Package `shinyjqui`

February 3, 2020

Type Package
Title 'jQuery UI' Interactions and Effects for Shiny
Version 0.3.3
Maintainer Yang Tang <tang_yang@outlook.com>
Description An extension to shiny that brings interactions and animation effects from 'jQuery UI library.
License MIT + file LICENSE
Encoding UTF-8
LazyData true
Depends R (>= 3.2.0)
Imports shiny (>= 0.14.0), htmltools, htmlwidgets, jsonlite
Suggests ggplot2, highcharter, knitr, rmarkdown
URL https://github.com/yang-tang/shinyjqui
BugReports https://github.com/yang-tang/shinyjqui/issues
RoxygenNote 7.0.2
VignetteBuilder knitr
NeedsCompilation no
Author Yang Tang [aut, cre]
Repository CRAN
Date/Publication 2020-02-03 15:50:05 UTC

R topics documented:

Animation_effects ................................. 2
Class_effects .................................. 3
draggableModalDialog ......................... 4
get_jqui_effects ............................... 5
includeJqueryUI ............................... 6
Interactions ................................... 6
Animation effects

Description
Allow element(s) to show animation effects.

- jqui_effect(): Apply an animation effect to matched element(s).
- jqui_hide(): Hide the matched element(s) with animation effect.
- jqui_show(): Display the matched element(s) with animation effect.
- jqui_toggle(): Display or hide the matched element(s) with animation effect.

Usage
jqui_effect(selector, effect, options = NULL, duration = 400, complete = NULL)
jqui_show(selector, effect, options = NULL, duration = 400, complete = NULL)
jqui_hide(selector, effect, options = NULL, duration = 400, complete = NULL)
jqui_toggle(selector, effect, options = NULL, duration = 400, complete = NULL)

Arguments
selector       Deprecated, just keep for backward compatibility. Please use ui and operation parameters instead.
effect         A string indicating which animation effect to use for the transition.
options        A list of effect-specific properties and easing.
duration       A string or number determining how long the animation will run.
complete       A function to call once the animation is complete, called once per matched element.

Details
These functions are R wrappers of effect(), hide(), show() and toggle() from jQuery UI library. They should be used in server of a shiny document.
Class_effects

Examples

```r
## Not run:

```r
# in shiny ui create a plot
plotOutput('foo')
```

```r
# in shiny server apply a 'bounce' effect to the plot
jqui_effect('#foo', 'bounce')
```

```r
# in shiny server hide the plot with a 'fold' effect
jqui_hide('#foo', 'fold')
```

```r
# in shiny server show the plot with a 'blind' effect
jqui_show('#foo', 'blind')
```

## End(Not run)

Class_effects

Class effects.

Description

Manipulate specified class(es) to matched elements while animating all style changes.

- `jqui_add_class()`: Add class(es).
- `jqui_remove_class()`: Remove class(es).
- `jqui_switch_class()`: Switch class(es).

Usage

```r
class = c("class1", "class2")
jqui_add_class(
  selector, class = class)
```

```r
jqui_remove_class(
  selector, class = class)
```

```r
jqui_switch_class(
  selector, class = class)
```

```r
class = c("class1", "class2")
jqui_remove_class(
  selector, class = class)
```

