Package 'dqshiny'

February 5, 2019

Type Package

Title Enhance Shiny Apps with Customizable Modules

Version 0.0.3

Author Richard Kunze, Mirjam Rehr

Maintainer Richard Kunze <richard.kunze@daqana.com>

Description Provides highly customizable modules to enhance your shiny apps. Includes layout independent collapsible boxes and value boxes, a very fast autocomplete input, rhandsontable extensions for filtering and paging and much more.

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

Imports shiny, htmltools

Suggests testthat, extrafont, ggplot2, jsonlite, rhandsontable, V8, htmlwidgets, shinytest

URL https://github.com/daqana/dqshiny

BugReports https://github.com/daqana/dqshiny/issues

NeedsCompilation no

Repository CRAN

Date/Publication 2019-02-05 14:23:19 UTC

R topics documented:

`add_class` *Change the state of a shiny interface element*

Description

Those functions can be used to change an elements classes.

Usage

```
add_class(ids, class_name)

remove_class(ids, class_name)

toggle_class(ids, class_name, condition = NULL)
```

Arguments

ids	Character vector, id(s) of the element(s) to change.
class_name	Name of the class to add/remove/toggle.
condition	Condition to use for toggling the class.

Note

If you have trouble with these functions, please make sure that you either - use any dqshiny element in your UI - load the package with library - use init at the beginning of your UI!

Author(s)

richard.kunze

See Also

Other js handler: enable, hide

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    tags$head(tags$style(".orange{background:#ff8f00}")),
    actionButton("add", "Add Class"),
    actionButton("remove", "Remove Class"),
    actionButton("toggle", "Toggle Class"),
    actionButton("toggle_cond", "Toggle Class with Condition"),
    checkboxInput("condition", "orange"),
    fluidRow(id = "row",
      dq_space(), # this is needed to make everything work
      actionButton("example", "EXAMPLE"),
      dq_space() # this is just for the alignment ;)
   )
  ),
  server = function(input, output) {
    observeEvent(input$add, add_class("row", "orange"))
    observeEvent(input$remove, remove_class("row", "orange"))
    observeEvent(input$toggle, toggle_class("row", "orange"))
    observeEvent(
      input$toggle_cond,
      toggle_class("row", "orange", input$condition)
    )
  }
)

}
```

add_js *Adds a custom JS function*

Description

Adds the given JS function definition to the current shiny web app.

Before adding anything the code will be parsed with V8 if the package is available on your machine. If package is available and parsing fails, an error will be thrown.

After successfully adding a function it can be used with run_js.

Usage

```
add_js(type, function_text)

run_js(type, ...)
```

Arguments

type	name of the function
function_text	JS function definition, this should be an anonymous function accepting exactly one argument
...	arguments to pass to the function, those will be combined to a list and end up as an array in JS, unnamed parameters will be available via params[0..], named parameters can also be used with params.name

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    add_js(type="addValues", "function(params) {
      alert('Result is: ' + (parseInt(params[0]) + parseInt(params[1])));
    }"),
    add_js(type="myName", "function(params) {
      alert('My name is ' + params.name);
    }"),
    actionButton("btn1", "Add Values"),
    actionButton("btn2", "What's my name?")
  ),
  server = function(input, output) {
```

```
    observeEvent(input$btn1, run_js(type = "addValues", 17, 25))
    observeEvent(input$btn2, run_js(type = "myName", name = "Paul"))
  }
)

}
```

`autocomplete_input` *Creates an autocomplete text input field*

Description

autocomplete_input creates an autocomplete text input field, showing all possible options from a given list under the input while typing. Alternative to very slow select(ize) inputs for (very) large option lists.

update_autocomplete_input changes the value or the options of an autocomplete input element on the client side.

Usage

```
autocomplete_input(id, label, options, value = "", width = NULL,
  placeholder = NULL, max_options = 0, hide_values = FALSE)

update_autocomplete_input(session, id, label = NULL, options = NULL,
  max_options = NULL, value = NULL, placeholder = NULL,
  hide_values = NULL)
```

Arguments

`id`	id of the element
`label`	label to show for the input, NULL for no label
`options`	list (or vector) of possible options
`value`	initial value
`width`	optional, the width of the input, see `validateCssUnit`
`placeholder`	optional character specifying the placeholder text
`max_options`	optional numeric specifying the maximum number of options to show (for performance reasons)
`hide_values`	optional boolean indicating whether to show values under labels or not
`session`	the shiny session object

Value

autocomplete_input: shiny input element

update_autocomplete_input: message to the client

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
opts <- sapply(1:100000, function(i) paste0(sample(letters, 9), collapse=""))
shinyApp(
  ui = fluidPage(
    fluidRow(
      column(3,
        autocomplete_input("auto1", "Unnamed:", opts, max_options = 1000),
        autocomplete_input("auto2", "Named:", max_options = 1000,
          structure(opts, names = opts[order(opts)])),
        autocomplete_input("auto3", "Big data:", NULL, max_options = 1000,
          placeholder = "Big data taking several seconds to load ..."),
        actionButton("calc", "Calculate")
      ), column(3,
        tags$label("Value:"), verbatimTextOutput("val1", placeholder = TRUE),
        tags$label("Value:"), verbatimTextOutput("val2", placeholder = TRUE)
       )
    )
  ),
  server = function(input, output, session) {
    output$val1 <- renderText(as.character(input$auto1))
    output$val2 <- renderText(as.character(input$auto2))
    observeEvent(input$calc, {
      Sys.sleep(3)
      update_autocomplete_input(session, "auto3", placeholder = "Loaded!",
        options = rownames(mtcars))
    })
  }
)

