Package 'reactable'

May 28, 2020

Type Package

Title Interactive Data Tables Based on 'React Table'

Version 0.2.0

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, crosstalk, dplyr, leaflet, rmarkdown, shiny, sparkline, testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Greg Lin [aut, cre], Tanner Linsley [ctb, cph] (React Table library), Emotion team and other contributors [ctb, cph] (Emotion library)

Maintainer Greg Lin <glin@glin.io>

Repository CRAN

Date/Publication 2020-05-28 05:50:03 UTC

R topics documented:

colDef 2
colFormat 4

colGroup 6
getReactableState 7
reactable 8
reactable-shiny 13
reactableLang 14
reactableTheme 17
updateReactable 20

Index **23**

`colDef` *Column definitions*

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", "median", "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.
cell	Custom cell renderer. An R function that takes the cell value, row index, and column name as arguments, or a JS() function that takes a cell info object as an argument.
aggregated	Custom aggregated cell renderer. A JS() function that takes a cell info object as an argument.
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 content as HTML? Raw 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.
width	Fixed width of the column in pixels. Overrides minWidth and maxWidth.
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")
  )
)
```

---

colFormat *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 = " \u00b0C")),
  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 content as HTML? Raw 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"))
  )
)
```

`getReactableState` *Get the state of a reactable instance*

Description

`getReactableState()` gets the state of a reactable instance within a Shiny application.

Usage

```
getReactableState(outputId, name = NULL, session = NULL)
```

Arguments

`outputId`	The Shiny output ID of the `reactable` instance.
`name`	Name of a state value to get. One of `"page"`, `"pageSize"`, `"pages"`, or `"selected"`. If unspecified, all values will be returned in a named list.
`session`	The Shiny session object. Defaults to the current Shiny session.

Value

If `name` is specified, one of the following values:

- `page`: the current page
- `pageSize`: the page size
- `pages`: the number of pages
- `selected`: the selected rows - a numeric vector of row indices, or `NULL` if no rows are selected

If `name` is unspecified, `getReactableState()` returns a named list containing all values.

If the table has not been rendered yet, `getReactableState()` returns `NULL`.

Examples

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

library(shiny)
library(reactable)

ui <- fluidPage(
  actionButton("prev_page_btn", "Previous page"),
  actionButton("next_page_btn", "Next page"),
  reactableOutput("table"),
```

```
  verbatimTextOutput("table_state")
)

server <- function(input, output) {
  output$table <- renderReactable({
    reactable(
      iris,
      showPageSizeOptions = TRUE,
      selection = "multiple"
    )
  })

  output$table_state <- renderPrint({
    state <- req(getReactableState("table"))
    print(state)
  })

  observeEvent(input$prev_page_btn, {
    # Change to the previous page
    page <- getReactableState("table", "page")
    if (page > 1) {
      updateReactable("table", page = page - 1)
    }
  })

  observeEvent(input$next_page_btn, {
    # Change to the next page
    state <- getReactableState("table")
    if (state$page < state$pages) {
      updateReactable("table", page = state$page + 1)
    }
  })
}

shinyApp(ui, server)
}
```

`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

```
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,
  defaultExpanded = FALSE,
  selection = NULL,
  selectionId = NULL,
  defaultSelected = 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",
  theme = getOption("reactable.theme"),
  language = getOption("reactable.language"),
  elementId = NULL
)
```

Arguments

data	A data frame or matrix. Can also be a crosstalk::SharedData object that wraps a data frame.
columns	Named list of column definitions. See colDef().
columnGroups	List of column group definitions. See colGroup().
rownames	Show row names? Defaults to TRUE if the data has row names. To customize 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.
defaultExpanded	Expand all rows by default?
selection	Enable row selection? Either "multiple" or "single" for multiple or single row selection. To get the selected rows in Shiny, use getReactableState(). To customize the selection column, use ".selection" as the column name.

selectionId	Shiny input ID for the selected rows. The selected rows are given as a numeric vector of row indices, or NULL if no rows are selected. **NOTE:** selectionId will be deprecated in a future release. Use getReactableState() to get the selected rows in Shiny instead.
defaultSelected	A numeric vector of default selected row indices.
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-striping 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.
theme	Theme options for the table, specified by reactableTheme(). Defaults to the global reactable.theme option. Can also be a function that returns a reactableTheme() or NULL.
language	Language options for the table, specified by reactableLang(). Defaults to the global reactable.language option.
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 Markdown 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.

See Also

`updateReactable()` for updating a reactable instance in Shiny.

`getReactableState()` for getting the state of a reactable instance in Shiny.

Examples