```r
jqui_switch_class(
  selector, class = class)
```
removeClassName,
addClassName,
duration = 400,
easing = "swing",
complete = NULL
)

Arguments

selector       Deprecated, just keep for backward compatibility. Please use ui and operation parameters instead.
className      One or more class names (space separated) to be added to or removed from the class attribute of each matched element.
duration       A string or number determining how long the animation will run.
easing         A string indicating which easing function to use for the transition.
complete       A js function to call once the animation is complete, called once per matched element.
removeClassName One or more class names (space separated) to be removed from the class attribute of each matched element.
addClassName  One or more class names (space separated) to be added to the class attribute of each matched element.

Details

These functions are the R wrappers of addClass(), removeClass() and switchClass() from jQuery UI library. They should be used in server of a shiny app.

Examples

## Not run:
# in shiny ui create a span
tags$span(id = 'foo', 'class animation demo')

# in shiny server add class 'lead' to the span
jqui_add_class('#foo', className = 'lead')

## End(Not run)

draggableModalDialog  Create a draggable modal dialog UI

Description

This creates the UI for a modal dialog similar to shiny::modalDialog except its content is draggable.
get_jqui_effects

Usage

draggableModalDialog(
...,
    title = NULL,
    footer = shiny::modalButton("Dismiss"),
    size = c("m", "s", "l"),
    easyClose = FALSE,
    fade = TRUE
)

Arguments

... UI elements for the body of the modal dialog box.
title An optional title for the dialog.
footer UI for footer. Use NULL for no footer.
size One of "s" for small, "m" (the default) for medium, or "l" for large.
easyClose If TRUE, the modal dialog can be dismissed by clicking outside the dialog box, or by pressing the Escape key. If FALSE (the default), the modal dialog can’t be dismissed in those ways; instead it must be dismissed by clicking on the dismiss button, or from a call to removeModal() on the server.
fade If FALSE, the modal dialog will have no fade-in animation (it will simply appear rather than fade in to view).

Value

A modified shiny modal dialog UI with its content draggable.

get_jqui_effects

Get available animation effects.

Description

Use this function to get all animation effects in jQuery UI.

Usage

get_jqui_effects()

Value

A character vector of effect names
Interactions

includeJqueryUI  \textit{Inject necessary js and css assets to the head of a shiny document (deprecated).}

Description

This function has to be called within the \texttt{ui} of a shiny document before the usage of other \texttt{shinyjqui} functions.

Usage

\begin{verbatim}
includeJqueryUI()
\end{verbatim}

Value

A shiny head tag with necessary js and css assets.

Examples

\begin{verbatim}
if (interactive()) {
  library(shiny)

  shinyApp(
    ui = fluidPage(
      includeJqueryUI(),
      # other ui codes
    ),
    server = function(input, output) {
      # server codes
    }
  )
}
\end{verbatim}

Interactions  \hspace{1cm} \textit{Mouse interactions}

Description

Attach mouse-based interactions to shiny html tags and input/output widgets, and provide ways to manipulate them. The interactions include:

- \texttt{draggable}: Allow elements to be moved using the mouse.
- \texttt{droppable}: Create targets for draggable elements.
- \texttt{resizable}: Change the size of an element using the mouse.
- \texttt{selectable}: Use the mouse to select elements, individually or in a group.
- \texttt{sortable}: Reorder elements in a list or grid using the mouse.
Usage

\texttt{jqui\_draggable(tag, options = NULL)}

\texttt{jqui\_droppable(tag, options = NULL)}

\texttt{jqui\_resizable(tag, options = NULL)}

\texttt{jqui\_selectable(tag, options = NULL)}

\texttt{jqui\_sortable(tag, options = NULL)}

\texttt{jqui\_draggable(}
  \texttt{ui,}
  \texttt{operation = c("enable", "disable", "destroy", "save", "load"),}
  \texttt{options = NULL,}
  \texttt{selector = NULL,}
  \texttt{switch = NULL}
\texttt{)}

\texttt{jqui\_droppable(}
  \texttt{ui,}
  \texttt{operation = c("enable", "disable", "destroy", "save", "load"),}
  \texttt{options = NULL,}
  \texttt{selector = NULL,}
  \texttt{switch = NULL}
\texttt{)}

\texttt{jqui\_resizable(}
  \texttt{ui,}
  \texttt{operation = c("enable", "disable", "destroy", "save", "load"),}
  \texttt{options = NULL,}
  \texttt{selector = NULL,}
  \texttt{switch = NULL}
\texttt{)}

\texttt{jqui\_selectable(}
  \texttt{ui,}
  \texttt{operation = c("enable", "disable", "destroy", "save", "load"),}
  \texttt{options = NULL,}