}
```

---

click *Programmatically click a button*

Description

Programmatically click a button

Usage

```
click(ids)
```

Arguments

ids	id(s) of button to be clicked

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(
      numericInput("clickBtn", "Click Btn Nr:", 1, 1, 3, 1),
      actionButton("btn1", "1"),
      actionButton("btn2", "2"),
      actionButton("btn3", "3"),
      dq_space(),
      textOutput("result")
    )
  ),
  server = function(input, output) {
    output$result <- renderText(paste("Buttons clicked:",
      toString(c(input$btn1, input$btn2, input$btn3))
    ))
    observeEvent(input$clickBtn, click(paste0("btn", input$clickBtn)),
                 ignoreInit = TRUE)
  }
)

}
```

---

create_dq_box_group *Create a dq box group*

Description

Create a dq box group which automatically collapses all other boxes whenever one box is opened.

Usage

```
create_dq_box_group(session, ...)
```

Arguments

session	shiny session object
...	box ids to be combined to one group

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(
      column(6,
        dq_box("Random Input1", dq_space(), "End of Content", id = "box1",
          title = "Box1", width = 12, open_callback = TRUE),
        dq_box("Random Input2", dq_space(), "End of Content", id = "box2",
          title = "Box2", width = 9, collapsed = TRUE, open_callback = TRUE),
        dq_box("Random Input3", dq_space(), "End of Content", id = "box3",
          title = "Box3", width = 12, collapsed = TRUE, open_callback = TRUE)
      )
    )
  ),
  server = function(input, output, session) {
    create_dq_box_group(session, "box1", "box2", "box3")
  }
)

}
```

---

dqshiny *dqshiny: Enhance Shiny Apps with Customizable Modules*

Description

Provides highly customizable modules to enhance your shiny apps. Includes layout independent collapsible boxes and value boxes, a very fast autocomplete input, rhandsontable extensions for filtering and paging and much more.

Details

There is a demo 'dqshiny-base-features' showing some of the functionalities provided in this package. Furthermore you can have a look at the github page for examples and additional information.

dq_accordion *Accordion module to show several collapsible boxes*

Description

Creates an accordion object where one of the contents can be shown and the others will be hidden. The currently activated panel will be available to R over the input$id element.

Usage

```
dq_accordion(id, titles, contents, options = NULL, sortable = FALSE,
  bg_color = "#ff8f00", hover = TRUE, style = "", icons = c(rotate
  = "angle-right"))
```

Arguments

id	id of the element
titles	character, titles to show in the accordion headers
contents	list of contents, can be character, shiny tags, nested lists of shiny tags ...
options	optional list of jquery-ui options to customize accordions behavior, for a full list of possible options have a look at: http://api.jqueryui.com/accordion
sortable	optional logical indicating whether the accordion parts should be rearrangeable or not
bg_color	optional character specifying the background color of the headers, can be any valid HTML color code
hover	optional logical indicating whether headers should have an hover effect or not
style	optional character for additional header style attributes
icons	optional named character vector of length one or two indicating the FontAwesome icons to be used in front of the header title showing the state of the content (open or closed)

Value

shiny div holding the accordion

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
titles <- c("Section 1", "Section 2", "Section 3")
contents <- list("Lorem ipsum..", "Lorem ipsum..", tags$p("Lorem ipsum.."))
shinyApp(
  ui = fluidPage(
    fluidRow(
      column(5, dq_accordion("myAccordion", titles, contents, hover = FALSE,
        style = "border:1px solid red;margin-top: 5px;color: red;"
      ), dq_space(),
      dq_accordion("myAccordion2", titles, contents,
        bg_color = NULL, options = list(animate = 500, collapsible = TRUE),
        icons = c(open = "hand-point-down", closed = "hand-point-right")
      ), dq_space(),
      dq_accordion("myAccordion3", titles, contents,
        bg_color = "pink", icons = NULL, sortable = TRUE
      ))
    )
  ), server = function(input, output) {
    observeEvent(input$myAccordion, print(input$myAccordion))
  }
)

}
```

`dq_add_selectize_options` *Adds selectizeOptions to a column of rhandsontable*

Description

dq_add_selectize_options adds selectizeOptions to a column of a rhandsontable to be used with the selectize editor. Especially useful if each cell should have individual dropdowns. It will also set the type and editor for the specified column.

dq_as_selectize_options converts the given vector of options into a proper selectize options list. Names of the given vector will be used to specify labels for the options. Further selectize attributes can be set via additional named parameters.