```
# 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)
}
```

---

reactableLang *Language options*

Description

Use `reactableLang()` to customize the language strings in a table. Language strings include both visible text and accessible labels that can be read by assistive technology, such as screen readers.

To set the default language strings for all tables, use the global `reactable.language` option.

Usage

```
reactableLang(
  sortLabel = "Sort {name}",
  filterPlaceholder = "",
  filterLabel = "Filter {name}",
  searchPlaceholder = "Search",
  searchLabel = "Search",
  noData = "No rows found",
  pageNext = "Next",
  pagePrevious = "Previous",
  pageNumbers = "{page} of {pages}",
  pageInfo = "{rowStart}{rowEnd} of {rows} rows",
  pageSizeOptions = "Show {rows}",
  pageNextLabel = "Next page",
  pagePreviousLabel = "Previous page",
  pageNumberLabel = "Page {page}",
```

```
  pageJumpLabel = "Go to page",
  pageSizeOptionsLabel = "Rows per page",
  defaultGroupHeader = "Grouped",
  detailsExpandLabel = "Expand details",
  detailsCollapseLabel = "Collapse details",
  selectAllRowsLabel = "Select all rows",
  deselectAllRowsLabel = "Deselect all rows",
  selectAllSubRowsLabel = "Select all rows in group",
  deselectAllSubRowsLabel = "Deselect all rows in group",
  selectRowLabel = "Select row",
  deselectRowLabel = "Deselect row"
)
```

Arguments

`sortLabel`	Accessible label for column sort buttons. Takes a `{name}` parameter for the column name.
`filterPlaceholder`	Placeholder for column filter inputs.
`filterLabel`	Accessible label for column filter inputs. Takes a `{name}` parameter for the column name.
`searchPlaceholder`	Placeholder for the table search input.
`searchLabel`	Accessible label for the table search input.
`noData`	Placeholder text when the table has no data.
`pageNext`	Text for the next page button.
`pagePrevious`	Text for the previous page button.
`pageNumbers`	Text for the page numbers info. Only used with the `"jump"` and `"simple"` pagination types. Takes the following parameters: • `{page}` for the current page • `{pages}` for the total number of pages
`pageInfo`	Text for the page info. Takes the following parameters: • `{rowStart}` for the starting row of the page • `{rowEnd}` for the ending row of the page • `{rows}` for the total number of rows
`pageSizeOptions`	Text for the page size options input. Takes a `{rows}` parameter for the page size options input.
`pageNextLabel`	Accessible label for the next page button.
`pagePreviousLabel`	Accessible label for the previous page button.
`pageNumberLabel`	Accessible label for the page number buttons. Only used with the the `"numbers"` pagination type. Takes a `{page}` parameter for the page number.

`pageJumpLabel` Accessible label for the page jump input. Only used with the `"jump"` pagination type.

`pageSizeOptionsLabel`
Accessible label for the page size options input.

`defaultGroupHeader`
Header for default column groups. Only used for `groupBy` columns.

`detailsExpandLabel`
Accessible label for the row details expand button.

`detailsCollapseLabel`
Accessible label for the row details collapse button.

`selectAllRowsLabel`
Accessible label for the select all rows checkbox.

`deselectAllRowsLabel`
Accessible label for the deselect all rows checkbox.

`selectAllSubRowsLabel`
Accessible label for the select all sub rows checkbox.

`deselectAllSubRowsLabel`
Accessible label for the deselect all sub rows checkbox.

`selectRowLabel` Accessible label for the select row checkbox

`deselectRowLabel`
Accessible label for the deselect row checkbox.

Value

A language options object that can be used to customize the language strings in `reactable()`.

Examples

```
reactable(
  iris[1:30, ],
  searchable = TRUE,
  paginationType = "simple",
  language = reactableLang(
    searchPlaceholder = "Search...",
    noData = "No entries found",
    pageInfo = "{rowStart}\u2013{rowEnd} of {rows} entries",
    pagePrevious = "\u276e",
    pageNext = "\u276f",

