# Package: blockr.ui (via r-universe)

May 26, 2026

**Title** A 'shiny' powered user interface for 'blockr.core'

**Version** 1.0.1

**Description** Provides a web-based point and click interface for creating data pipelines and visualizations with 'blockr.core'.

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Imports** shiny (>= 1.5.0), bslib, scoutbaR, shinyWidgets, blockr.core (>= 0.1.1), shinyjs, jsonlite, htmltools, dockViewR (>= 0.3.0), g6R (>= 0.2.0), ellmer (>= 0.3.0), grDevices, rlang, promises, shinychat (>= 0.2.0.9000), cli

**Remotes** BristolMyersSquibb/blockr.core, cynkra/g6R, cynkra/scoutbaR, cynkra/dockViewR, posit-dev/shinychat/pkg-r

**Suggests** shinytest2, testthat (>= 3.0.0), cyclocomp, roxy.shinylive, chromote, quarto, knitr, withr, R6, coro

**Config/testthat/edition** 3

**URL** https://github.com/BristolMyersSquibb/blockr.ui/, https://bristolmyerssquibb.github.io/blockr.ui/

**BugReports** https://github.com/BristolMyersSquibb/blockr.ui/issues

**Depends** R (>= 2.10)

**Config/Needs/website** rmarkdown

**VignetteBuilder** quarto

**Collate** 'apps.R' 'block.R' 'blockr.ui-dependencies.R' 'busy-load-dependencies.R' 'context-menu.R' 'dag-board.R' 'main.R' 'module-ai-chat.R' 'module-dashboard.R' 'options.R' 'pkg.R' 'plugin-add_rm_blocks.R' 'plugin-block.R' 'plugin-code.R' 'plugin-links.R' 'plugin-serialize.R' 'plugin-stacks.R' 'stacks.R' 'utils-ai-tools.R' 'utils-board.R' 'utils-dashboard.R' 'utils-links.R' 'utils-module.R' 'utils-serialize.R' 'utils-ui.R' 'utils.R'

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

**Repository** https://cynkra.r-universe.dev

**Date/Publication** 2025-10-27 16:59:50 UTC

**RemoteUrl** https://github.com/cynkra/blockr.ui

**RemoteRef** HEAD

**RemoteSha** df797753389c7c4febe50bc5c0d220dcec3077c1

# Contents



---

`add_blockr.ui_deps` *blockr.ui dependencies utils*

---

## Description

This function attaches blockr.ui dependencies to the given tag

### Usage

```
add_blockr.ui_deps(tag)
```

### Arguments

| | |
|---|---|
| tag | Element to attach the dependencies. |

---

## add_busy_load_deps *busy-load dependencies utils*

---

### Description

This function attaches busy-load. dependencies to the given tag

### Usage

```
add_busy_load_deps(tag)
```

### Arguments

| | |
|---|---|
| tag | Element to attach the dependencies. |

---

## add_rm_block_server *Add/remove block module*

---

### Description

Customizable logic for adding/removing blocks to the board.

### Usage

```
add_rm_block_server(id, board, update, parent, ...)

add_rm_block_ui(id, board)
```

### Arguments

| | |
|---|---|
| id | Namespace ID. |
| board | The initial board object |
| update | Reactive value object to initiate board updates. |
| parent | App state |
| ... | Extra arguments passed from parent scope. Useful to communicate between plugins and surface information at the top level (for testing ...). |

### Value

NULL.

## add_rm_stack_server *Add/remove block stacks module*

### Description

Customizable logic for adding/removing stacks grouping blocks together on the board.

### Usage

```
add_rm_stack_server(id, board, update, parent, ...)

add_rm_stack_ui(id, board)
```

### Arguments

| | |
|---|---|
| id | Namespace ID. |
| board | The initial board object |
| update | Reactive value object to initiate board updates. |
| parent | App state |
| ... | Extra arguments passed from parent scope. |

### Value

A reactive value that evaluates to NULL or a list with components add and rm, where add is either NULL or a stacks object and rm is either NULL or a character vector of link IDs.

## block_subtitle_id *Block subtitle id method*

### Description

Block subtitle id method

### Usage

```
block_subtitle_id(x, id)
```

### Arguments

| | |
|---|---|
| x | Board. |
| id | Block id. |

## block_ui.dag_board *Block custom UI*

### Description

Block custom UI

### Usage

```
## S3 method for class 'dag_board'
block_ui(id, x, block = NULL, edit_ui = NULL, ...)

## S3 method for class 'dag_board'
insert_block_ui(id, x, blocks = NULL, ...)

## S3 method for class 'dag_board'
remove_block_ui(id, x, blocks = NULL, ...)
```

### Arguments

| | |
|---|---|
| id | Block module id. |
| x | Board object. |
| block | Block to generate the UI for. |
| edit_ui | Block edit plugin. |
| ... | Generic consistency. |
| blocks | Blocks to insert or remove. |

## can_connect *Check whether the node can receive connection*

### Description

Check whether the node can receive connection

### Usage

```
can_connect(x, target, rv)
```

### Arguments

| | |
|---|---|
| x | Block object. |
| target | Connection target id. |
| rv | Board reactive values |