Usage

```
dq_add_selectize_options(hot, rows, col, options, ...)

dq_as_selectize_options(options, ...)
```

Arguments

hot	rhandsontable object
rows	vector of row indices, NULL means the whole column will be filled
col	column index or name, must be scalar
options	character vector or list to be used as selectize options, names will be used as labels for the options, dq_add_selectize_options can also handle lists of vectors, where each vector will be used to specify the options of one cell
...	additional parameters to be passed to selectize

Value

dq_add_selectize_options: updated rhandsontable object

dq_as_selectize_options: list containing all options and additional settings

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(rhandsontable)
library(shiny)
hw <- c("Hello", "my", "funny", "world!",
  "Those", "are", "some", "really", "random", "words!")

options <- lapply(1:10, function(x) c(Name1 = sample(hw, 1),
  Name2 = sample(hw, 1), Name3 = sample(hw, 1)))
ch <- sample(hw, 3)
names(ch) <- sample(hw, 3)
selectize <- dq_as_selectize_options(ch, create = TRUE)

empty <- rep("", 10)
df <- data.frame(Unlabled=empty, Labled=empty,
  Multiple=empty, stringsAsFactors = F)

shinyApp(
  ui = fluidPage(
    dq_space(),
    rHandsontableOutput("randomTable")
  ),
  server = function(input, output) {
    output$randomTable <- renderRHandsontable({
      rhandsontable(df, stretchH = "all") %>%
        dq_add_selectize_options(NULL, 1, lapply(options, unname)) %>%
        hot_col(2, editor = "selectize", selectizeOptions = selectize) %>%
        dq_add_selectize_options(NULL, "Multiple", options, maxItems = 2)
    })
```

```
  }
)

}
```

---

dq_box *Creates a html box with specified parameters*

Description

Creates a fully customizable HTML box holding the given content. Can be made collapsible and nested.

Function to update the collapsed state of a dq_box.

Usage

```
dq_box(..., id = NULL, title = NULL, color = "#000",
  bg_color = "#ff8f00", fill = TRUE, width = 6L, height = NULL,
  offset = 0L, collapsible = FALSE, collapsed = FALSE,
  open_callback = FALSE)

update_dq_box(id, collapsed = NULL, silent = FALSE)
```

Arguments

...	Tags to add to the box as children.
id	ID of the box, can be used to observe collapse events via input[[paste0(id, "_collapser")]].
title	Title of the box, always visible.
color	Optional, sets the font color of the box, can be any valid html color, defaults to black.
bg_color	Optional, sets the background color of the box, can be any valid html color, defaults to dq primary orange.
fill	Optional logical, decide whether to fill the body with background color or not, if not, a border of this color will be set instead.
width	Optional, width of the box measured in bootstrap columns.
height	Optional, height of the box, can be numeric, which will result in pixels, or measured in any valid CSS3 length format, if the given body is larger than this value overflow will be set to auto, resulting in a scrollable body.
offset	Optional, offset of the resulting column measured in bootstrap columns.
collapsible	Optional logical, whether the box is collapsible or not.
collapsed	Optional logical, whether the box is initially collapsed or not.
open_callback	optional logical, whether to send messages whenever the state of the box changes or not, events will be available via input[[paste0(id, "_open")]]
silent	optional logical indicating to suppress events or not

Value

bootstrap column holding the box

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(
      dq_box(
        title = "Say Hello to my children", id = "bigBox", collapsed = TRUE,
        dq_infobox("hallo", 2, "Welt", icon("hashtag"),
                   bg_color = "black", color = "#D00"),
        dq_box(
          title = "Box in the box", bg_color = "pink", width = 8,
          dq_infobox("in the box...", 2, "in the box!", width = 12,
                     bg_color = "white", color = "#0D0")
        )
        ),
      column(3, actionButton("toggle", "Toggle Box"))
     )
  ),
  server = function(input, output) {
    observeEvent(input$toggle, update_dq_box("bigBox"))
  }
)

}
```

---

dq_busy *Adds a loading image if shiny is busy*

Description

Adds a loading image to the page which is visible when shiny is busy.

Usage

```
dq_busy(icon_path = NULL, time = 500, animation = "fadeIn")
```

Arguments

icon_path	optional character, icon source path
time	optional integer indicating the animation time of the loader. Can be useful to omit loader for many short loading moments, but show it if calculation really needs some time. Set to 0 to disable animations.
animation	optional character specifying the animation of the loader

Value

shiny tag holding the icon

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    dq_busy(time = 1500),
    actionButton("button", "make me busy")
  ),
  server = function(input, output) {
    observeEvent(input$button, Sys.sleep(3))
  }
)

}
```

---

dq_handsontable_output

Adds an uiOutput and renders an enhanced rhandsontable html widget

---

Description

dq_handsontable_output adds a fluidRow containing a column with the given width, ready to support a dq_handsontable.

dq_render_handsontable renders a rhandsontable into the given uiOutput id with the given data and parameters. Can also contain several filters to filter the data and a feature to split the table into several pages with a given page size. The function will also add all needed observeEvents to establish the required functionalities. If table is not readOnly, all user inputs will automatically stored and updated independent from any filters, sortings or pages.

Usage