    # Accessible labels for assistive technology, such as screen readers
    pagePreviousLabel = "Previous page",
    pageNextLabel = "Next page"
  )
)

# Set the default language for all tables
options(reactable.language = reactableLang(
  searchPlaceholder = "Search...",
  noData = "No entries found",
```

```
  pageInfo = "{rowStart} to {rowEnd} of {rows} entries"
))

reactable(iris[1:30, ], searchable = TRUE)
```

`reactableTheme` *Theme options*

Description

Use `reactableTheme()` to customize the default styling of a table. You can set theme variables to change the default styles, or add custom CSS to specific elements of the table.

The `color` variables are specified as character strings of CSS color values. The `width` and `padding` variables are specified as either character strings of CSS width and padding values, or numeric pixel values. The `style` arguments take custom CSS as named lists of camelCased properties.

To set the default theme for all tables, use the global `reactable.theme` option.

Usage

```
reactableTheme(
  color = NULL,
  backgroundColor = NULL,
  borderColor = NULL,
  borderWidth = NULL,
  stripedColor = NULL,
  highlightColor = NULL,
  cellPadding = NULL,
  style = NULL,
  tableStyle = NULL,
  headerStyle = NULL,
  groupHeaderStyle = NULL,
  tableBodyStyle = NULL,
  rowGroupStyle = NULL,
  rowStyle = NULL,
  rowStripedStyle = NULL,
  rowHighlightStyle = NULL,
  rowSelectedStyle = NULL,
  cellStyle = NULL,
  footerStyle = NULL,
  inputStyle = NULL,
  filterInputStyle = NULL,
  searchInputStyle = NULL,
  selectStyle = NULL,
  paginationStyle = NULL,
  pageButtonStyle = NULL,
  pageButtonHoverStyle = NULL,
```

```
  pageButtonActiveStyle = NULL,
  pageButtonCurrentStyle = NULL
)
```

Arguments

`color`	Default text color.
`backgroundColor`	Default background color.
`borderColor`	Default border color.
`borderWidth`	Default border width.
`stripedColor`	Default row stripe color.
`highlightColor`	Default row highlight color.
`cellPadding`	Default cell padding.
`style`	Additional CSS for the table.
`tableStyle`	Additional CSS for the table element (excludes the pagination bar and search input).
`headerStyle`	Additional CSS for header cells.
`groupHeaderStyle`	Additional CSS for group header cells.
`tableBodyStyle`	Additional CSS for the table body element.
`rowGroupStyle`	Additional CSS for row groups.
`rowStyle`	Additional CSS for rows.
`rowStripedStyle`	Additional CSS for striped rows.
`rowHighlightStyle`	Additional CSS for highlighted rows.
`rowSelectedStyle`	Additional CSS for selected rows.
`cellStyle`	Additional CSS for cells.
`footerStyle`	Additional CSS for footer cells.
`inputStyle`	Additional CSS for inputs.
`filterInputStyle`	Additional CSS for filter inputs.
`searchInputStyle`	Additional CSS for the search input.
`selectStyle`	Additional CSS for table select controls.
`paginationStyle`	Additional CSS for the pagination bar.
`pageButtonStyle`, `pageButtonHoverStyle`, `pageButtonActiveStyle`, `pageButtonCurrentStyle`	Additional CSS for page buttons, page buttons with hover or active states, and the current page button.

Details

You can use nested CSS selectors in `style` arguments to target the current element, using & as the selector, or other child elements (just like in Sass). This is useful for adding pseudo-classes like &:hover, or adding styles in a certain context like .outer-container &.

Value

A theme options object that can be used to customize the default styling in `reactable()`.

Examples

```
reactable(
  iris[1:30, ],
  searchable = TRUE,
  striped = TRUE,
  highlight = TRUE,
  bordered = TRUE,
  theme = reactableTheme(
    borderColor = "#dfe2e5",
    stripedColor = "#f6f8fa",
    highlightColor = "#f0f5f9",
    cellPadding = "8px 12px",
    style = list(
    fontFamily = "-apple-system, BlinkMacSystemFont, Segoe UI, Helvetica, Arial, sans-serif"
    ),
    searchInputStyle = list(width = "100%")
  )
)

