule for the Source Arrow Color

Description

Specify how edge attributes – that is, data values of the specified edge attribute – control the color of the source arrow, found at the end of an edge, where it connects to the source node.

Usage

```
setEdgeSourceArrowColorRule(obj, edge.attribute.name, attribute.values, colors,
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will, when this ColorRule is applied, determine the color of the source arrow for each edge.
attribute.values	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
colors	A color for each of the attribute.values
default.color	The color to use when an explicit mapping is not provided. (Note: this is broken in Cytoscape 2.7)

Value

None.

Author(s)

Paul Shannon

See Also

setEdgeSourceArrowColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
colors <- c ("#AA00AA", "#AAAA00", "#AA0000")
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeSourceArrowColorRule (cw, 'edgeType', edgeType.values, colors)
```

setEdgeSourceArrowRule *specify the arrow types to be used at the end of an edge, at the 'source' node*

Description

Specify how data attributes – for the specified named attribute – are mapped to the source arrow type.

Usage

```
setEdgeSourceArrowRule(obj, edge.attribute.name, attribute.values, arrows, defau
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will, when this rule is applied, determine the sourceArrow of each edge.
attribute.values	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
arrows	One arrow type for each of the attribute.values
default	The arrow type to use when an explicit mapping is not provided.

Value

None.

Author(s)

Paul Shannon

See Also

getArrowShapes

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
arrows <- c ('Arrow', 'Diamond', 'Circle')
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeSourceArrowRule (cw, 'edgeType', edgeType.values, arrows)
redraw (cw)
```

`setEdgeTargetArrowColorRule` *Specify Rule for the Target Arrow Color*

Description

Specify how edge attributes – that is, data values of the specified edge attribute – control the color of the target arrow, found at the end of an edge, where it connects to the target node.

Usage

```
setEdgeTargetArrowColorRule(obj, edge.attribute.name, attribute.values, colors,
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will, when this ColorRule is applied, determine the color of the target arrow of each edge.
`attribute.values`	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
`colors`	A color for each of the attribute.values
`default.color`	The color to use when an explicit mapping is not provided. (Note: this is broken in Cytoscape 2.7)

Value

None.

Author(s)

Paul Shannon

See Also

`setEdgeSourceArrowColorRule`

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
colors <- c ("#AA00AA", "#AAAA00", "#AA0000")
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeTargetArrowColorRule (cw, 'edgeType', edgeType.values, colors)
```

`setEdgeTargetArrowRule` *specify the arrow types to be used at the end of an edge, at the 'target' node*

Description

Specify how data attributes – for the specified named attribute – are mapped to the target arrow type.

Usage

```
setEdgeTargetArrowRule(obj, edge.attribute.name, attribute.values, arrows, defau
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`edge.attribute.name`	the edge attribute whose values will, when this rule is applied, determine the targetArrow of each edge.
`attribute.values`	A list of scalar, discrete values. For instance, interaction types: 'phosphorylates', 'ubiquinates', 'represses', 'activates'
`arrows`	One arrow type for each of the attribute.values
`default`	The arrow type to use when an explicit mapping is not provided.

Value

None.

Author(s)

Paul Shannon

See Also

`getArrowShapes`

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
arrows <- c ('Arrow', 'Diamond', 'Circle')
edgeType.values <- c ('phosphorylates', 'synthetic lethal', 'undefined')
setEdgeTargetArrowRule (cw, 'edgeType', edgeType.values, arrows)
redraw (cw)
```

setEdgeTooltipRule *setEdgeTooltipRule*

Description

Specify the edge attribute to be used as the tooltip for each edge. Non-character attributes are converted to strings before they are used as tooltips.

Usage

```
setEdgeTooltipRule(obj, edge.attribute.name)
```

Arguments

obj	a CytoscapeWindowClass object.
edge.attribute.name	the edge attribute whose values will, when this rule is applied, determine the tooltip on each edge.