```
dq_handsontable_output(id, width = 12L, offset = 0L)

dq_render_handsontable(id, data, context = NULL, filters = "T",
  page_size = 25L, reset = TRUE, sorting = NULL, columns = NULL,
  width_align = FALSE, horizontal_scroll = FALSE, table_param = NULL,
  cols_param = NULL, col_param = NULL, cell_param = NULL,
  session = shiny::getDefaultReactiveDomain())
```

Arguments

id	id of the element
width	width of the table in bootstrap columns
offset	optional offset of the column
data	data to show in the table, should be a data.frame'ish object, can also be reactive(Val) or a reactiveValues object holding the data under the given id (e.g. myReactiveValues[[id]] <- data). In case of reactiveVal(ues) data will always be in sync with user inputs.
context	the context used to specify all ui elements used for this table, can be omitted which ends up in a randomly generated context NOTE: this parameter is deprecated and will be removed soon
filters	optional, adds filters for each column, types must be one of "Text", "Select", "Range", "Date", "Auto" or "" (can be abbreviated) to add a Text-, Select-, Range-, DateRange-, AutocompleteInput or none, vectors of length one will add a filter of this type for each column and NA will try to guess proper filters, can also contain nested lists specifying type and initial value (e.g. list(list(type = "T", value = "init"), NA, "T", ...))
page_size	optional integer, number of items per page, can be one of 10, 25, 50, 100 or any other value(s) which will be added to this list, first value will be used initially, NULL will disable paging at all
reset	optional logical, specify whether to add a button to reset filters and sort buttons to initial values or not
sorting	optional, specify whether to add sort buttons for every column or not, as normal rhandsontable sorting won't work properly when table is paged, value can be logical of length one or a vector specifying the initial sort "col"umn and "dir"ection e.g. c(dir="down", col="Colname")
columns	optional, specify which columns to show in the table, useful in combination with reactive values, which will still hold all the data
width_align	optional boolean to align filter widths with hot columns, should only be used with either horizontal_scroll, stretchH = "all" or a table fitting in its output element
horizontal_scroll	optional boolean to scroll the filter row according to the hot table, especially useful for tables with many columns
table_param	optional list, specify parameters to hand to rhandsontable table element

cols_param	optional list, specify parameters to hand to rhandsontable cols elements
col_param	optional list of lists to specify parameters to hand to rhandsontable col elements
cell_param	optional list of lists to specify parameters to hand to rhandsontable cells
session	shiny session object

Value

dq_handsontable_output: fluidRow containing the output fields

dq_render_handsontable: the given data

Author(s)

richard.kunze

See Also

rhandsontable, hot_cols and hot_col

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    dq_handsontable_output("randomTable", 9L)
  ),
  server = function(input, output, session) {
    hw <- c("Hello", "my", "funny", "world!")
    data <- data.frame(A = rep(hw, 500), B = hw[c(2,3,4,1)],
      C = 1:500, D = Sys.Date() - 0:499, stringsAsFactors = FALSE)
    dq_render_handsontable("randomTable", data,
      filters = c("A", NA, NA, NA), sorting = c(dir = "up", col = "B"),
      page_size = c(17L, 5L, 500L, 1000L), width_align = TRUE,
      col_param = list(list(col = 1L, type = "dropdown", source = letters)),
      cell_param = list(list(row = 2:9, col = 1:2, readOnly = TRUE))
    )
  }
)

}
```

dq_helptag *Creates a help symbol with the given title as popover*

Description

Creates a help symbol with the given title as a popover. Trigger to show the popover can either be a mouse hover or a mouse click to show it until the user clicks again. Info tags can be added to shiny input labels via tagLists.

Usage

```
dq_helptag(title, trigger = "hover", width = 200, style = NULL)
```

Arguments

title	Title to show in the popover.
trigger	Optional, can either be 'hover' to show the popover while the mouse is over the icon or 'focus' to show it while the tag is in focus.
width	Optional, width of the popover, can be numeric or character including any valid CSS unit.
style	Optional character, additional style attributes for the tag.

Value

shiny tag holding the help icon

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init(),
    fluidRow(
      column(3,
        dq_helptag(
        "This info is visible after an click!<br>
        Line breaks are also possible btw...",
                  trigger = "focus"
                  ),
        textInput("importantValue",
          tagList("Important Value",
```

```
            dq_helptag(
            "This is an important value, you have to put something in!"
            ))
      ))
    )),
  server = function(input, output) {
  }
)

}
```

---

dq_hot_cell *Configure individual cells of rhandsontable widget*

Description

Configure individual cells of a rhandsontable widget. Can be used just like hot_cols or hot_col to specify custom options or cells. All possible combinations of row and column indices will be set.

Usage

```
dq_hot_cell(hot, row, col, ...)
```

Arguments

hot	rhandsontable object
row	integer vector specifying the rows to configure
col	integer vector specifying the columns to configure
...	parameters to be set in the cells, can be all rhandsontable parameters and additional custom ones used with custom renderers or editors

Author(s)

richard.kunze

Examples

