

Package: dockViewR (via r-universe)

October 12, 2026

Title Layout Manager Widget for R and 'shiny' Apps

Version 0.4.0.9001

Description Provides R bindings to the 'dockview' 'JavaScript' library [<https://dockview.dev/>](https://dockview.dev/). Create fully customizable grid layouts (docks) in seconds to include in interactive R reports with R Markdown or 'Quarto' or in 'shiny' apps [<https://shiny.posit.co/>](https://shiny.posit.co/). In 'shiny' mode, modify docks by dynamically adding, removing or moving panels or groups of panels from the server function. Choose among 18 stunning themes (dark and light), serialise the state of a dock to restore it later.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE, packages = c("`roxy.shinylive`"))

RoxygenNote 7.3.3

Imports htmlwidgets, htmltools, shiny

Suggests knitr, rmarkdown, roxy.shinylive, shinytest2, testthat (>= 3.0.0), visNetwork, quarto, thematic, listviewer, jsonlite, withr

URL <https://github.com/cynkra/dockViewR>,
<https://cynkra.github.io/dockViewR/>

BugReports <https://github.com/cynkra/dockViewR/issues>

Config/testthat/edition 3

Depends R (>= 2.10)

VignetteBuilder quarto

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

Repository <https://cynkra.r-universe.dev>

Date/Publication 2026-10-12 18:12:37 UTC

RemoteUrl <https://github.com/cynkra/dockViewR>

RemoteRef HEAD

RemoteSha df2ebad4f6c4da1cd14703392a9ea3812a0430e8

Contents

add_edge_group	2
add_panel	3
default_add_tab_callback	5
default_remove_tab_callback	6
dock_view	6
dock_view_plugins	8
dock_view_proxy	9
dockViewOutput	9
edge_group	10
get_dock	12
is_add_tab_plugin	13
is_dock_view_plugin	14
is_edge_group	14
is_remove_tab_plugin	15
panel	15
update_dock_view	16

Index	18
--------------	-----------

add_edge_group	<i>Dockview edge group operations</i>
----------------	---------------------------------------

Description

Manipulate edge groups, that is groups pinned to one of the four edges of a dock, from the server.
See <https://dockview.dev/docs/core/groups/edgeGroups>.

Usage

```
add_edge_group(dock, edge_group)

remove_edge_group(dock, position)

set_edge_group_collapsed(dock, position, collapsed)

set_edge_group_visible(dock, position, visible)

set_edge_group_size(dock, position, size)
```

Arguments

dock	Dock proxy object created with dock_view_proxy() .
edge_group	An edge group object created with edge_group() .
position	Edge position. One of "left", "right", "top", "bottom".
collapsed	Whether the edge group should be collapsed.

visible	Whether the edge group should be visible.
size	Target size of the edge group, as a fraction between 0 and 1 of the dock's width (left and right rails) or height (top and bottom ones).

Details

- `add_edge_group()`: pins a new edge group to the requested edge.
- `remove_edge_group()`: removes the edge group pinned to `position`, disposing of the panels it holds.
- `set_edge_group_visible()`: shows or hides the edge group at `position` without removing it. An invisible rail renders at zero and keeps whatever collapsed state it had.
- `set_edge_group_collapsed()`: collapses the edge group at `position` to its `collapsed_size`, or expands it again. A collapsed rail keeps its header strip standing, which is what distinguishes it from an invisible one. This is the server-side equivalent of the user gesture, a click on the rail's active tab.

It is the only route to the collapsed state of a dock that is already running. The state can also be *declared* two other ways, neither of which needs this: `edge_group(collapsed = TRUE)` at construction, and the `edgeGroups` entry of a payload handed to `restore_dock()`, which carries collapsed beside `size` and `visible` and is honoured on restore.

- `set_edge_group_size()`: resizes the edge group at `position` along its own axis, to `size` of the dock: of its width for a "left" or "right" rail, of its height for a "top" or "bottom" one. A fraction rather than pixels, as for `set_size()`, since the server does not know how large the dock is. It is converted to pixels when the call lands, and the rail then holds that width as the dock resizes, as rails do. `dockview` clamps it to the rail's `minimum_size` / `maximum_size` and to the space available. On a collapsed rail the size is the one it expands to.