Value

None.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
setEdgeTooltipRule (cw, 'edgeType')
```

setGraph *setGraph*

Description

Assigns the supplied graph object to the appropriate slot in the specified CytoscapeWindow object.

Usage

```
setGraph(obj, graph)
```

Arguments

obj	a CytoscapeWindowClass object.
graph	a graph object.

Value

The modified CytoscapeWindow object.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test')  # an empty graph is created by default
graph <- makeSimpleGraph ()
setGraph (cw, graph)
print (length (nodes (getGraph (cw))))
```

---

setNodeBorderColorRule
setNodeBorderColorRule

Description

Specify how data attributes – for the specified named attribute – are mapped to node color. There are two modes: 'interpolate' and 'lookup'. In the former, you specify data values ('control points') and colors; when a node's corresponding data attribute value is exactly that of a control point, the specified color is used. If the node's data attribute falls between control points, then the color is interpolated. Note! In the 'interpolate' mode, you almost always want to provide two additional colors: one for node data values falling below the minimum control point, one for node data values falling above the maximum control point. If you provide an equal number of colors and control.points, the default.color is used to paint nodes above and below the specified range. A useful data exploration strategy would be to use `default.color <- '#000000'` causing all extreme nodes to be painted black.

The 'lookup' mode provides no interpoloation, and is useful when you have a node attribute with a finite set of discrete values, each of which you you want to display in a specific color. For example: render all receptors in yellow, all transcription factors in blue, and all kinases in dark red.

Usage

```
setNodeBorderColorRule(obj, node.attribute.name, control.points, colors, mode='i
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will, when this rule is applied, determine the color of each node.
`control.points`	a list of values. In the interpolate mode, a typical choice is the minimum, the maximum, some sensible midpoint.

`colors` a list of colors, either two more than the number of control points (if mode='interpolate'), in which case the first color is used for all attributes values below the minimum, and the last color is used for those above the maximum. Or, if mode='lookup', the same number of colors as control.points are expected. Colors are expressed as quoted hexadecimal RGB strings, e.g., '#FF0000' or '#FA8800'

`mode` 'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 colors: adding a 'below' and an 'above' color. In lookup mode, specify exactly as many control.points as colors. If are data attribute values are found on the nodes which do not appear in your list, they will displayed in the default color.

`default.color` '#000000' (black) by default, to catch your eye. Used primarily in mode=='lookup' and in mode='interpolate' if you fail to specify 'above' and 'below' values.

Value

None.

Author(s)

Paul Shannon

See Also

setNodeShapeRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
control.points <- c (-3.0, 0.0, 3.0)   # typical range of log-fold-change ratio values
  # paint negative values shades of green, positive values shades of
  # red, out-of-range low values are dark green; out-of-range high
  # values are dark red
colors <- c ("#00AA00", "#00FF00", "#FFFFFF", "#FF0000", "#AA0000")
setNodeBorderColorRule (cw, node.attribute.name='lfc', control.points, colors, mode='in
redraw (cw)
data.values <- c ("kinase", "transcription factor", "glycoprotein")
colors <- c ("#0000AA", "#FFFF00", "#0000AA")
setNodeBorderColorRule (cw, node.attribute.name='type', data.values,   colors, mode='lo
```

`setNodeColorRule` *setNodeColorRule*

Description

Specify how data attributes – for the specified named attribute – are mapped to node color. There are two modes: 'interpolate' and 'lookup'. In the former, you specify data values ('control points') and colors; when a node's corresponding data attribute value is exactly that of a control point, the specified color is used. If the node's data attribute falls between control points, then the color

is interpolated. Note! In the 'interpolate' mode, you almost always want to provide two additional colors: one for node data values falling below the minimum control point, one for node data values falling above the maximum control point. If you provide an equal number of colors and control.points, the default.color is used to paint nodes above and below the specified range. A useful data exploration strategy would be to use `default.color <- '#000000'` causing all extreme nodes to be painted black.