```
df <- data.frame(readOrWrite = rep(c("readOnly", "change me!"), 5),
  secret = rep("tops3cr3t", 10), stringsAsFactors = FALSE)

hot <- rhandsontable::rhandsontable(df, rowHeaders = NULL)
hot <- dq_hot_cell(hot, seq(1, 10, 2), 1:2, readOnly = TRUE)
hot <- dq_hot_cell(hot, seq(1, 10, 2), 2, type = "password")
hot
```

dq_hot_date_renderer *rhandsontable renderer*

Description

dq_hot_date_renderer: Renderer to show rhandsontable dates in proper formatting.

dq_hot_empty_renderer: Renderer to highlight empty cells in rhandsontable.

dq_hot_html_renderer: Renderer to replace missing "html" rhandsontable renderer.

dq_hot_selectize_renderer: Renderer to properly display multiple selectize options.

Usage

```
dq_hot_date_renderer()

dq_hot_empty_renderer(renderer = "Autocomplete")

dq_hot_html_renderer()

dq_hot_selectize_renderer()
```

Arguments

renderer	rhandsontable base renderer to be adjusted, can be one of ("Autocomplete", "Base", "Checkbox", "Date", "Dropdown", "Html", "Numeric", "Password", "Text", "Time")

Value

character containing js renderer

Author(s)

richard.kunze

Examples

```
df <- data.frame(empty = rep(c("value", ""), 5),
  html = paste0("<div style='background:#ff",sprintf("%x",25*1:10),"ff'> </div>"),
  date = seq(from = Sys.Date(), by = "days", length.out = 10),
  stringsAsFactors = FALSE)

hot <- rhandsontable::rhandsontable(df, rowHeaders = NULL)
hot <- rhandsontable::hot_col(hot, 1, renderer = dq_hot_empty_renderer())
hot <- rhandsontable::hot_col(hot, 2, renderer = dq_hot_html_renderer())
hot <- rhandsontable::hot_col(hot, 3, renderer = dq_hot_date_renderer())
hot
```

dq_htmltable *Create a HTML table containing the given list of elements*

Description

Creates a HTML table containing the given list of elements. Every element of the list should be a vector of the same length to ensure correct design. The align parameter can be either a character, which will result in all cells having the same alignment, a character vector showing the alignment for every column or a list of character vectors specifying the alignment of every cell.

Usage

```
dq_htmltable(elements, id = NULL, align = "left",
  head_align = "center", borders = "none")
```

Arguments

elements	List of elements to show in the list, each element of the list should be a row of the table. Can also be a data.frame with names used as header.
id	Optional, character specifying the elements id.
align	Optional, character (vector) or list of characters showing the alignment of the table for each column/cell, can be one of "left", "right" or "center", defaults to "left", can be abbreviated.
head_align	Optional character vector of header alignments, defaults to "center", can be abbreviated.
borders	Optional character specifying the desired borders to show. Can be either a single character vector of length one with one of c("inner", "outer", "all", "tex") or a list of character vectors specifying all borders by hand. Possible values are "top", "right", "bottom", "left". The first entry of the list will be used for the header row if given.

Value

HTML table

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
lets <- data.frame(lower = letters[1:5], UPPER = LETTERS[1:5])
shinyApp(
```

```
  ui = fluidPage(
    dq_space(), fluidRow(
      column(3, dq_htmltable(
        list(list("Description", icon("hashtag")),
             list("Value", textInput("value", NULL)),
             list("Result", textOutput("result"))),
        borders = "inner"
      )),
      column(2, dq_htmltable(lets, borders = "outer")),
      column(3, dq_htmltable(
        list(c("Left", "Center", "Right"), c("Center", "Right", "Left")),
        align = list(c("LEFT", "center", "right"), c("c", "R", "l")),
        borders = "all"
      )),
      column(2, dq_htmltable(lets, borders = "tex")),
      column(2, dq_htmltable(lets, borders = list(
        c("top left", "bottom right"), # header
        c("", ""), c("", ""), c("", ""), c("", ""), c("bottom", "bottom")
      )))
    )),
  server = function(input, output) {
    output$result <- renderText(input$value)
  }
)

}
```

---

dq_icon *Creates an icon element*

Description

Creates a html icon element with the specified icon name from the given library.

Usage

```
dq_icon(icon, lib = "font-awesome", color = "#ff8f00",
  bg_color = NULL, size = NULL, ...)
```

Arguments

icon	name of the icon to show
lib	library used, needed to append the proper dependency
color	icon color, can be any valid CSS color code
bg_color	icon background color, can be any valid CSS color code
size	character specifying the size of the icon, can be one of "xs", "sm", "lg", "2x", "3x", "4x", "5x", "6x", "7x", "8x", "9x", "10x"
...	additional attributes like style or class

Value

icon html element

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
addResourcePath("images", system.file("www", "img", package = "dqshiny"))
shinyApp(
  ui = fluidPage(
    dq_icon("table", size = "4x"),
    dq_icon("check", color = "red", lib = "glyphicon", size = "2x"),
    dq_icon("phone", bg_color = "green", size = "lg"),
    dq_icon("images/logo_daqana.svg", size = "3x")
  ),
  server = function(input, output, session) {}
)

