

Package ‘shinySelect’

May 9, 2026

Title A Wrapper of the 'react-select' Library

Version 1.3.0

Description Provides a select control widget for 'Shiny'. It is easily customizable, and one can easily use HTML in the items and KaTeX to type mathematics.

License GPL-3

URL <https://github.com/stla/shinySelect>

BugReports <https://github.com/stla/shinySelect/issues>

Imports fontawesome, htmltools, reactR, stats, utils

Suggests bslib, jsTreeR, shiny

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation no

Author Stéphane Laurent [aut, cre],
Jed Watson [cph] (author of the 'react-select' library),
Clauderic Demers [cph] (author of the 'react-sortable-hoc' library)

Maintainer Stéphane Laurent <laurent_step@outlook.fr>

Repository CRAN

Date/Publication 2022-11-18 11:00:02 UTC

Contents

| | |
|------------------------------------|----|
| HTMLchoices | 2 |
| HTMLgroupedChoices | 3 |
| katex | 4 |
| selectControlInput | 4 |
| toggleMenu | 11 |
| updateSelectControlInput | 11 |

| | |
|--------------|-----------|
| Index | 13 |
|--------------|-----------|

HTMLchoices

Choices with HTML

Description

Create an object for choices resorting to HTML.

Usage

```
HTMLchoices(labels, values)
```

Arguments

| | |
|--------|--|
| labels | the labels of the select control, can be HTML elements created with the HTML function or shiny.tag objects such as <code>tags\$span(style = "color:lime;", "label")</code> |
| values | the values associated to the labels, they must be character strings, given in a vector or in a list |

Value

An object (the values object with some attributes) to be passed on to the choices argument of the [selectControlInput](#) function.

See Also

[HTMLgroupedChoices](#) for choices with groups.

Examples

```
library(shinySelect)
library(fontawesome)
library(shiny)
food <- HTMLchoices(
  labels = list(
    tags$span(fa_i("hamburger"), "Hamburger"),
    tags$span(fa_i("pizza-slice"), "Pizza"),
    tags$span(fa_i("fish"), "Fish")
  ),
  values = list("hamburger", "pizza", "fish")
)
```

HTMLgroupedChoices *Choices with groups and HTML*

Description

Create an object for grouped choices resorting to HTML.

Usage

```
HTMLgroupedChoices(groups, labels, values)
```

Arguments

| | |
|--------|--|
| groups | list of HTML elements which can be created with the HTML function or shiny.tag objects, the headings |
| labels | list of lists, one list for each group, made of HTML elements |
| values | list of lists of character strings, each label must have a value |

Value

An object to be passed on to the choices argument of the [selectControlInput](#) function.

Examples

```
library(shinySelect)
library(shiny)
states <- HTMLgroupedChoices(
  groups = lapply(list("East Coast", "West Coast", "Midwest"), function(x){
    tags$h2(x, style="text-decoration: underline")
  }),
  labels = list(
    lapply(list("NY", "NJ", "CT"), function(x){
      tags$span(HTML("&bull;"), x, style="color: red")
    }),
    lapply(list("WA", "OR", "CA"), function(x){
      tags$span(HTML("&bull;"), x, style="color: green")
    }),
    lapply(list("MN", "WI", "IA"), function(x){
      tags$span(HTML("&bull;"), x, style="color: blue")
    })
  ),
  values = list(
    list("NY", "NJ", "CT"),
    list("WA", "OR", "CA"),
    list("MN", "WI", "IA")
  )
)
```

katex

KaTeX code

Description

Create an object to be decoded by KaTeX.

Usage

```
katex(x)
```

Arguments

x string, some KaTeX code (this is similar to LaTeX)

Value

A list containing the url-encoding of x.

Examples

```
library(shinySelect)
choices <- HTMLchoices(
  values = list("alpha", "beta", "gamma"),
  labels = list(katex("\\alpha"), katex("\\beta"), katex("\\gamma"))
)
```

selectControlInput*Select control widget*

Description

Create a select control widget to be included in a Shiny UI.

Usage

```
selectControlInput(
  inputId,
  label,
  choices,
  selected = NULL,
  multiple = FALSE,
  sortable = FALSE,
  optionsStyles = list(),
  controlStyles = list(),
  multiValueStyles = list(),
```

```

    multiValueLabelStyles = list(),
    multiValueRemoveStyles = list(),
    containerClass = "mt-4 col-md-6 col-offset-4",
    animated = FALSE,
    displayGroupSizes = TRUE,
    closeMenuOnSelect = !multiple,
    ignoreCaseOnFilter = TRUE,
    ignoreAccentsOnFilter = TRUE
  )