  \texttt{selector = NULL,}
  \texttt{switch = NULL}
\texttt{)}

\texttt{jqui\_sortable(}
  \texttt{ui,}
  \texttt{operation = c("enable", "disable", "destroy", "save", "load"),}
  \texttt{options = NULL,}
  \texttt{selector = NULL,}
 switch = NULL
)

Arguments

options            A list of interaction_specific_options. Ignored when operation is set as destroy. This parameter also accept a shiny option that controls the shiny input value returned from the element. See Details.

ui                 The target ui element(s) to be manipulated. Can be
                   • A shiny.tag or shiny.tag.list object
                   • A string of jQuery_selector
                   • A JS() wrapped javascript expression that returns a jQuery object.

operation          A string to determine how to manipulate the mouse interaction. Should be one of enable, disable, destroy, save and load. Ignored when ui is a shiny.tag or shiny.tag.list object. See Details.

selector, tag, switch            Deprecated, just keep for backward compatibility. Please use ui and operation parameters instead.

Details

The first parameter ui determines the target shiny ui element(s) to work with. It accepts objects with different classes. When you provide a shiny.tag (e.g., shiny inputs/outputs or ui created by shiny::tags) or a shiny.tag.list (by tagList()) object, the functions return the same ui object with interaction effects attached. When a jQuery_selector or a javascript expression is provided, the functions first use it to locate the target ui element(s) in shiny app, and then attach or manipulate the interactions. Therefore, you can use the first way in ui of a shiny app to created elements with interaction effects, or use the second way in server to manipulate the interactions.

The operation parameter is valid only in the second way. It determines how to manipulate the interaction, which includes:

• enable: Attach the corresponding mouse interaction to the target(s).
• disable: Attach the interaction if not and disable it at once (only set the options).
• destroy: Destroy the interaction.
• save: Attach the interaction if not and save the current interaction state.
• load: If interaction attached, restore the target(s) to the last saved interaction state.

With mouse interactions attached, the corresponding interaction states, e.g. position of draggable, size of resizable, selected of selectable and order of sortable, will be send to server in the form of input$<id>_<state>. The default values can be overridden by setting the shiny option in the options parameter. Please see the vignette Introduction to shinyjqui for more details.

The functions jqui_draggabled(), jqui_droppabled(), jqui_resizabled(), jqui_selectabled() and jqui_sortabled() are deprecated. Please use the corresponding -able() functions instead.

Value

The same object passed in the ui parameter
library(shiny)
library(highcharter)

## used in ui
jquiResizable(actionButton('btn', 'Button'))
jquiDraggable(plotOutput('plot', width = '400px', height = '400px'),
  options = list(axis = 'x'))
jquiSelectable(
  div(
    id = 'sel_plots',
    highchartOutput('highchart', width = '300px'),
    plotOutput('ggplot', width = '300px'),
    options = list(classes = list('ui-selected' = 'ui-state-highlight'))
  )
)
jquiSortable(tags$ul(
  id = 'lst',
  tags$li('A'),
  tags$li('B'),
  tags$li('C'))
))

## used in server
## Not run:
  jquiDraggable('#foo', options = list(grid = c(80, 80))
  jquiDroppable('.foo', operation = "enable")

## End(Not run)

## use shiny input
if (interactive()) {
  shinyApp(
    server = function(input, output) {
      output$foo <- renderHighchart({
        hchart(mtcars, "scatter", hcaes(x = cyl, y = mpg))
      })
      output$position <- renderPrint({
        print(input$foo_position)
      })
    },
    ui = fluidPage(
      verbatimTextOutput('position'),
      jquiDraggable(highchartOutput('foo', width = '200px', height = '200px'))
    )
  )
}

## custom shiny input
func <- JS('function(event, ui){return $(event.target).offset();}')
options <- list(
  shiny = list(
    abs_position = list(
      dragcreate = func, # send returned value back to shiny when interaction is created.
      drag = func # send returned value to shiny when dragging.
    )
  )
)

jqui_draggable(highchartOutput('foo', width = '200px', height = '200px'),
  options = options)

---

### jqui_bookmarking

**Enable bookmarking state of mouse interactions**

**Description**

Enable shiny *bookmarking_state* of mouse interactions. By calling this function in server, the elements' position, size, selection state and sorting state changed by mouse operations can be saved and restored through an URL.

**Usage**