}
```

---

dq_infobox *Creates an info box with given texts*

Description

Creates an info box with given texts

Usage

```
dq_infobox(title, value = NULL, subtitle = NULL, icon = NULL,
  bg_color = "#FF8F00", color = "white", width = 4, href = NULL,
  fill = TRUE)
```

Arguments

title	Title of the box.
value	Optional, value to show under the title.
subtitle	Optional, subtitle to show under the value.
icon	Optional, icon to show on the left, can be any string, shiny icon or just NULL to omit it.

bg_color	Optional, sets the background color of the box, can be any valid html color, standard is dq primary orange.
color	Optional, sets the font color of the box, can be any valid html color, standard is white.
width	Optional, width of the box measured in bootstrap columns.
href	Optional, link target of the box.
fill	Optional, logical, fill the box with background color or surround it by box-shadow.

Value

column holding the info box

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init(),
    fluidRow(
      dq_infobox(
      "hello", 2, "world", shiny::icon("hashtag"),
      bgColor = "black", color = "#D00"
      ),
      dq_infobox("hello", "2", href="https://www.google.com"),
      dq_infobox("hello", 2, "world", "YOU", fill=F)
    )
  ),
  server = function(input, output) {
  }
)

}
```

---

dq_render_svg *Render ggplot2 figure as svg*

Description

Returns ggplot2 svg image for shiny::imageOutput

Usage

```
dq_render_svg(gg, path = NULL, width = 1200, height = 500,
  alt = "", pdf = FALSE, png = FALSE, delete_file = TRUE)
```

Arguments

gg	reactive or non-reactive ggplot2 object
path	directory where images are stored (optional, default = NULL, creates tmp files)
width	plot width (optional, default = 1200)
height	plot height (optional, default = 500)
alt	alternative image title (optional, default = "")
pdf	boolean variable controlling if pdf output is generated (optional, default = FALSE)
png	boolean variable controlling if png output is generated (optional, default = FALSE)
delete_file	parameter for shiny::renderImage function

Value

list containing ggplot2 figure information

Author(s)

david.breuer

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    selectInput("select","Number of bars", choices=c(4,5,6)),
    br(),
    imageOutput("plot")
  ),
  server = function(input, output) {
    gf <- reactive({L <- as.integer(input$select)
                    gg <- ggplot2::ggplot(data=data.frame(x=seq(L), y=seq(L)),
                      ggplot2::aes(x=x, y=y)) + ggplot2::geom_bar(stat = "identity")
    })
    output$plot <- dq_render_svg(gf)
  }
)

}
```

dq_space *Create an empty div for spacing*

Description

Creates an empty div with the desired height clearing the space on the desired sites.

Usage

```
dq_space(height = 30, clear = "both")
```

Arguments

height	Height of the space, can be any valid CSS unit, validation will be done with validateCssUnit.
clear	Optional, can be one of 'both', 'left', 'right' and 'none' to specify which elements should be cleared, will be 'both' if given value is omitted or none of these.

Value

shiny div for spacing

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(
      dq_box(title = "I need space!!", dq_space(200)),
      dq_box(
      title = "I need more space", collapsible = TRUE, collapsed = TRUE,
            "...but I'm collapsible!", dq_space("50vh")
            )
    )),
  server = function(input, output) {
  }
)

}
```

enable *Change the state of a shiny interface element*

Description

Those functions can be used to change an elements disabled status.

Usage

```
enable(ids)

disable(ids)

toggle_state(ids, condition = NULL)
```

Arguments

ids	Character vector, id(s) of the element to enable/disable.
condition	Condition to be used for toggling the state (TRUE = disabled).

Note

If you have trouble with these functions, please make sure that you either - use any dqshiny element in your UI - load the package with `library` - use `init` at the beginning of your UI!

Author(s)

richard.kunze

See Also

Other js handler: `add_class`, `hide`

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    actionButton("btn_enable", "Enable"),
    actionButton("btn_disable", "Disable"),
    actionButton("btn_toggle", "Toggle State"),
    actionButton("btn_toggle_cond", "Toggle State with Condition"),
    actionButton("btn_toggle_all", "Toggle All States"),
    checkboxInput("condition", "Disabled"),
    dq_space(), # this is needed to make everything work
    actionButton("example1", "EXAMPLE1"),
```

```
    actionButton("example2", "EXAMPLE2"),
    actionButton("example3", "EXAMPLE3")
  ),
  server = function(input, output) {
    observeEvent(input$btn_enable, enable("example1"))
    observeEvent(input$btn_disable, disable("example1"))
    observeEvent(input$btn_toggle, toggle_state("example1"))
    observeEvent(input$btn_toggle_cond,
      toggle_state("example1", input$condition)
    )
    observeEvent(input$btn_toggle_all,
      toggle_state(c("example1", "example2", "example3"))
    )
  }
)

}
```

`hidden` *Sets the initial state of all given tags*

Description

Hidden only hides elements on the top level, so showing these elements will also show all children of them.

Disabled will recursively traverse the given elements and its children and set all inputs, buttons, selects and textfields to be disabled.

Usage

```
hidden(...)

disabled(...)
```

Arguments

`...`	tags to add to the ui, can be a single element or nested tagLists

Value

tags with the state change

Note

If you have trouble with these functions, please make sure that you either - use any dqshiny element in your UI - load the package with `library` - use `init` at the beginning of your UI!

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init(),
    actionButton("btn1", "Toggle Display"),
    actionButton("btn2", "Toggle State"),
    hidden(disabled(actionButton("btn3", "Hello")))
  ),
  server = function(input, output) {
    observeEvent(input$btn1, toggle("btn3"))
    observeEvent(input$btn2, toggle_state("btn3"))
  }
)

}
```

---

hide *Change the state of a shiny interface element*

Description

Those functions can be used to change an elements visibility status.

