

# Package: muiDataGrid (via r-universe)

June 21, 2026

**Type** Package

**Title** 'MUI X Data Grid' for 'shiny' Apps and 'Quarto'

**Version** 0.1.2

**Maintainer** Felix Luginbuhl <felix.luginbuhl@protonmail.ch>

**Description** Provides access to 'MUI X Data Grid', a fast and extensible React data table and React data grid, with filtering, sorting, pagination, and more. Bundles the MIT-licensed community edition of the '@mui/x-data-grid' JavaScript library (the commercial 'Pro' and 'Premium' tiers are not included).

**License** MIT + file LICENSE

**Encoding** UTF-8

**Language** en-US

**Depends** R (>= 3.4)

**Imports** htmltools, shiny, shiny.react (>= 0.4.0), muiMaterial (>= 0.2.0), utils

**Suggests** knitr, rmarkdown, dplyr, testthat (>= 3.0.0)

**RoxygenNote** 7.3.3

**URL** <https://felixluginbuhl.com/muiDataGrid/>

**BugReports** <https://github.com/lgnbhl/muiDataGrid/issues>

**Config/testthat/edition** 3

**Config/pak/sysreqs** cmake make libicu-dev libuv1-dev zlib1g-dev

**Repository** <https://lgnbhl.r-universe.dev>

**Date/Publication** 2026-06-14 10:41:49 UTC

**RemoteUrl** <https://github.com/lgnbhl/muidatagrid>

**RemoteRef** HEAD

**RemoteSha** 7c175d67eb7e932f7f7ac7ec875fe7a32dfadbe0

## Contents

DataGrid	2
DataGridServer	3
muiDataGridDependency	5
print.muiDataGrid	6
processGridParams	6
<b>Index</b>	<b>8</b>

---

DataGrid	<i>DataGrid</i>
----------	-----------------

---

### Description

DataGrid

### Usage

```
DataGrid(rows = NULL, columns = NULL, ...)
```

### Arguments

rows	A data.frame of rows. An id column of 1-based row numbers is added automatically if not already present.
columns	Column definitions (list of lists, or a data.frame with one row per column). If NULL, auto-generated from names(rows). Any column without an explicit type has one inferred from the matching rows column: numeric columns get MUI's number type and logical columns its boolean type; all other classes (including Date, POSIXct, and factors) default to string. A type you set yourself is always kept. To get MUI's date-picker filtering, set type = "date" together with a valueGetter that returns a JS Date explicitly.
...	Additional props passed directly to the MUI DataGrid component.

### Value

A shiny.react element (also classed muiDataGrid) that renders the MUI X Data Grid. Use it directly in Shiny UI, inside renderReact(), or in a Quarto/R Markdown document.

### Examples

```
# Minimal: column definitions are auto-generated from the data frame.
DataGrid(rows = head(iris))

# Custom columns plus an initial page size of 5 rows.
DataGrid(
  rows = head(mtcars),
  columns = list(
    list(field = "mpg", headerName = "MPG"),
```

```

    list(field = "cyl", headerName = "Cylinders")
  ),
  initialState = list(
    pagination = list(paginationModel = list(pageSize = 5))
  )
)

```

---

DataGridServer

*Server-Side DataGrid*


---

### Description

A DataGrid component with server-side pagination, sorting, and filtering. The component sends pagination, sort, and filter state to R via a Shiny input.

### Usage

```

DataGridServer(
  inputId,
  rows = NULL,
  columns = NULL,
  rowCount = NULL,
  loading = NULL,
  initialPageSize = NULL,
  pageSizeOptions = NULL,
  filterDebounce = NULL,
  ...
)

```

### Arguments

inputId	Character. The Shiny input ID. When pagination, sorting, or filtering changes, the new state is available as <code>input\$&lt;inputId&gt;</code> in the server. The value is a list with elements <code>pagination_model</code> (list with page and <code>pageSize</code> ), <code>sort_model</code> (list of sort items), and <code>filter_model</code> (list with items).
rows	A data.frame. Pass the <b>full</b> dataset (like <code>DataGrid()</code> ) and let <code>DataGridServer()</code> handle pagination automatically, or pass a pre-sliced page together with an explicit <code>rowCount</code> for manual control.
columns	Column definitions (list of lists). If <code>NULL</code> , auto-generated from <code>names(rows)</code> , with each column's type inferred from its R class (see <a href="#">DataGrid</a> ).
rowCount	Integer. When provided, <code>rows</code> is assumed to be already paginated and <code>rowCount</code> is used as the total row count (manual mode). When <code>NULL</code> (default), pagination is handled automatically from the full rows dataset.
loading	Logical. Whether to show the loading indicator. If <code>NULL</code> , MUI defaults to <code>FALSE</code> .

<code>initialPageSize</code>	Integer. Convenience for setting the initial page size. Builds MUI's <code>initialState</code> prop. If NULL, MUI defaults to 100. Also sets the page size for the first automatic render before the grid has sent state. Must be included in <code>pageSizeOptions</code> .
<code>pageSizeOptions</code>	Integer vector. Available page size options. If NULL, MUI defaults to <code>c(25, 50, 100)</code> .
<code>filterDebounce</code>	Integer. Milliseconds to debounce filter input before sending state to R. If NULL, defaults to 300 ms.
<code>...</code>	Additional props passed directly to the MUI DataGrid component. Note: <code>paginationMode</code> , <code>sortingMode</code> , <code>filterMode</code> , and <code>pagination</code> are set automatically and should not be overridden.