```
jqui_bookmarking()
```

---

### jqui_icon

**Create a jQuery UI icon**

**Description**

Create an jQuery UI pre-defined icon. For lists of available icons, see [http://api.jqueryui.com/theming/icons/](http://api.jqueryui.com/theming/icons/).

**Usage**

```
jqui_icon(name)
```

**Arguments**

- **name**
  
  Class name of icon. The "ui-icon-" prefix can be omitted (i.e. use "ui-icon-flag" or "flag" to display a flag icon)

**Value**

An icon element
OrderInput

Examples

```r
jqui_icon('caret-l-n')
library(shiny)

# add an icon to an actionButton
actionButton('button', 'Button', icon = jqui_icon('refresh'))

# add an icon to a tabPanel
tabPanel('Help', icon = jqui_icon('help'))
```

orderInput

A shiny input control to show the order of a list of items

Description

Display a list items whose order can be changed by drag and drop within or between orderInput(s). The current items order can be obtained from input$inputId_order.

Usage

```r
orderInput(
  inputId,
  label,
  items,
  as_source = FALSE,
  connect = NULL,
  item_class = c("default", "primary", "success", "info", "warning", "danger"),
  placeholder = NULL,
  width = "500px",
  ...
)
```

Arguments

- **inputId**: The input slot that will be used to access the current order of items.
- **label**: Display label for the control, or NULL for no label.
- **items**: Items to display, can be a list, an atomic vector or a factor. For list and atomic vector, if named, the names are displayed and the order is given in values. For factor, values are displayed and the order is given in levels.
- **as_source**: A boolean value to determine whether the orderInput is set as source mode. If source mode, items in this orderInput can only be dragged (copied) to the connected non-source orderInput(s) defined by connect argument. If non-source mode, items in the orderInput can be dragged (moved) within or toward other connected non-source orderInput(s) defined by connect argument.
connect optional, a vector of inputId(s) of other orderInput connects to. The behavior of the connected orderInput(s) depend on the as_source argument.

item_class One of the Bootstrap Button Styles to apply to each item.

placeholder A character string to show when there is no item left in the orderInput.

width The width of the input, e.g. '400px', or '100\ validateCssUnit.

... Arguments passed to shiny::tags$div which is used to build the container of the orderInput.

Value
A orderInput control that can be added to a UI definition.

Examples
orderInput('items1', 'Items1', items = month.abb, item_class = 'info')

## build connections between orderInputs
orderInput('items2', 'Items2 (can be moved to Items1 and Items4)', items = month.abb,
     connect = c('items1', 'items4'), item_class = 'primary')

## build connections in source mode
orderInput('items3', 'Items3 (can be copied to Items2 and Items4)', items = month.abb,
     as_source = TRUE, connect = c('items2', 'items4'), item_class = 'success')

## show placeholder
orderInput('items4', 'Items4 (can be moved to Items2)', items = NULL, connect = 'items2',
     placeholder = 'Drag items here...')

selectableTableOutput  Create a table output element with selectable rows or cells

Description
Render a standard HTML table with its rows or cells selectable. The server will receive the index of selected rows or cells stored in input$<outputId>_selected.

Usage
selectableTableOutput(outputId, selection_mode = c("row", "cell"))

Arguments
outputId output variable to read the table from

selection_mode one of "row" or "cell" to define either entire row or individual cell can be selected.
sortableCheckboxGroupInput

Details

Use mouse click to select single target, lasso (mouse dragging) to select multiple targets, and Ctrl + click to add or remove selection. In row selection mode, input$<outputId>_selected will receive the selected row index in the form of numeric vector. In cell selection mode, input$<outputId>_selected will receive a dataframe with rows and columns index of each selected cells.

Value

A table output element that can be included in a panel

See Also

shiny::tableOutput, sortableTableOutput

Examples

```r
## Only run this example in interactive R sessions
if (interactive()) {
  shinyApp(
    ui = fluidPage(
      verbatimTextOutput("selected"),
      selectableTableOutput("tbl")
    ),
    server = function(input, output) {
      output$selected <- renderPrint({input$tbl_selected})
      output$tbl <- renderTable(mtcars, rownames = TRUE)
    }
  )
}
```

Sortable Checkbox Group Input Control with Sortable Choices

Description

Render a group of checkboxes with multiple choices toggleable. The choices are also sortable by drag and drop. In addition to the selected values stored in input$<inputId>, the server will also receive the order of choices in input$<inputId>_order.

Usage

