Package ‘reactable’

February 29, 2020

Type Package

Title Interactive Data Tables Based on 'React Table'

Version 0.1.0.1

Description Interactive data tables for R, based on the 'React Table'
JavaScript library. Provides an HTML widget that can be used in 'R Markdown'
documents and 'Shiny' applications, or viewed from an R console.

License MIT + file LICENSE

URL https://glin.github.io/reactable,
https://github.com/glin/reactable

BugReports https://github.com/glin/reactable/issues

Depends R (>= 3.1)

Imports digest, htmltools, htmlwidgets, jsonlite, reactR

Suggests covr, dplyr, rmarkdown, shiny, sparkline, testthat

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Greg Lin [aut, cre],
Tanner Linsley [ctb, cph] (React Table library)

Maintainer Greg Lin <glin@glin.io>

Repository CRAN

Date/Publication 2020-02-29 17:40:02 UTC

R topics documented:

colDef .................. .................................................. 2
colFormat .................. .................................................. 4
colGroup .................. .................................................. 5
reactable .................. .................................................. 6
reactable-shiny .................. .............................................. 9
Description
Column definitions

Usage

colDef(name = NULL, aggregate = NULL, sortable = NULL,
resizable = NULL, filterable = NULL, show = TRUE,
defaultSortOrder = NULL, sortNALast = FALSE, format = NULL,
cell = NULL, aggregated = NULL, header = NULL, footer = NULL,
details = NULL, html = FALSE, na = "", minWidth = NULL,
maxWidth = NULL, width = NULL, align = NULL, class = NULL,
style = NULL, headerClass = NULL, headerStyle = NULL,
footerClass = NULL, footerStyle = NULL)

Arguments

name  Column header name.
aggregate  Aggregate function. The name of a built-in aggregate function or a custom JS() aggregate function. Built-in aggregate functions are: "mean", "sum", "max", "min", "count", "unique", "frequency".
sortable  Enable sorting? Overrides the table option.
resizable  Enable column resizing? Overrides the table option.
filterable  Enable column filtering? Overrides the table option.
show  Show the column? Defaults to TRUE.
defaultSortOrder  Default sort order. Either "asc" for ascending order or "desc" for descending order. Overrides the table option.
sortNALast  Always sort missing values (NA or NaN) last?
format  Column formatting options. A colFormat() object to format all cells, or a named list of colFormat() objects to format standard cells ("cell") and aggregated cells ("aggregated") separately.

header  Custom header renderer. An R function that takes the header value and column name as arguments, or a JS() function that takes a column info object as an argument.
footer  Footer content or render function. Render functions can be an R function that takes two arguments, the column values and column name, or a JS() function that takes a column info object as an argument.

details  Additional content to display when expanding a row. An R function that takes a row index argument or a JS() function that takes a row info object as an argument. Cannot be used on a grouping column.

html  Render cells as HTML? HTML strings are escaped by default.

na  String to display for missing values (i.e. NA or NaN). By default, missing values are displayed as blank cells.

minWidth  Min width of the column in pixels.

maxWidth  Max width of the column in pixels. Overrides minWidth and maxWidth.

width  Fixed width of the column in pixels.

align  Column alignment. One of "left", "right", "center".

class  Additional CSS classes to apply to cells. Can also be an R function that takes the cell value, row index, and column name as arguments, or a JS() function that takes a row info object, column info object, and table state object as arguments. Note that R functions cannot apply classes to aggregated cells.

style  Inline styles to apply to cells. A named list or character string. Can also be an R function that takes the cell value and row index as arguments, or a JS() function that takes a row info object, column info object, and table state object as arguments. Note that R functions cannot apply styles to aggregated cells. If style is a named list, property names should be camelCased.

headerClass  Additional CSS classes to apply to the header.

headerStyle  Inline styles to apply to the header. A named list or character string. Note that if headerStyle is a named list, property names should be camelCased.

footerClass  Additional CSS classes to apply to the footer.

footerStyle  Inline styles to apply to the footer. A named list or character string. Note that if footerStyle is a named list, property names should be camelCased.