## chat_mod_ui *Chat module UI*

### Description

Chat module UI

Chat module server

### Usage

```
chat_mod_ui(id, board, ...)

chat_mod_srv(id, board, update, session, parent, ...)
```

### Arguments

| | |
|---|---|
| id | Module id. |
| board | Board reactiveValues. Read-only. |
| ... | Extra parameters. |
| update | Update reactiveVal to signal change to the board. |
| session | Shiny session. |
| parent | Parent global reactiveValues. |

## check_connections *Check node connection*

### Description

Check node connection

### Usage

```
check_connections(x, target, rv)
```

### Arguments

| | |
|---|---|
| x | block. |
| target | Connection target id. |
| rv | Board reactive values. |

create_block_tool_factory *Create a block tool factory*

## Description

create a tool for a specific block type constructor.

## Usage

```
create_block_tool_factory(provider, app_request, board, parent, session)

create_block_names_tool(provider)

create_remove_block_tool(provider, app_request, board, parent, session)

create_add_stack_tool(
  provider,
  stackable_blocks,
  app_request,
  board,
  parent,
  session
)

create_stackable_blocks_tool(provider, stackable_blocks)

create_get_stack_ids_tool(provider, board)

create_add_block_to_stack_tool(
  provider,
  stackable_blocks,
  app_request,
  board,
  parent,
  session
)

create_add_to_dash_tool(provider, app_request, board, parent, session)

create_remove_from_dash_tool(provider, app_request, board, parent, session)
```

## Arguments

| | |
|---|---|
| provider | AI provider object. |
| app_request | Reactive value containing the tool answer. Maybe useful for post processing. |

| | |
|---|---|
| board | Board object. Not used yet but may be useful. |
| parent | Parent reactive values object. |
| session | Shiny session object. |
| stackable_blocks | Reactive expression returning a vector of blocks that can be stacked. |

---

`create_or_show_block_panel` *Create/Show a block panel*

---

## Description

If panel does not exist, create it and move the block UI from offcanvas to the panel container. If it exists, just select it.

Move block from panel to offcanvas-body.

## Usage

```
create_or_show_block_panel(proxy, id, parent)

hide_block_panel(proxy, id)
```

## Arguments

| | |
|---|---|
| proxy | Dock proxy. |
| id | Block id to show |
| parent | Parent reactive values. |

---

`dashboard_ui` *Dashboard UI*

---

## Description

Dashboard UI

Dashboard grid server

## Usage

```
dashboard_ui(id, board, ...)

dashboard_server(id, board, update, session, parent, ...)
```

### Arguments

| | |
|---|---|
| id | Module id. |
| board | Board reactiveValues. Read-only. |
| ... | Extra parameters. |
| update | Update reactiveVal to signal change to the board. |
| session | Shiny session. |
| parent | Parent global reactiveValues. |

---

## define_conlabel *Define connection label*

---

### Description

Define connection label

### Usage

```
define_conlabel(x, target, rv)
```

### Arguments

| | |
|---|---|
| x | Block object. |
| target | Connection target id. |
| rv | Board reactive values |

---

## dropdown_button *Create a Bootstrap dropdown*

---

### Description

Creates a dropdown menu.

### Usage

```
dropdown_button(..., icon, class = NULL)
```

### Arguments

| | |
|---|---|
| ... | Content. |
| icon | Icon. |
| class | Additional CSS classes for the button. |

### Value

A HTML tag object representing the dropdown element.

## gen_add_rm_link_server *Add/remove block links module*

### Description

Customizable logic for adding/removing links between blocks on the board.

### Usage

```
gen_add_rm_link_server(context_menu)

add_rm_link_ui(id)
```

### Arguments

| | |
|---|---|
| context_menu | Context menu. |
| id | Module ID. |

### Value

NULL.

## generate_code_server *Code generation plugin module*

### Description

All code necessary for reproducing a data analysis as set up in blockr can be made available to the user. Several ways of providing such a script or code snippet are conceivable and currently implemented, we have a modal with copy-to-clipboard functionality. This is readily extensible, for example by offering a download button, by providing this functionality as a generate_code module.

### Usage

```
generate_code_server(id, board, parent, ...)
```

### Arguments

| | |
|---|---|
| id | Namespace ID |
| board | Reactive values object |
| parent | App state |
| ... | Extra arguments passed from parent scope |

### Value

A plugin container inheriting from `generate_code` is returned by `generate_code()`, while the UI component (e.g. `generate_code_ui()`) is expected to return shiny UI (i.e. `shiny::tagList()`) and the server component (i.e. `generate_code_server()`) is expected to return `NULL`.

---

## `generate_code_ui` *Code generation module*

---

### Description

Generate reproducible code from a board.

### Usage

```
generate_code_ui(id, board)
```

### Arguments

| | |
|---|---|
| `id` | Module id. |
| `board` | The initial `board` object |

---

## `list_empty_connections` *List node connections*

---

### Description

List node connections

### Usage

```
list_empty_connections(x, target, rv)
```

### Arguments

| | |
|---|---|
| `x` | block. |
| `target` | Connection target id. |
| `rv` | Board reactive values. |

## main_ui *Main ui for blockr2*

### Description

The board is composed of 2 views pointing to the network module or the grid/dashboard module. Server module for board.