```r
sortableCheckboxGroupInput(
  inputId,
  label,
  choices = NULL,
  selected = NULL,
```
Arguments

inputId  The input slot that will be used to access the value.
label   Display label for the control, or NULL for no label.
choices List of values to show checkboxes for. If elements of the list are named then that name rather than the value is displayed to the user. If this argument is provided, then choiceNames and choiceValues must not be provided, and vice-versa. The values should be strings; other types (such as logicals and numbers) will be coerced to strings.
selected The values that should be initially selected, if any.
inline  If TRUE, render the choices inline (i.e. horizontally)
width   The width of the input, e.g. '400px', or '100%'; see validateCssUnit()
choiceNames List of names and values, respectively, that are displayed to the user in the app and correspond to the each choice (for this reason, choiceNames and choiceValues must have the same length). If either of these arguments is provided, then the other must be provided and choices must not be provided. The advantage of using both of these over a named list for choices is that choiceNames allows any type of UI object to be passed through (tag objects, icons, HTML code, ...), instead of just simple text. See Examples.
choiceValues List of names and values, respectively, that are displayed to the user in the app and correspond to the each choice (for this reason, choiceNames and choiceValues must have the same length). If either of these arguments is provided, then the other must be provided and choices must not be provided. The advantage of using both of these over a named list for choices is that choiceNames allows any type of UI object to be passed through (tag objects, icons, HTML code, ...), instead of just simple text. See Examples.

Value

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

See Also

shiny::checkboxGroupInput, sortableRadioButtons(), sortableTableOutput(), sortableTabsetPanel()

Examples

```r
## Only run this example in interactive R sessions
if (interactive()) {
  shinyApp(
    ui = fluidPage(
      sortableCheckboxGroupInput("foo", "SortableCheckboxGroupInput",
```
sortableRadioButtons

```r
choices = month.abb,
verbatimTextOutput("order")
)
server = function(input, output) {
  output$order <- renderPrint({input$foo_order})
}
```

**sortableRadioButtons**  *Create radio buttons with sortable choices*

**Description**
Create a set of radio buttons used to select an item from a list. The choices are sortable by drag and drop. In addition to the selected values stored in `input$<inputId>`, the server will also receive the order of choices in `input$<inputId>_order`.

**Usage**
sortableRadioButtons(
  inputId,
  label,
  choices = NULL,
  selected = NULL,
  inline = FALSE,
  width = NULL,
  choiceNames = NULL,
  choiceValues = NULL
)

**Arguments**
- **inputId**: The input slot that will be used to access the value.
- **label**: Display label for the control, or NULL for no label.
- **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). If this argument is provided, then `choiceNames` and `choiceValues` must not be provided, and vice-versa. The values should be strings; other types (such as logicals and numbers) will be coerced to strings.
- **selected**: The initially selected value (if not specified then defaults to the first value)
- **inline**: If TRUE, render the choices inline (i.e. horizontally)
- **width**: The width of the input, e.g. '400px', or '100%'; see `validateCssUnit()`.
choiceNames List of names and values, respectively, that are displayed to the user in the app and correspond to each choice (for this reason, choiceNames and choiceValues must have the same length). If either of these arguments is provided, then the other must be provided and choices must not be provided. The advantage of using both of these over a named list for choices is that choiceNames allows any type of UI object to be passed through (tag objects, icons, HTML code, ...), instead of just simple text. See Examples.

choiceValues List of names and values, respectively, that are displayed to the user in the app and correspond to each choice (for this reason, choiceNames and choiceValues must have the same length). If either of these arguments is provided, then the other must be provided and choices must not be provided. The advantage of using both of these over a named list for choices is that choiceNames allows any type of UI object to be passed through (tag objects, icons, HTML code, ...), instead of just simple text. See Examples.

Value A set of radio buttons that can be added to a UI definition.

See Also shiny::radioButtons, sortableCheckboxGroupInput, sortableTableOutput, sortableTabsetPanel

Examples