Value

A column definition object that can be used to customize columns in reactable().

Examples

reactable(
  iris,
  columns = list(
    Sepal.Length = colDef(name = "Sepal Length"),
    Sepal.Width = colDef(filterable = TRUE),
    Petal.Length = colDef(show = FALSE),
    Petal.Width = colDef(defaultSortOrder = "desc")
  )
)
Column formatting options

Description

Column formatting options

Usage

```
colFormat(prefix = NULL, suffix = NULL, digits = NULL, 
  separators = FALSE, percent = FALSE, currency = NULL, 
  datetime = FALSE, date = FALSE, time = FALSE, hour12 = NULL, 
  locales = NULL)
```

Arguments

- `prefix`: Prefix string.
- `suffix` : Suffix string.
- `digits` : Number of decimal digits to use for numbers.
- `separators` : Whether to use grouping separators for numbers, such as thousands separators or thousand/lakh/crore separators. The format is locale-dependent.
- `percent` : Format number as a percentage? The format is locale-dependent.
- `currency` : Currency format. An ISO 4217 currency code such as "USD" for the US dollar, "EUR" for the euro, or "CNY" for the Chinese RMB. The format is locale-dependent.
- `datetime` : Format as a locale-dependent date-time?
- `date` : Format as a locale-dependent date?
- `time` : Format as a locale-dependent time?
- `hour12` : Whether to use 12-hour time (TRUE) or 24-hour time (FALSE). The default time convention is locale-dependent.
- `locales` : Locales to use for number and date/time formatting. A character vector of BCP 47 language tags, such as "en-US" for English (United States), "hi" for Hindi, or "sv-SE" for Swedish (Sweden). Defaults to the locale of the browser.

Value

A column format object that can be used to customize data formatting in `colDef()`.

Examples

```
data <- data.frame(  
  price_USD = c(123456.56, 132, 5650.12),  
  price_INR = c(350, 23208.552, 1773156.4),  
  temp = c(22, NA, 31),  
  percent = c(0.9525556, 0.5, 0.112),
)```
```
date = as.Date(c("2019-01-02", "2019-03-15", "2019-09-22"))

reactable(data, columns = list(
  price_USD = colDef(format = colFormat(prefix = "$", separators = TRUE, digits = 2)),
  price_INR = colDef(format = colFormat(currency = "INR", separators = TRUE, locale = "hi-IN")),
  temp = colDef(format = colFormat(suffix = "°C")),
  percent = colDef(format = colFormat(percent = TRUE, digits = 1)),
  date = colDef(format = colFormat(date = TRUE, locale = "en-GB")))
))
```

---

**colGroup**

*Column group definitions*

### Description

Column group definitions

### Usage

```
colGroup(name = NULL, columns = NULL, header = NULL, html = FALSE,
         align = NULL, headerClass = NULL, headerStyle = NULL)
```

### Arguments

- **name**: Column group header name.
- **columns**: Character vector of column names in the group.
- **header**: Custom header renderer. An R function that takes the header value as an argument, or a JS() function that takes a column info object as an argument.
- **html**: Render header as HTML? HTML strings are escaped by default.
- **align**: Column group header alignment. One of "left", "right", "center".
- **headerClass**: Additional CSS classes to apply to the header.
- **headerStyle**: Inline styles to apply to the header. A named list or character string. Note that if headerStyle is a named list, property names should be camelCased.

### Value

A column group definition object that can be used to create column groups in reactable().
**Examples**

reactable(
    iris,
    columns = list(
        Sepal.Length = colDef(name = "Length"),
        Sepal.Width = colDef(name = "Width"),
        Petal.Length = colDef(name = "Length"),
        Petal.Width = colDef(name = "Width")
    ),
    columnGroups = list(
        colGroup(name = "Sepal", columns = c("Sepal.Length", "Sepal.Width")),
        colGroup(name = "Petal", columns = c("Petal.Length", "Petal.Width"))
    )
)

---

**reactable**

Create an interactive data table

**Description**

`reactable()` creates a data table from tabular data with sorting and pagination by default. The data table is an HTML widget that can be used in R Markdown documents and Shiny applications, or viewed from an R console.

**Usage**