# Set the default theme for all tables
options(reactable.theme = reactableTheme(
  color = "hsl(233, 9%, 87%)",
  backgroundColor = "hsl(233, 9%, 19%)",
  borderColor = "hsl(233, 9%, 22%)",
  stripedColor = "hsl(233, 12%, 22%)",
  highlightColor = "hsl(233, 12%, 24%)",
  inputStyle = list(backgroundColor = "hsl(233, 9%, 25%)"),
  selectStyle = list(backgroundColor = "hsl(233, 9%, 25%)"),
  pageButtonHoverStyle = list(backgroundColor = "hsl(233, 9%, 25%)"),
  pageButtonActiveStyle = list(backgroundColor = "hsl(233, 9%, 28%)")
))

reactable(
  iris[1:30, ],
  filterable = TRUE,
  showPageSizeOptions = TRUE,
  striped = TRUE,
  highlight = TRUE,
  details = function(index) paste("Details for row", index)
)

# Use nested selectors to highlight headers when sorting
```

```
reactable(
  iris[1:30, ],
  columns = list(Sepal.Length = colDef(sortable = FALSE)),
  showSortable = TRUE,
  theme = reactableTheme(
    headerStyle = list(
      "&:hover[aria-sort]" = list(background = "hsl(0, 0%, 96%)"),
    "&[aria-sort='ascending'], &[aria-sort='descending']" = list(background = "hsl(0, 0%, 96%)"),
      borderColor = "#555"
    )
  )
)
```

`updateReactable` *Update a reactable instance*

Description

`updateReactable()` updates a reactable instance within a Shiny application.

Usage

```
updateReactable(
  outputId,
  selected = NULL,
  expanded = NULL,
  page = NULL,
  session = NULL
)
```

Arguments

`outputId`	The Shiny output ID of the `reactable` instance.
`selected`	Selected rows. Either a numeric vector of row indices, or `NA` to deselect all rows.
`expanded`	Expanded rows. Either `TRUE` to expand all rows, or `FALSE` to collapse all rows.
`page`	The current page. A single, positive integer.
`session`	The Shiny session object. Defaults to the current Shiny session.

Value

None

Examples

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

library(shiny)
library(reactable)

ui <- fluidPage(
  actionButton("select_btn", "Select rows"),
  actionButton("clear_btn", "Clear selection"),
  actionButton("expand_btn", "Expand rows"),
  actionButton("collapse_btn", "Collapse rows"),
  actionButton("page_btn", "Change page"),
  reactableOutput("table")
)

server <- function(input, output) {
  output$table <- renderReactable({
    reactable(
      iris,
      selection = "multiple",
      details = function(index) paste("Details for row:", index)
    )
  })

  observeEvent(input$select_btn, {
    # Select rows
    updateReactable("table", selected = c(1, 3, 5))
  })

  observeEvent(input$clear_btn, {
    # Clear row selection
    updateReactable("table", selected = NA)
  })

  observeEvent(input$expand_btn, {
    # Expand all rows
    updateReactable("table", expanded = TRUE)
  })

  observeEvent(input$collapse_btn, {
    # Collapse all rows
    updateReactable("table", expanded = FALSE)
  })

  observeEvent(input$page_btn, {
    # Change current page
    updateReactable("table", page = 3)
  })
}

shinyApp(ui, server)
```

```
}
```

Index

colDef, 2
colDef(), *10*
colFormat, 4
colFormat(), *3*
colGroup, 6
colGroup(), *10*
crosstalk::SharedData, *10*

getReactableState, 7
getReactableState(), *10, 11*, *13*

JS(), *3*, *4*, *6*, *10, 11*

NA, *3*
NaN, *3*

quote(), *13*

reactable, 8, *13*
reactable-shiny, 13
reactableLang, 14
reactableLang(), *11*
reactableOutput *(*reactable-shiny*)*, 13
reactableOutput(), *12*
reactableTheme, 17
reactableTheme(), *11*
renderReactable *(*reactable-shiny*)*, 13
renderReactable(), *12*

updateReactable, 20
updateReactable(), *13*