The 'lookup' mode provides no interpoloation, and is useful when you have a node attribute with a finite set of discrete values, each of which you you want to display in a specific color. For example: render all receptors in yellow, all transcription factors in blue, and all kinases in dark red.

Usage

```
setNodeColorRule(obj, node.attribute.name, control.points, colors, mode='interpo
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will, when this rule is applied, determine the color of each node.
`control.points`	a list of values. In the interpolate mode, a typical choice is the minimum, the maximum, some sensible midpoint.
`colors`	a list of colors, either two more than the number of control points (if mode='interpolate'), in which case the first color is used for all attributes values below the minimum, and the last color is used for those above the maximum. Or, if mode='lookup', the same number of colors as control.points are expected. Colors are expressed as quoted hexadecimal RGB strings, e.g., '#FF0000' or '#FA8800'
`mode`	'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 colors: adding a 'below' and an 'above' color. In lookup mode, specify exactly as many control.points as colors. If are data attribute values are found on the nodes which do not appear in your list, they will displayed in the default color.
`default.color`	'#000000' (black) by default, to catch your eye. Used primarily in mode=='lookup' and in mode='interpolate' if you fail to specify 'above' and 'below' values.

Value

None.

Author(s)

Paul Shannon

See Also

setNodeShapeRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
control.points <- c (-3.0, 0.0, 3.0)   # typical range of log-fold-change ratio values
  # paint negative values shades of green, positive values shades of
  # red, out-of-range low values are dark green; out-of-range high
  # values are dark red
node.colors <- c ("#00AA00", "#00FF00", "#FFFFFF", "#FF0000", "#AA0000")
setNodeColorRule (cw, node.attribute.name='lfc', control.points, node.colors, mode='int
redraw (cw)
data.values <- c ("kinase", "transcription factor", "glycoprotein")
node.colors <- c ("#0000AA", "#FFFF00", "#0000AA")
setNodeColorRule (cw, node.attribute.name='type', data.values,   node.colors, mode='loc
```

---

setNodeLabelRule *setNodeLabelRule*

Description

Specify the node attribute to be used as the label for each node. Non-character attributes are converted to strings before they are used as labels.

Usage

```
setNodeLabelRule(obj, node.attribute.name)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will, when this rule is applied, determine the label on each node.

Value

None.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
setNodeLabelRule (cw, 'label')
redraw (cw)
setNodeLabelRule (cw, 'type')
redraw (cw)
setNodeLabelRule (cw, 'lfc')
redraw (cw)
setNodeLabelRule (cw, 'count')
redraw (cw)
```

```
setNodeLabelRule (cw, 'label')
redraw (cw)
```

`setNodeShapeRule` *setNodeShapeRule*

Description

Specify how data attributes how the specified node attribute values determine the node shape.

Usage

```
setNodeShapeRule (obj, node.attribute.name=, attribute.values,
node.shapes, default.shape)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will, when this rule is applied, determine the shape of each node.
`attribute.values`	A list of scalar, discrete values. For instance, molecule types: 'transporter', 'receptor', 'kinase'
`node.shapes`	A list of nodes selected from among those supported.
`default.shape`	A single string, the shape used if no explicit mapping is provided.

Value

None.

Author(s)

Paul Shannon

See Also

setNodeColorRule setNodeLabelRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
shapes <- c ("trapezoid", "round_rect", "ellipse")
molecule.types <- c ("kinase", "transcription factor", "glycoprotein")
setNodeShapeRule (cw, node.attribute.name='type', molecule.types,  shapes)
redraw (cw)
```

setNodeSizeRule *setNodeSizeRule*

Description

Specify how data attributes how the specified node attribute values determine the node size.