```r
reactable(data, columns = NULL, columnGroups = NULL, rownames = NULL,
          groupBy = NULL, sortable = TRUE, resizable = FALSE,
          filterable = FALSE, searchable = FALSE, defaultColDef = NULL,
          defaultColGroup = NULL, defaultSortOrder = "asc",
          defaultSorted = NULL, pagination = TRUE, defaultPageSize = 10,
          showPageSizeOptions = FALSE, pageSizeOptions = c(10, 25, 50, 100),
          paginationType = "numbers", showPagination = NULL,
          showPageInfo = TRUE, minRows = 1, details = NULL,
          selection = NULL, selectionId = NULL, onClick = NULL,
          highlight = FALSE, outlined = FALSE, bordered = FALSE,
          borderless = FALSE, striped = FALSE, compact = FALSE,
          wrap = TRUE, showSortIcon = TRUE, showSortable = FALSE,
          class = NULL, style = NULL, rowClass = NULL, rowStyle = NULL,
          fullWidth = TRUE, width = "auto", height = "auto",
          elementId = NULL)
```

**Arguments**

- **data**
  A data frame or matrix.

- **columns**
  Named list of column definitions. See `colDef()`.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| columnGroups  | List of column group definitions. See `colGroup()`.
| rownames      | Show row names? Defaults to TRUE if the data has row names. To customize or group the row names column, use ".rownames" as the column name.
|groupBy        | Character vector of column names to group by.
| sortable      | Enable sorting? Defaults to TRUE.
| resizable     | Enable column resizing?
| filterable    | Enable column filtering?
| searchable    | Enable global table searching?
| defaultColDef | Default column definition used by every column. See `colDef()`.
| defaultColGroup| Default column group definition used by every column group. See `colGroup()`.
| defaultSortOrder| Default sort order. Either "asc" for ascending order or "desc" for descending order. Defaults to "asc".
| defaultSorted | Optional vector of column names to sort by default. Or to customize sort order, a named list with values of "asc" or "desc".
| pagination    | Enable pagination? Defaults to TRUE.
| defaultPageSize| Default page size for the table. Defaults to 10.
| showPageSizeOptions| Show page size options?
| pageSizeOptions| Page size options for the table. Defaults to 10, 25, 50, 100.
| paginationType| Pagination control to use. Either "numbers" for page number buttons (the default), "jump" for a page jump, or "simple" to show 'Previous' and 'Next' buttons only.
| showPagination| Show pagination? Defaults to TRUE if the table has more than one page.
| showPageInfo  | Show page info? Defaults to TRUE.
| minRows       | Minimum number of rows to show per page. Defaults to 1.
| details       | Additional content to display when expanding a row. An R function that takes a row index argument or a `JS()` function that takes a row info object as an argument. Can also be a `colDef()` to customize the details expander column.
| selection     | Enable row selection? Either "multiple" or "single" for multiple or single row selection.
| selectionId   | Shiny input ID for the selected rows. The selected rows are represented as a vector of row indices, or NULL if no rows are selected.
| onClick       | Action to take when clicking a cell. Either "expand" to expand the row, "select" to select the row, or a `JS()` function that takes a row info object, column info object, and table state object as arguments.
| highlight     | Highlight table rows on hover?
outlined: Add borders around the table?
bordered: Add borders around the table and every cell?
borderless: Remove inner borders from table?
striped: Add zebra-stripping to table rows?
compact: Make tables more compact?
wrap: Enable text wrapping? If TRUE (the default), long text will be wrapped to multiple lines. If FALSE, text will be truncated to fit on one line.
showSortIcon: Show a sort icon when sorting columns?
showSortable: Show an indicator on sortable columns?