Usage

```
hide(ids)

show(ids)

toggle(ids, condition = NULL)
```

Arguments

ids	Character vector, id(s) of the element to hide/show.
condition	Condition to be used for toggling the visibility (TRUE = visible).

Note

If you have trouble with these functions, please make sure that you either - use any dqshiny element in your UI - load the package with `library` - use `init` at the beginning of your UI!

Author(s)

richard.kunze

See Also

Other js handler: add_class, enable

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init(),
    actionButton("hide", "Hide"),
    actionButton("show", "Show"),
    actionButton("toggle", "Toggle"),
    actionButton("toggle_cond", "Toggle Visibility with Condition"),
    actionButton("toggle_all", "Toggle All Visibilities"),
    checkboxInput("condition", "Visible"),
    actionButton("example1", "EXAMPLE1"),
    actionButton("example2", "EXAMPLE2"),
    actionButton("example3", "EXAMPLE3")
  ),
  server = function(input, output) {
    observeEvent(input$hide, hide("example1"))
    observeEvent(input$show, show("example1"))
    observeEvent(input$toggle, toggle("example1"))
    observeEvent(input$toggle_cond,
      toggle("example1", input$condition)
    )
    observeEvent(input$toggle_all,
      toggle(c("example1", "example2", "example3"))
    )
  }
)

}
```

---

icon_state_button *Creates a state button showing different icons*

Description

Creates a state button showing different states by different icons.

Changes the value or the options of an icon_state_button on the client side.

Usage

```
icon_state_button(id, states, value = NULL, ...)

update_icon_state_button(session, id, states = NULL, value = NULL)
```

Arguments

id	id of the element
states	character of possible states, must be valid FontAwesome icon names icon
value	optional value, can be integer position in states or character giving the state, must be of length one
...	additional parameters like CSS classes or styles
session	the shiny session object

Value

icon_state_button: shiny input element

update_icon_state_button: message to the client

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
hands <- paste0("hand-o-", c("up", "right", "down", "left"))
shinyApp(
  ui = fluidPage(
    fluidRow(column(12,
      icon_state_button("sort", c("sort", "sort-up", "sort-down")),
      icon_state_button("hands", hands, 1),
      icon_state_button("mood", c("smile-o", "meh-o", "frown-o"), "smile-o"),
      br(), actionButton("makeStars", "I like stars")
    ))
  ),
  server = function(input, output, session) {
    observeEvent(input$makeStars, update_icon_state_button(
      session, "mood", c("star", "star-half-o", "star-o"), "star"
    ))
  }
)

}
```

init *Initializes dqshiny*

Description

Can be used inside the shinyUI to add dq shiny resource path. Will be automatically called by loading the package and using any of the dqshiny elements.

Usage

```
init()
```

Value

dqshiny CSS and JS dependency

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init()
  ),
  server = function(input, output) {}
)

}
```

init_fonts *Initializes dqshiny fonts for figures*

Description

Only needed for highcharter/ggplot2 figures

Usage

```
init_fonts(paths, pattern = "ttf$", silent = TRUE)
```

Arguments

paths	vector of paths to look in
pattern	pattern to th files have to match
silent	optional logical indicating whether message and warnings should be printed or not

Author(s)

david.breuer

---

range_filter *numeric range filter for data.frames*

Description

range_filter filters a given data frame with the given filter values. Names of the given ranges vector should be the indices of the corresponding data frame columns.

Usage

```
range_filter(df, ranges)
```

Arguments

df	data frame to filter
ranges	numeric (or convertable) vector with the filter ranges, should have length of data or being named

Value

range_filter: filtered data frame

Author(s)

richard.kunze

render_dq_box_group *Directly render a set of dq_boxes as a group*

Description

Directly render a set of dq_boxes into an uiOutput element. All given dq_boxes will be prepared automatically, meaning that they become 'collapsible' and 'open_callback'ed.

Usage

```
render_dq_box_group(..., open = NULL)
```

Arguments

...	a set of dq_boxes
open	optional integer or character of length one, specifying the initially opened box

Value

fluidRow containing the grouped dq_boxes

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(column(6, uiOutput("myGroup")))
  ),
  server = function(input, output) {
    output$myGroup <- render_dq_box_group(
      dq_box("Random Input1", dq_space(), "End of Content",
        title = "TestBox1", width = 12),
      dq_box("Random Input2", dq_space(), "End of Content",
        title = "TestBox2", width = 9),
      dq_box("Random Input3", dq_space(), "End of Content",
        title = "TestBox3", width = 12),
      open = 3L)
  }
)

}
```

render_hot *Function to (re)render an existing rhandsontable*

Description

Function to send a message to js to render an existing rhandsontable object. E.g. needed when handsontable is used with stretchH = 'all' and a select callback to choose a dataset and present its data to the user. Handsontable won't properly render the table if the height of the page increases due to some visibility changes and a scroll bar appears.