### Details

Pass the full dataset via `rows` — just like `DataGrid()` — and `DataGridServer()` handles pagination, sorting, and filtering automatically. For manual control (e.g. database queries), supply pre-sliced rows together with an explicit `rowCount`.

### Value

A shiny.react element.

### Lifecycle

**Experimental.** `DataGridServer()` (together with [processGridParams](#)) is *specific to this R package* and has no equivalent in MUI X Data Grid: MUI ships only the building blocks (`paginationMode = "server"` plus callbacks) and leaves the data layer to you. This wrapper supplies that layer in R, and in doing so encodes a number of opinionated decisions that may change in future releases. Pin the package version if you rely on the current behavior. Decisions worth knowing about:

- Mode is selected by the presence of `rowCount`: supply it to pass a pre-sliced page (manual mode); omit it to let the full rows be paginated automatically.
- Changing the sort or any filter resets the grid to the first page.
- Unrecognized filter operators pass all rows through with a warning (see [processGridParams](#)).
- When `rows` has no `id` column, `ids` are generated positionally and are *not* stable across sort/filter changes. Supply a stable, unique `id` column if you use row selection.
- `initialPageSize`, `initialState`, and any initial sort or filter seed the grid *only on the first render*. The React component reads them once when it mounts, so changing them reactively afterwards (e.g. from a `selectInput`) has no effect on the already-mounted grid. To change page size after mount, drive it from the grid's own controls rather than re-rendering with a new `initialPageSize`.

### Examples

```
## Not run:
# Simple usage: pass the full dataset, pagination is handled automatically
output$grid <- renderReact({
```

```
DataGridServer("grid_params",
  rows = my_data,
  initialPageSize = 10L,
  pageSizeOptions = c(10L, 25L, 50L)
)
})

# Manual usage: handle pagination yourself (e.g. database queries)
output$grid <- renderReact({
  result <- processGridParams(my_data, input$grid_params, pageSize = 10L)
  DataGridServer("grid_params",
    rows = result$rows,
    rowCount = result$rowCount,
    initialPageSize = 10L,
    pageSizeOptions = c(10L, 25L, 50L)
  )
})

## End(Not run)
```

---

`muiDataGridDependency` *Mui X Data Grid UI JS dependency*

---

## Description

Mui X Data Grid UI JS dependency

## Usage

```
muiDataGridDependency()
```

## Value

A list of HTML dependency objects (`htmltools::htmlDependency`) that load the bundled MUI X Data Grid JavaScript and its shared MUI Material runtime. Attach it to custom HTML when building a grid by hand.

## Examples

```
# Inspect the dependencies attached to every DataGrid() element.
muiDataGridDependency()
```

print.muiDataGrid      *Print muiDataGrid components*

---

### Description

When called interactively, renders the component in the IDE viewer panel. Otherwise, falls back to standard shiny.tag printing (raw HTML text).

### Usage

```
## S3 method for class 'muiDataGrid'  
print(x, browse = interactive(), ...)
```

### Arguments

x	A muiDataGrid object (also inherits shiny.tag).
browse	Whether to render in viewer. Defaults to TRUE in interactive sessions.
...	Additional arguments passed to print.

### Value

Invisibly returns x.

### Examples

```
grid <- DataGrid(rows = head(iris))  
# browse = FALSE prints the underlying HTML instead of opening the viewer.  
print(grid, browse = FALSE)
```

---

processGridParams      *Process server-side grid parameters*

---

### Description

Applies pagination, sorting, and filtering from DataGridServer parameters to a data.frame. Use inside renderReact() with input\$<inputId> to get the current page of data.

### Usage

```
processGridParams(data, params, pageSize = 100L)
```

## Arguments

data	A data.frame with all rows.
params	The grid params list from input\$<inputId>, or NULL on initial render.
pageSize	Page size to use when params is NULL (i.e. the first render before the grid has sent state). Should match the initialPageSize passed to DataGridServer, or MUI's default of 100 if none was specified.

## Value

A list with rows (data.frame for the current page) and rowCount (integer, total matching rows).

## Lifecycle

**Experimental.** This helper is *specific to this R package* and has no equivalent in MUI X Data Grid, which leaves the server-side data layer entirely to the developer. It reimplements MUI's server-mode filtering and sorting semantics in R, so its behavior may change between releases and can differ in edge cases from MUI's own client-side filtering. Decisions worth knowing about: string filters are case-insensitive (except `is`, which is case-sensitive), missing values always sort last, and unrecognized filter operators pass all rows through *with a warning*.

## Examples

```
df <- data.frame(name = paste("Row", 1:50), value = 1:50)

# Initial render (no params yet)
processGridParams(df, params = NULL, pageSize = 10)

# Page 2, sorted descending, filtered
params <- list(
  pagination_model = list(page = 1, pageSize = 10),
  sort_model = list(list(field = "value", sort = "desc")),
  filter_model = list(items = list(
    list(field = "value", operator = ">", value = "10")
  ))
)
processGridParams(df, params)
```

# Index

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

DataGridServer, [3](#)

muiDataGridDependency, [5](#)

print.muiDataGrid, [6](#)

processGridParams, [4](#), [6](#)