class: Additional CSS classes to apply to the table.
style: Inline styles to apply to the table. A named list or character string.
    Note that if style is a named list, property names should be camelCased.
rowClass: Additional CSS classes to apply to table rows. A character string, a JS() function that takes a row info object and table state object as arguments, or an R function that takes a row index argument.
rowStyle: Inline styles to apply to table rows. A named list, character string, JS() function that takes a row info object and table state object as arguments, or an R function that takes a row index argument.
    Note that if rowStyle is a named list, property names should be camelCased. If rowStyle is a JS() function, it should return a JavaScript object with camelCased property names.
fullWidth: Stretch the table to fill the full width of its container? Defaults to TRUE.
width: Width in pixels. Defaults to "auto" for automatic sizing.
height: Height in pixels. Defaults to "auto" for automatic sizing.
elementId: Element ID for the widget.

Value

A reactable HTML widget that can be used in R Markdown documents and Shiny applications, or viewed from an R console.

Note

See the online documentation for additional details and examples.

See Also

renderReactable() and reactableOutput() for using reactable in Shiny applications or interactive R Markdown documents.
Examples

# Basic usage
reactable(iris)

# Grouping and aggregation
reactable(iris, groupBy = "Species", columns = list(
  Sepal.Length = colDef(aggregate = "count"),
  Sepal.Width = colDef(aggregate = "mean"),
  Petal.Length = colDef(aggregate = "sum"),
  Petal.Width = colDef(aggregate = "max"))
)

# Row details
reactable(iris, details = function(index) {
  htmltools::div(
    "Details for row: ", index,
    htmltools::tags$pre(paste(capture.output(iris[index, ]), collapse = "\n"))
  )
})

# Conditional styling
reactable(sleep, columns = list(
  extra = colDef(style = function(value) {
    if (value > 0) {
      color <- "green"
    } else if (value < 0) {
      color <- "red"
    } else {
      color <- "#777"
    }
    list(color = color, fontWeight = "bold")
  }))


---

reactable-shiny  Shiny bindings for reactable

Description

Output and render functions for using reactable within Shiny applications and interactive R Mark-
down documents.

Usage

reactableOutput(outputId, width = "auto", height = "auto",
inline = FALSE)

renderReactable(expr, env = parent.frame(), quoted = FALSE)
Arguments

- **outputId**: Output variable to read from.
- **width, height**: A valid CSS unit (like "100\%", "400px", "auto") or a number, which will be coerced to a string and have "px" appended.
- **inline**: Use an inline element for the table's container?
- **expr**: An expression that generates a `reactable` widget.
- **env**: The environment in which to evaluate `expr`.
- **quoted**: Is `expr` a quoted expression (with `quote()`)? This is useful if you want to save an expression in a variable.

Value

- `reactableOutput()` returns a reactable output element that can be included in a Shiny UI.
- `renderReactable()` returns a reactable render function that can be assigned to a Shiny output slot.

Note

See the online demo for additional examples of using reactable in Shiny.

Examples

```r
# Run in an interactive R session
if (interactive()) {

  library(shiny)
  library(reactable)

  ui <- fluidPage(
    titlePanel("reactable example"),
    reactableOutput("table")
  )

  server <- function(input, output, session) {
    output$table <- renderReactable({
      reactable(iris)
    })
  }

  shinyApp(ui, server)
}
```
Index

colDef, 2
colDef(), 6, 7
colFormat, 4
colFormat(), 2
colGroup, 5
colGroup(), 7

JS(), 2, 3, 5, 7, 8

NA, 2, 3
NaN, 2, 3

quote(), 10

reactable, 6, 10
reactable-shiny, 9
reactableOutput (reactable-shiny), 9
reactableOutput(), 8
renderReactable (reactable-shiny), 9
renderReactable(), 8