```

Arguments

| | |
|------------------------|--|
| inputId | the input slot that will be used to access the value |
| label | a label for the widget, can be a HTML element; NULL for no label |
| choices | a list of single choices or grouped choices; to use HTML, see the functions HTMLchoices and HTMLgroupedChoices |
| selected | the initially selected value; can be NULL and can be a vector or a list of values if <code>multiple = TRUE</code> |
| multiple | Boolean, whether the selection of multiple items is allowed |
| sortable | Boolean, whether the multiple selections are sortable |
| optionsStyles | styles for the options, given as a list |
| controlStyles | styles for the control bar, given as a list |
| multiValueStyles | styles for the item boxes when <code>multiple = TRUE</code> , such as the background color |
| multiValueLabelStyles | styles for the item labels when <code>multiple = TRUE</code> , such as the font style |
| multiValueRemoveStyles | styles for the box containing the cross used to remove an item |
| containerClass | CSS class(es) for the container; the default value assumes you use the 'bslib' package with <code>bs_theme(version = 4)</code> |
| animated | Boolean; this has an effect only when <code>multiple = TRUE</code> : the removal of the items is animated |
| displayGroupSizes | only for grouped choices, whether to display the number of elements of each group |
| closeMenuOnSelect | Boolean, whether to close the menu when doing a selection |
| ignoreCaseOnFilter | Boolean, whether to ignore the case when searching an option |
| ignoreAccentsOnFilter | Boolean, whether to ignore the accents when searching an option |

Value

An input element that can be included in a Shiny UI definition.

Examples

```

# an example using KaTeX ####
library(shiny)
library(shinySelect)
library(bslib)

choices <- HTMLchoices(
  values = list("alpha", "beta", "gamma"),
  labels = list(katex("\\alpha"), katex("\\beta"), katex("\\gamma"))
)

ui <- fluidPage(
  theme = bs_theme(version = 4),
  titlePanel("KaTeX example"),
  selectControlInput(
    "select",
    label = tags$h1("Make a choice", style="color: red;"),
    choices = choices,
    selected = "alpha",
    multiple = FALSE
  ),
  br(),
  verbatimTextOutput("textOutput")
)

server <- function(input, output, session) {
  output[["textOutput"]] <- renderPrint({
    sprintf("You selected: %s.", input[["select"]])
  })
}

if(interactive()){
  shinyApp(ui, server)
}

# An example of `sortable = TRUE`, with fontawesome icons ####
library(shiny)
library(shinySelect)
library(bslib)
library(fontawesome)

food <- HTMLchoices(
  labels = list(
    tags$span(fa_i("hamburger"), "Hamburger"),
    tags$span(fa_i("pizza-slice"), "Pizza"),
    tags$span(fa_i("fish"), "Fish")
  ),
  values = list("hamburger", "pizza", "fish")
)

styles <- list(
  borderBottom = "2px solid orange",

```

```

    backgroundColor = list(
      selected = "cyan",
      focused = "lemonchiffon",
      otherwise = "seashell"
    )
  )
)

ui <- fluidPage(
  theme = bs_theme(version = 4),
  titlePanel("Sortable example"),
  selectControlInput(
    "select",
    label = tags$h1("Make a choice", style="color: red;"),
    optionsStyles = styles,
    choices = food,
    selected = "hamburger",
    multiple = TRUE,
    sortable = TRUE,
    animated = TRUE
  ),
  br(),
  verbatimTextOutput("textOutput")
)

server <- function(input, output, session) {
  output[["textOutput"]] <- renderPrint({
    sprintf("You selected: %s.", toString(input[["select"]]))
  })
}

if(interactive()){
  shinyApp(ui, server)
}

# An example with tooltips ####
library(shiny)
library(bslib)
library(shinySelect)

data(Countries, package = "jsTreeR")

continents <- unique(Countries[["continentName"]])

L <- lapply(continents, function(continent){
  indices <- Countries[["continentName"]] == continent
  countries <- Countries[["countryName"]][indices]
  pop <- Countries[["population"]][indices]
  mapply(function(x, y){tags$span(x, `data-toggle`="tooltip", title=y)},
    countries, pop, SIMPLIFY = FALSE, USE.NAMES = FALSE)
})

countries <- lapply(continents, function(continent){
  indices <- Countries[["continentName"]] == continent

```

```

  Countries[["countryName"]][indices]
})

countries <- HTMLgroupedChoices(
  groups = lapply(continents, function(nm) tags$h2(nm, style="color: blue;")),
  labels = L,
  values = countries
)

CSS <- '
.tooltip {
  pointer-events: none;
}
.tooltip > .tooltip-inner {
  pointer-events: none;
  background-color: #73AD21;
  color: #FFFFFF;
  border: 1px solid green;
  padding: 5px;
  font-size: 15px;
  text-align: justify;
  margin-left: 10px;
  max-width: 1000px;
}
.tooltip > .arrow::before {
  border-top-color: #73AD21;
}
'

ui <- fluidPage(
  theme = bs_theme(version = 4),
  tags$head(
    tags$style(HTML(CSS))
  ),
  titlePanel("Tooltips example"),
  sidebarLayout(
    sidebarPanel(
      selectControlInput(
        "select",
        label = tags$h3("Choose some countries", style="color: red;"),
        containerClass = NULL,
        choices = countries,
        selected = c("Tonga", "Austria"),
        multiple = TRUE,
        animated = TRUE
      )
    ),
    mainPanel(
      verbatimTextOutput("textOutput")
    )
  )
)