### Usage

```
main_ui(id, board, board_id, plugins = board_plugins(board))

main_server(id, board, board_id, plugins = board_plugins(board))
```

### Arguments

| | |
|---|---|
| id | Unique id. |
| board | Board object. |
| board_id | Board ID. |
| plugins | Board plugins. |

## new_chat_module *Create a board module*

### Description

Extend a DAG board by adding modules.

### Usage

```
new_chat_module(id = "blockr_assistant", title = "AI chat")

new_dashboard_module(id = "dashboard", title = "Dashboard")

new_board_module(
  ui,
  server,
  on_restore = function(board, parent, session, ...) {
     TRUE
 },
  id,
  title,
  context_menu = list(),
  position = NULL,
  options = new_board_options(),
  class = character()
)
```

### Arguments

| | |
|---|---|
| id, title | Module ID and title |
| ui, server | UI and server functions |
| on_restore | Function called when restoring the board state. |
| context_menu | List of context menu entries |
| position | Panel position |
| options | Module options (see blockr.core::new_board_options()). |
| class | (Optional) additional class(es) |

---

## new_context_menu_entry

*Create a context menu entry*

---

### Description

Adds a new entry to the context menu of a board.

### Usage

```
new_context_menu_entry(
  name,
  js,
  action = NULL,
  condition = TRUE,
  id = tolower(gsub(" +", "_", name))
)

context_menu_items(x)

## S3 method for class 'board_module'
context_menu_items(x)

## S3 method for class 'list'
context_menu_items(x)

## S3 method for class 'dag_board'
context_menu_items(x)
```

### Arguments

| | |
|---|---|
| name | Name of the context menu entry. |
| js | JavaScript code to execute when the entry is selected. |
| action | Action to perform when the entry is selected. |
| condition | Condition to determine if the entry should be shown. |

| | |
|---|---|
| id | Unique identifier for the context menu entry. Infered from name if not provided. |
| x | Object |

---

`off_canvas` *Create a Bootstrap Off-Canvas Element*

---

## Description

Creates an off-canvas element, which is a sliding panel that appears from the edges of the viewport. This is a wrapper for Bootstrap's off-canvas component.

## Usage

```
off_canvas(
  id,
  title,
  ...,
  width = "w-25",
  position = c("start", "top", "bottom", "end")
)
```

## Arguments

| | |
|---|---|
| id | Character string. The ID of the off-canvas element. This ID is used to show/hide the element via JavaScript. |
| title | Character string. The title to display in the header of the off-canvas element. |
| ... | Additional UI elements to include in the body of the off-canvas element. |
| width | Character string. Bootstrap width class to apply to the off-canvas element. Defaults to "w-100" (100% width). Common values include "w-75", "w-50", etc. |
| position | Character string. The position from which the off-canvas element should appear. Must be one of "start" (left), "end" (right), "top", or "bottom". Defaults to "start". |

## Value

A HTML tag object representing the off-canvas element.

## See Also

https://getbootstrap.com/docs/5.0/components/offcanvas/

https://getbootstrap.com/docs/5.0/utilities/sizing/

## Examples

```
if (interactive()) {
  library(shiny)
  library(bslib)

  ui <- page_fillable(
    actionButton(
     "toggle",
     "Toggle offcanvas",
     `data-bs-toggle` = "offcanvas",
     `data-bs-target` = "#demo",
     `aria-controls` = "demo"
    ),
    off_canvas(
      id = "demo",
      title = "Settings",
      position = "end",
      sliderInput("n", "Number", 1, 100, 50)
    )
  )

  server <- function(input, output) {}

  shinyApp(ui, server)
}
```

---

`options_ui` *Custom board UI*

---

## Description

Custom board UI

## Usage

```
options_ui(id, x, ...)

## S3 method for class 'dag_board'
board_ui(id, x, plugins = board_plugins(x), ...)
```

## Arguments

| | |
|---|---|
| `id` | Namespace ID. |
| `x` | Board. |
| `...` | Generic consistency. |
| `plugins` | Board plugins. |

run_demo_app *Demo app*

## Description

Run demo app

## Usage

```
run_demo_app(...)

new_dag_board(
  ...,
  modules = list(new_dashboard_module(), new_chat_module()),
  options = dag_board_options(),
  class = character()
)

is_dag_board(x)

as_dag_board(x, ...)

## S3 method for class 'dag_board'
as_dag_board(x, ...)

## S3 method for class 'list'
as_dag_board(x, ...)

dag_board_options()
```

## Arguments

| | |
|---|---|
| ... | Forwarded to new_board. |
| modules | Further modules to pass. |
| options | Board options (see dag_board_options()). |
| class | Additional class(es). |
| x | (Board) object |

`ser_deser_ui` *Ser/deser module UI*

## Description

Ser/deser module UI

## Usage

```
ser_deser_ui(id, board)
```

## Arguments

| | |
|---|---|
| `id` | module ID. |
| `board` | The initial `board` object |

# Index