Usage

```
render_hot(id)
```

Arguments

id	id of the hot object

Value

sent message

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
library(rhandsontable)
shinyApp(
  ui = fluidPage(
    dq_handsontable_output("random", 9),
    actionButton("render", "Render HoT"),
    fluidRow(id="bigRow", class="hidden",
      style="height:100vh;background:#ff8f00;")
  ),
  server = function(input, output) {
    hw <- c("Hello", "my", "funny", "world!")
    data <- data.frame(A=hw, B=hw[c(2,3,4,1)], C=1:4, D=Sys.Date() - 0:3,
      stringsAsFactors = FALSE)
    dq_render_handsontable("random", data, "rand",
      filters = c("S", "T", "R", "R"),
      table_param = list(rowHeaders = NULL, selectCallback = TRUE))
    observeEvent(input$random_select, toggle("bigRow"))
    observeEvent(input$render, render_hot("random"))
```

```
  }
)

}
```

`reset_slider_input` *Function to reset a slider input*

Description

Function to send a message to js to reset an existing slider input. Use it ro restore a slider input's initial values.

Usage

```
reset_slider_input(id)
```

Arguments

`id`	id of the slider input to reset

Value

sent message

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    init(),
    sliderInput("mySlider", "Change me", 0, 200, c(90, 117)),
    actionButton("btn1", "Reset slider")
  ),
  server = function(input, output) {
    observeEvent(input$btn1, reset_slider_input("mySlider"))
  }
)

}
```

text_filter *text filter for data.frames*

Description

text_filter filters a given data frame with the given filter values. Names of the given values vector should be the indices of the corresponding data frame columns. All filters are case-ignoring.

Usage

```
text_filter(df, values)
```

Arguments

df	data frame to filter
values	character array with the filter values, should have length of data or being named

Value

text_filter: filtered data frame

Author(s)

richard.kunze

time_input *Creates a time input field*

Description

time_input creates a time input field for correctly formatted time values

update_time_input changes the value/label or placeholder of an time input element on the client side.

Usage

```
time_input(id, label, value = "", min = NULL, max = NULL,
  format = "HH:mm", placeholder = NULL, width = NULL, color = NULL,
  use_material_picker = FALSE)

update_time_input(session, id, label = NULL, value = NULL,
  min = NULL, max = NULL, placeholder = NULL)
```

Arguments

id	id of the element
label	label to show for the input, NULL for no label
value	initial value
min	minimum time value, must follow the specified format, e.g. "08:00"
max	maximum time value, must follow the specified format, e.g. "17:00"
format	format to use for the time string, can be any valid moment.js time format (NOTE: this will only work with the material time picker and can't be updated!)
placeholder	optional character specifying the placeholder text
width	optional, the width of the input, see validateCssUnit
color	color of the watch hand (of material time picker)
use_material_picker	boolean to specify if the input should be a simple time (text) input or use the bootstrap material time picker
session	the shiny session object

Value

time_input: shiny input element

update_time_input: message to the client

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
shinyApp(
  ui = fluidPage(
    fluidRow(
      column(3,
        time_input("time1", "Simple:", value = "12:34"),
        time_input("time2", "Fancy:", use_material_picker = TRUE),
        actionButton("update", "Update")
      )
    )
  ),
  server = function(input, output, session) {
    observeEvent(input$update, {
      update_time_input(session, "time2", value = "12:34")
    })
  }
)
```

```
}
```

`video_box` *Adds a video box to the app*

Description

video_box adds a box holding the given video file, which will be hidden initially. Supports multiple sources.

video_tag adds a small button to show the corresponding video box and jump directly to the given position in the video.

Usage

```
video_box(id, src, title = NULL, type = "video/mp4")

video_tag(id, time = NULL, title = NULL)
```

Arguments

`id`	id of the video element
`src`	source(s) for the video,
`title`	title to be shown above the video
`type`	character specifying the video type(s)
`time`	optional integer specifying the time to jump to (in seconds)

Author(s)

richard.kunze

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {

library(shiny)
base_url <- "http://download.blender.org/peach/bigbuckbunny_movies/"
shinyApp(
  ui = fluidPage(
    video_box("lowRes",
      paste0(base_url, "BigBuckBunny_320x180.mp4"),
      "Low Resolution"
    ),
    video_box("highRes", type = "audio/ogg",
      paste0(base_url, "big_buck_bunny_720p_stereo.ogg")
    ),
```

```
    fluidRow(column(12,
      "Low resolution: ", video_tag("lowRes"), br(),
      "Higher resolution: ", video_tag("highRes", 300, "Bookmark at 5min")
    ))
  ),
  server = function(input, output) {}
)

}
```

Index