Usage

```
setNodeSizeRule (obj, node.attribute.name, control.points, node.sizes,mode='inte
```

Arguments

obj	a CytoscapeWindowClass object.
node.attribute.name	the node attribute whose values will, when this rule is applied, determine the size of each node.
control.points	A list of (currently, exactly 3) values, which specify the 'control points' to control the coloring of nodes
node.sizes	The nodes sizes which correspond to the control points.
mode	'interpolate' or 'lookup'. This roughly corresponds to the visual mapping of continuously varying data (i.e., lfc or pValue), versus visual mapping of discrete data (i.e., molecule type, or phosphorylation status). With the interpolation mode, you must specify n+2 colors: adding a 'below' and an 'above' color. In lookup mode, specify exactly as many control.points as colors. If are data attribute values are found on the nodes which do not appear in your list, they will displayed in the default color.
default.size	the size of nodes not otherwise specified. Does not work in Cytoscape 2.7.

Value

None.

Author(s)

Paul Shannon

See Also

setNodeColorRule

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
control.points <- c (10, 30, 80)
node.sizes  <- c (20, 50, 80)
node.attribute.name <- 'count'   # previously defined, has values which  range between
```

```
   # remind yourself of the values of count on each of the three nodes
print (noa (getGraph (cw), 'count'))
   #  A.A   B.B   C.C
   #  "2"  "30" "100"
setNodeSizeRule (cw, node.attribute.name, control.points, node.sizes, mode='interpolate

   # now make a new rule.  explicitly specify below and above sizes
 node.sizes    <- c (1, 20, 50, 80, 200)   # anything below 20 will have size of 1; and
 setNodeSizeRule (cw, node.attribute.name, control.points, node.sizes, mode='interpolat
```

`setNodeTooltipRule` *setNodeTooltipRule*

Description

Specify the node attribute to be used as the tooltip for each node. Non-character attributes are converted to strings before they are used as tooltips.

Usage

```
setNodeTooltipRule(obj, node.attribute.name)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.
`node.attribute.name`	the node attribute whose values will, when this rule is applied, determine the tooltip on each node.

Value

None.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
setNodeTooltipRule (cw, 'type')
setNodeTooltipRule (cw, 'lfc')
setNodeTooltipRule (cw, 'count')
```

setPosition *setPosition*

Description

Set the position of the specified nodes on the CytoscapeWindow canvas. Use this for any hand-crafted layouts, or novel layout algorithms, you wish to use.

Usage

```
setPosition(obj, node.names, x.coords, y.coords)
```

Arguments

obj	a CytoscapeWindowClass object.
node.names	a list of strings, the names of nodes to select.
x.coords	a list of floating point numbers, one for each node in the node.names list.
y.coords	a list of floating point numbers, one for each node in the node.names list.

Value

None.

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw)
setPosition (cw, c ('A', 'B', 'C'), c (10.0, 20.0, 500), c (0.0,
             100.0, 3))
```

sfn *sfn*

Description

In the Cytoscape window, select all first neighbors of the curently selected nodes.

Usage

```
sfn(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

getSelectedNodeCount selectNodes getSelectedNodes

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
redraw (cw)
layout (cw, 'jgraph-spring')
selectNodes (cw, 'A')
sfn (cw)
getSelectedNodes (cw)  # should be c ('A', 'B', 'C')
```

---

unhideAll *unhideAll*

Description

Currently (in Cytoscape 2.7) broken. The redisplay of hidden nodes and edges does not alwasy work...

Usage

```
unhideAll(obj)
```

Arguments

obj	a CytoscapeWindowClass object.

Value

None.

Author(s)

Paul Shannon

See Also

selectNodes clearSelection

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
displayGraph (cw)
layout (cw, 'jgraph-spring')
redraw (cw)
clearSelection (cw)
selectNodes (cw, 'A')
hideSelectedNodes (cw)
```

`version` *version*

Description

Test the connection to Cytoscape.

Usage

```
version(obj)
```

Arguments

`obj`	a `CytoscapeWindowClass` object.

Value

"A string describing the current version of the CytoscapeRPC plugin."

Author(s)

Paul Shannon

Examples

```
cw <- CytoscapeWindow ('test', graph=makeSimpleGraph())
print (version (cw))
  # e.g., "1.1.2 Paul"
```

Index