```r
## Only run this example in interactive R sessions
if (interactive()) {
  shinyApp(
    ui = fluidPage(
      sortableRadioButtons("foo", "SortableRadioButtons",
        choices = month.abb),
      verbatimTextOutput("order"))
    ),
    server = function(input, output) {
      output$order <- renderPrint({input$foo_order})
    }
  }
}
```

---

sortableTableOutput Create a table output element with sortable rows

Description

Render a standard HTML table with table rows sortable by drag and drop. The order of table rows is recorded in input$<outputId>_order.
Usage

```r
sortableTableOutput(outputId)
```

Arguments

```r
outputId output variable to read the table from
```

Value

A table output element that can be included in a panel

See Also

`shiny::tableOutput, sortableRadioButtons, sortableCheckboxGroupInput, sortableTabsetPanel, selectableTableOutput`

Examples

```r
## Only run this example in interactive R sessions
if (interactive()) {
  shinyApp(
    ui = fluidPage(
      verbatimTextOutput("rows"),
      sortableTableOutput("tbl")
    ),
    server = function(input, output) {
      output$rows <- renderPrint({input$tbl_row_index})
      output.tbl <- renderTable(mtcars, rownames = TRUE)
    }
  )
}
```

## sortableTabsetPanel

Create a tabset panel with sortable tabs

Description

Create a tabset that contains `shiny::tabPanel` elements. The tabs are sortable by drag and drop. In addition to the activated tab title stored in `input$id`, the server will also receive the order of tabs in `input$id_order`.

Usage

```r
sortableTabsetPanel(
  ..., id = NULL, selected = NULL,
)```
Arguments

... tabPanel() elements to include in the tabset

id
If provided, you can use input$id in your server logic to determine which of the current tabs is active. The value will correspond to the value argument that is passed to tabPanel().

selected
The value (or, if none was supplied, the title) of the tab that should be selected by default. If NULL, the first tab will be selected.

type
Use "tabs" for the standard look; Use "pills" for a more plain look where tabs are selected using a background fill color.

position
This argument is deprecated; it has been discontinued in Bootstrap 3.

Value

A tabset that can be passed to shiny::mainPanel

See Also

shiny::tabsetPanel, sortableRadioButtons, sortableCheckboxGroupInput, sortableTableOutput

Examples

## Only run this example in interactive R sessions
if (interactive()) {
  shinyApp(
    ui = fluidPage(
      sortableTabsetPanel(
        id = "tabs",
        tabPanel(title = "A", "AAA"),
        tabPanel(title = "B", "BBB"),
        tabPanel(title = "C", "CCC")
      ),
     verbatimTextOutput("order")
    ),
    server = function(input, output) {
      output$order <- renderPrint({input$tabs_order})
    }
  )
}
Index

Animation_effects, 2
Class_effects, 3
draggableModalDialog, 4
get_jqui_effects, 5
includeJqueryUI, 6
Interactions, 6
jqui_add_class (Class_effects), 3
jqui_draggable (Interactions), 6
jqui_resizable (Interactions), 6
jqui_selectable (Interactions), 6
jqui_sortable (Interactions), 6
jqui_show (Animation_effects), 2
jqui_icon, 10
jqui_remove_class (Class_effects), 3
jqui_resizable (Interactions), 6
jqui_resizabled (Interactions), 6
jqui_selectable (Interactions), 6
jqui_selectabled (Interactions), 6
jqui_sortable (Interactions), 6
jqui_sortabled (Interactions), 6
jqui_switch_class (Class_effects), 3
jqui_toggle (Animation_effects), 2
JS(), 8

orderInput, 11
removeModal(), 5
selectableTableOutput, 12, 17
shiny::checkboxGroupInput, 14
shiny::mainPanel, 18
shiny::modalDialog, 4
shiny::radioButtons, 16

shiny::tableOutput, 13, 17
shiny::tabPanel, 17
shiny::tabsetPanel, 18
shiny::tags, 8
sortableCheckboxGroupInput, 13, 16–18
sortableRadioButtons, 15, 17, 18
sortableRadioButtons(), 14
sortableTableOutput, 13, 16, 16, 18
sortableTableOutput(), 14
sortableTabsetPanel, 16, 17, 17
sortableTabsetPanel(), 14

tabPanel(), 18
tagList(), 8

validateCssUnit, 12
validateCssUnit(), 14, 15