Package 'shinyTree'

July 23, 2025

Type Package

Title jsTree Bindings for Shiny

Version 0.3.1

Date 2023-7-21

Maintainer Michael Bell <bell_michael_a@lilly.com>

Description Exposes bindings to jsTree -- a JavaScript library that supports interactive trees -- to enable a rich, editable trees in Shiny.

License MIT + file LICENSE

Depends R (>= 2.15.1), methods

Imports shiny (>= 0.9.0), htmlwidgets, jsonlite, stringr, promises

Suggests testthat, shinytest, data.tree

BugReports https://github.com/shinyTree/shinyTree/issues

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation no

Author Trestle Technology, LLC [aut],
Jeff Allen [aut],
Institut de Radioprotection et de Sûreté Nucléaire [cph],
Ivan Bozhanov [ctb, cph] (jsTree),
The Dojo Foundation [ctb, cph] (require.js),
jQuery Foundation, Inc. [ctb, cph],
Mike Schaffer [ctb],
Timm Danker [ctb],
Michael Bell [cre],
Sebastian Gatscha [ctb],
Thorn Thaler [ctb]

Repository CRAN

Date/Publication 2023-08-07 18:30:02 UTC

Contents

`depth` *Check depth of a list*

Description

Check depth of a list

Usage

```
depth(x)
```

Arguments

`x`	list

Value

integer

Author(s)

Jasper Schelfhout <jasper.schelfhout@openanalytics.eu>

dfrapply *Recursively apply function to all data.frames in a nested list*

Description

Recursively apply function to all data.frames in a nested list

Usage

```
dfrapply(list, f, ...)
```

Arguments

list	(nested) list containing data.frames
f	function to apply to each data.frame
...	extra arguments to f

Value

list

Author(s)

Jasper Schelfhout <jasper.schelfhout@openanalytics.eu>

dfToTree *Converts a data.frame to a data.tree format*

Description

Converts a data.frame to a data.tree format

Usage

```
dfToTree(df, hierarchy = colnames(df))
```

Arguments

df	data.frame
hierarchy	ordered character vector of column names defining the hierarchy

Value

nested list

Author(s)

Jasper Schelfhout <jasper.schelfhout@openanalytics.eu>

Examples

```
## Not run:
df <- data.frame(Titanic)
tree <- dfToTree(df, c("Sex", "Class", "Survived"))

## End(Not run)
```

---

get_checked *Get the checked nodes from a tree*

Description

Extract the nodes from the tree that are checked in a more convenient format. You can choose which format you prefer.

Usage

```
get_checked(tree, format = c("names", "slices", "classid"))
```

Arguments

tree	The input$tree shinyTree you want to inspect.
format	In which format you want the output. Use names to get a simple list of the names (with attributes describing the node's ancestry), or slices to get a list of lists, each of which is a slice of the list used to get down to the selected node.

---

get_selected *Get the selected nodes from a tree*

Description

Extract the nodes from the tree that are selected in a more convenient format. You can choose which format you prefer.

Usage

```
get_selected(tree, format = c("names", "slices", "classid"))
```

Arguments

tree	The input$tree shinyTree you want to inspect.
format	In which format you want the output. Use names to get a simple list of the names (with attributes describing the node's ancestry), or slices to get a list of lists, each of which is a slice of the list used to get down to the selected node.

renderEmptyTree *Render an empty ShinyTree*

Description

Renders a tree with no defined nodes.

Usage

```
renderEmptyTree()
```

See Also

shinyTree

renderTree *Render a ShinyTree*

Description

Should return a list from the given expression which will be converted into a shinyTree.

Usage

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

Arguments

expr	The expression to be evaluated which should produce a list.
env	The environment in which expr should be evaluated.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

See Also

shinyTree

renderTreeAsync *Render an asynchronous ShinyTree*

Description

Should return a list from the given expression which will be converted into a shinyTree.

Usage

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

Arguments

expr	The expression to be evaluated which should produce a list.
env	The environment in which expr should be evaluated.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

See Also

shinyTree

set_node_attrs *Tree traversal*

Description

Traverse through tree/list to set node attributes, e.g. change icons. Useful for directory structure icons where inner nodes are directories, leafs are files.

Usage

```
set_node_attrs(tree, attr_name, inner_val, leaf_val)
```

Arguments

tree	named nested list
attr_name	name of attribute to set
inner_val	value of attribute for inner tree nodes
leaf_val	value of attribute for outer tree nodes

Value

named nested list

Examples

```
tree <- dfToTree(data.frame(Titanic),  c("Sex", "Survived"))
str(set_node_attrs(tree, attr_name = "sttype", inner_val = "directory", leaf_val = "file"))
```

---

shinyTree *Create a Shiny Tree*

Description

This creates a spot in your Shiny UI for a shinyTree which can then be filled in using renderTree.

