Package 'regexSelect'

May 9, 2026

Version 1.0.0

Date 2017-09-22

Title Regular Expressions in 'shiny' Select Lists

Description 'shiny' extension that adds regular expression filtering capabilities to the choice vector of the select list.

Depends R (>= 2.3.0)

Imports shiny, shinyjs

License GPL-2 | GPL-3

URL https://github.com/yonicd/regexSelect

BugReports https://github.com/yonicd/regexSelect/issues

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation no

Author Jonathan Sidi [aut, cre]

Maintainer Jonathan Sidi <yonicd@gmail.com>

Repository CRAN

Date/Publication 2017-09-22 15:45:17 UTC

Contents

regexSelect *Create a selectize list input control with regular expression capabilities*

Description

Create a selectize list that can be used to choose a single or multiple items from a list of values with extension for regular expression.

Usage

```
regexSelect(input, output, session, data)
```

Arguments

input	The input slot that will be used to access the value.
output	The output variable to read the list of values returned be regex query
session	The session of the shiny application
data	reactive element contains a character vector where matches are sought, or an object which can be coerced by as.character to a character vector

Value

reactive character vector

Examples

```
if(interactive()){
ui <- shiny::fluidPage(
regexSelectUI(id = "a", label = "Variable:",choices = names(iris)),
shiny::tableOutput("data")
)


ui.show <- shiny::fluidPage(
regexSelectUI(id = "a", label = "Variable:",choices = names(iris),checkbox.show = TRUE),
shiny::tableOutput("data")
)

server <- function(input, output, session) {
 curr_cols<-shiny::callModule(regexSelect, "a",shiny::reactive(names(iris)))

 shiny::observeEvent(curr_cols(),{
 cols_now<-curr_cols()
 if(length(cols_now)==0)  cols_now<-names(data())
 output$data <- shiny::renderTable({iris[,cols_now , drop = FALSE]}, rownames = TRUE)
 })
}
```

```
#do not show regex checkboxes
shiny::shinyApp(ui, server)

#show regex checkboxes
shiny::shinyApp(ui.show, server)
 }
```

`regexSelectUI` *Create UI object for a selectize list input control with regular expression capabilities*

Description

Create UI object for a selectize list that can be used to choose a single or multiple items from a list of values with extension for regular expression.

Usage

```
regexSelectUI(id, label, choices, checkbox.selected = c("enable",
  "ignore.case"), checkbox.inline = TRUE, checkbox.show = FALSE)
```

Arguments

`id`	id of shiny module used in regexSelect
`label`	character, label of the selectize object
`choices`	List of values to select from. If elements of the list are named, then that name rather than the value is displayed to the user. This can also be a named list whose elements are (either named or unnamed) lists or vectors. If this is the case, the outermost names will be used as the "optgroup" label for the elements in the respective sublist. This allows you to group and label similar choices.
`checkbox.selected`	character, options of the checkbox to set as TRUE, see details, Default: c("enable", "ignore.case")
`checkbox.inline`	boolean, render the checkbox choices inline (i.e. horizontally), Default: TRUE
`checkbox.show`	boolean, show the checkbox options as part of UI output or hide them, Default: FALSE

Details

checkbox.selected is used as a proxy for ellipses to pass arguments to a grep(selectize value, selectize choices,value=TRUE,...). This makes the options in checkbox.selected the same as the arguments that pass to grep: ignore.case, perl,fixed and invert.

In addition there are two more arguments that the user can set enable which toggles the grep functionality to return it to regular selectize with options multiple=TRUE and create=TRUE. The other

argument is retain, this lets the user control if the search terms are added to the selectize choices or to keep it as originally entered, there by converting the selectize into a search field. If checkbox.show is false the initial values passed through checkbox.selected will be used.

Value

A list of HTML elements that can be added to a UI definition.

Examples

```
if(interactive()){
ui <- shiny::fluidPage(
regexSelectUI(id = "a", label = "Variable:",choices = names(iris)),
shiny::tableOutput("data")
)


ui.show <- shiny::fluidPage(
regexSelectUI(id = "a", label = "Variable:",choices = names(iris),checkbox.show = TRUE),
shiny::tableOutput("data")
)

server <- function(input, output, session) {
 curr_cols<-shiny::callModule(regexSelect, "a",shiny::reactive(names(iris)))

 shiny::observeEvent(curr_cols(),{
 cols_now<-curr_cols()
 if(length(cols_now)==0)  cols_now<-names(data())
 output$data <- shiny::renderTable({iris[,cols_now , drop = FALSE]}, rownames = TRUE)
 })
}

#do not show regex checkboxes
shiny::shinyApp(ui, server)

#show regex checkboxes
shiny::shinyApp(ui.show, server)
 }
```

Index