```

```

server <- function(input, output, session) {
  output[["textOutput"]] <- renderPrint({
    sprintf("You selected: %s.", toString(input[["select"]]))
  })
}

if(interactive()){
  shinyApp(ui, server)
}

# An example of custom styles ####
library(shiny)
library(shinySelect)

states <- HTMLgroupedChoices(
  groups = lapply(list("East Coast", "West Coast", "Midwest"), function(x){
    tags$h2(x, style="text-decoration: underline")
  }),
  labels = list(
    lapply(list("NY", "NJ", "CT"), function(x){
      tags$span(HTML("&bull;"), x, style="color: red")
    }),
    lapply(list("WA", "OR", "CA"), function(x){
      tags$span(HTML("&bull;"), x, style="color: green")
    }),
    lapply(list("MN", "WI", "IA"), function(x){
      tags$span(HTML("&bull;"), x, style="color: blue")
    })
  ),
  values = list(
    list("NY", "NJ", "CT"),
    list("WA", "OR", "CA"),
    list("MN", "WI", "IA")
  )
)

styles <- list(
  borderBottom = "2px dotted orange",
  backgroundColor = list(
    selected = "cyan",
    focused = "lemonchiffon",
    otherwise = "seashell"
  )
)

controlStyles = list(
  marginTop = "0",
  marginRight = "50px",
  boxShadow = toString(c(
    "rgba(50, 50, 93, 0.25) 0px 50px 100px -20px",
    "rgba(0, 0, 0, 0.3) 0px 30px 60px -30px",
    "rgba(10, 37, 64, 0.35) 0px -2px 6px 0px inset;"
  ))
)

```

```

)
multiValueStyles = list(
  backgroundColor = "lavenderblush"
)
multiValueLabelStyles = list(
  fontStyle = "italic",
  fontWeight = "bold"
)
multiValueRemoveStyles = list(
  color = "hotpink",
  ":hover" = list(
    backgroundColor = "navy",
    color = "white"
  )
)
)

CSS <- '
div[class$="-group"][id^="react-select"][id$="-heading"] {
  background: #0F2027; /* fallback for old browsers */
  background: -webkit-linear-gradient(to right, #2C5364, #203A43, #0F2027);
  background: linear-gradient(to right, #2C5364, #203A43, #0F2027);
}'

ui <- fluidPage(
  tags$head(
    tags$style(HTML(CSS))
  ),
  titlePanel("Custom styles example"),
  splitLayout(
    selectControlInput(
      "select",
      label = tags$h1("Choose some states", style="color: red;"),
      containerClass = NULL,
      optionsStyles = styles,
      controlStyles = controlStyles,
      multiValueStyles = multiValueStyles,
      multiValueLabelStyles = multiValueLabelStyles,
      multiValueRemoveStyles = multiValueRemoveStyles,
      choices = states,
      selected = list("NY", "CT"),
      multiple = TRUE,
      sortable = TRUE,
      animated = TRUE
    ),
    tagList(
      verbatimTextOutput("textOutput"),
      br(),
      actionButton("toggle", "Toggle menu", class = "btn-primary")
    )
  )
)

server <- function(input, output, session) {

```

```
output[["textOutput"]] <- renderPrint({
  sprintf("You selected: %s.", toString(input[["select"]]))
})
observeEvent(input[["toggle"]], {
  toggleMenu(session, "select")
})
}

if(interactive()){
  shinyApp(ui, server)
}
```

`toggleMenu`*Toggle a select control widget*

Description

Toggle (open/close) a select control widget.

Usage

```
toggleMenu(session, inputId)
```

Arguments

| | |
|----------------------|------------------------------------|
| <code>session</code> | the Shiny session object |
| <code>inputId</code> | the input id of the select control |

Value

No value; called for side effect.

Examples

```
# See the last example of 'selectControlInput'.
```

`updateSelectControlInput`*Update a select control widget*

Description

Change the value of a select control input.

Usage

```
updateSelectControlInput(session, inputId, choices = NULL, selected = NULL)
```

Arguments

| | |
|----------|---|
| session | the Shiny session object |
| inputId | the id of the select control widget to be updated |
| choices | new choices, or NULL |
| selected | new value(s) for the selected items, or NULL |

Value

No returned value, called for side effect.

Index

HTML, [2](#), [3](#)

HTMLchoices, [2](#), [5](#)

HTMLgroupedChoices, [2](#), [3](#), [5](#)

katex, [4](#)

selectControlInput, [2](#), [3](#), [4](#)

toggleMenu, [11](#)

updateSelectControlInput, [11](#)