Usage

```
shinyTree(
  outputId,
  checkbox = FALSE,
  search = FALSE,
  searchtime = 250,
  dragAndDrop = FALSE,
  types = NULL,
  theme = "default",
  themeIcons = TRUE,
  themeDots = TRUE,
  sort = FALSE,
  unique = FALSE,
  wholerow = FALSE,
  stripes = FALSE,
  multiple = TRUE,
  animation = 200,
  contextmenu = FALSE,
  three_state = TRUE,
  whole_node = TRUE,
  tie_selection = TRUE
)
```

Arguments

outputId	The ID associated with this element
checkbox	If TRUE, will enable checkboxes next to each node to make the selection of multiple nodes in the tree easier.
search	If TRUE, will enable search functionality in the tree by adding a search box above the produced tree. Alternatively, you can set the parameter to the ID of the text input you wish to use as the search field.
searchtime	Determines the reaction time of the search algorithm. Default is 250ms.
dragAndDrop	If TRUE, will allow the user to rearrange the nodes in the tree.

types	enables jstree types functionality when sent proper json (please see the types example)
theme	jsTree theme, one of default, default-dark, or proton.
themeIcons	If TRUE, will show theme icons for each item.
themeDots	If TRUE, will include level dots.
sort	If TRUE, will sort the nodes in alphabetical/numerical order.
unique	If TRUE, will ensure that no node name exists more than once.
wholerow	If TRUE, will highlight the whole selected row.
stripes	If TRUE, the tree background is striped.
multiple	If TRUE, multiple nodes can be selected.
animation	The open / close animation duration in milliseconds. Set this to FALSE to disable the animation (default is 200).
contextmenu	If TRUE, will enable a contextmenu to create/rename/delete/cut/copy/paste nodes.
three_state	If TRUE, a boolean indicating if checkboxes should cascade down and have an undetermined state
whole_node	If TRUE,a boolean indicating if clicking anywhere on the node should act as clicking on the checkbox
tie_selection	If TRUE, controls if checkbox are bound to the general tree selection or to an internal array maintained by the checkbox plugin.

Details

A shinyTree is an output *and* an input element in the same time. While you can fill it via renderTree you can access its content via input$tree (for example after the user rearranged some nodes). By default, input$tree will return a list similiar to the one you use to fill the tree. This behaviour is controlled by getOption("shinyTree.defaultParser"). It defaults to "list", but can be set to "tree", in which case a data.tree is returned.

See Also

renderTree

---

treeToDf *Convert tree into data.frame*

Description

Convert tree into data.frame

Usage

```
treeToDf(tree, hierarchy = NULL)
```

Arguments

tree	named nested list
hierarchy	sorted character vector with name for each level of the list

Value

data.frame

Author(s)

Michael Bell

Examples

```
## Not run:
df <- data.frame(Titanic)
tree <- dfToTree(df, c("Sex", "Class", "Survived"))
newDf <- treeToDf(tree, c("Sex", "Class", "Survived"))

## End(Not run)
```

---

treeToJSON *Converts a data.tree to a JSON format*

Description

Walk through a data.tree and constructs a JSON string, which can be rendered by shinyTree.

Usage

```
treeToJSON(
  tree,
  keepRoot = FALSE,
  topLevelSlots = c("default", "all"),
  createNewId = TRUE,
  pretty = FALSE
)
```

Arguments

tree	the data.tree which should be parses
keepRoot	logical. If FALSE (default) the root node from the tree is pruned
topLevelSlots	determines which slots should be moved to the top level of the node. If default or NULL slots used in the jsTree JSON are kept on the top level, while any other atomic / list slots from the tree are stored in an own slot called 'data'. If all *all* nodes are stored on the top level. Alternatively, it can be an explicit vector of slot names which should be kept. In the latter case it is the user's responsibility to ensure that jsTree slots stay on the top level.

createNewId	logical. If TRUE a new id will be generated. Any old 'id' will be stored in 'id.orig' and a warning will be issued, If FALSE, any existing id will be re-used.
pretty	logical. If TRUE the resulting JSON is prettified

Details

The JSON string generated follows the jsTree specifications. In particular it encodes children nodes via the 'children' slot.

All atomic or list slots of a node in the tree are stored in a data slot in the resulting JSON.

If the user wants to store some slots not in the data slot but on the top level of the node, parameter topLevelSlots can be used. This is useful for additional parameters such as 'icon', 'li_attr' or 'a_attr', which jsTree expect to be on the top level of the node.

An example of how to make use of this functionality can be found in the example folder of this library.

Value

a JSON string representing the data.tree

Note

updateTree and renderTree need an unevaluated JSON string. Hence, this function returns a string rather than the JSON object itself.

Author(s)

Thorn Thaler, <thorn.thaler@thothal.at>

---

updateTree *Update the tree with new data*

Description

Extract the nodes from the tree that are selected in a more convenient format. You can choose which format you prefer.

Usage

```
updateTree(session, treeId, data = NULL)
```

Arguments

session	The current session variable.
treeId	The identifier for the shinyTree object
data	JSON data or nested list representing the new tree structure.

Index