Read either state back with `is_edge_group_visible()` and `is_edge_group_collapsed()`, and the size, in pixels, from `get_edge_groups()`.

Value

All functions return the dock proxy object invisibly, so they can be chained.

See Also

`edge_group()`, `is_edge_group_visible()`, `is_edge_group_collapsed()`

add_panel

Dockview Panel Operations

Description

Functions to dynamically manipulate panels in a dockview instance.

Usage

```

add_panel(dock, panel, ...)

remove_panel(dock, id)

set_panel_title(dock, id, title)

select_panel(dock, id)

move_panel(dock, id, position)

move_group(dock, from, to, position = NULL)

move_group2(dock, from, to, position = NULL)

set_size(dock, id, size)

```

Arguments

dock	Dock proxy object created with dock_view_proxy() .
panel	A panel object (for <code>add_panel</code>). See panel for parameters.
...	Additional options (currently unused).
id	Panel ID (character string).
title	New panel title.
position	For <code>move_group()</code> and <code>move_group2()</code> , placement relative to the destination group: one of "left", "right", "top", "bottom", "center". For <code>move_panel()</code> , a placement list sharing the vocabulary of panel() 's position: a referencePanel <i>or</i> referenceGroup (not both) to move next to, a direction (one of "above", "below", "left", "right", "within"), and an index used only with direction = "within" to set the tab position in the target group. "above" / "below" / "left" / "right" put the panel in a new group on that side of the reference; "within" drops it into the reference's group. With no reference, the panel moves to a new group at the dock edge in direction.
from	Source group/panel ID (for move operations).
to	Destination group/panel ID (for move operations).
size	Target size for a group along its splitview axis, as a fraction between 0 and 1 of that splitview (for <code>set_size</code>). The other groups in the split keep their relative sizes and share the remaining space. Sizing is best-effort: dockview's minimum group size can clamp a group, in which case the requested fraction is approximate.

Details

- `set_panel_title()`: Sets the title of a panel dynamically.
- `add_panel()`: Adds a new panel to the dockview

- `remove_panel()`: Removes an existing panel
- `select_panel()`: Selects/focuses a specific panel
- `move_panel()`: Moves an existing panel next to a reference panel or group, using the same position vocabulary as `panel()`
- `move_group()`: Moves a group using group IDs
- `move_group2()`: Moves a group using panel IDs
- `set_size()`: Resizes the group a panel belongs to, as a fraction of its splitview; the other groups in the split keep their relative sizes

Value

All functions return the dock proxy object invisibly, allowing for method chaining.

See Also

`panel()`

default_add_tab_callback

Default add tab callback

Description

An example of a JavaScript function that can be used as a default when adding a new tab/panel.

Usage

`default_add_tab_callback()`

Details

An add tab callback runs when the "+" button in a group's header is clicked, and is called with three arguments: the group's header config, the click event and the button itself. The config carries the dock's id in `config.dockId` and the group's in `config.group.id`. To place something at the button, such as a menu, use the button's box, `button.getBoundingClientRect()`. The button stays available to a callback that defers its work, whereas the browser resets `event.currentTarget` to null once the event has been handled, and the click point, `event.clientX` and `event.clientY`, is 0, 0 when the click does not come from a pointer, as for `element.click()` and some assistive technology. The default callback uses neither and sets `input[["<dock_ID>_panel-to-add"]]` to the group's id.

Value

An object of class `JS_EVAL` representing the JavaScript callback.

Examples

```
new_add_tab_plugin(
  enable = TRUE,
  callback = htmlwidgets::JS(
    "(config, event, button) => {
      const box = button.getBoundingClientRect();
      Shiny.setInputValue(
        `${config.dockId}_add-tab`,
        { group: config.group.id, left: box.left, bottom: box.bottom },
        { priority: 'event' }
      );
    }"
  )
)
```

default_remove_tab_callback

Default remove tab callback

Description

An example of a JavaScript function that can be used as a default when removing a tab/panel.

Usage

```
default_remove_tab_callback()
```

Value

An object of class JS_EVAL representing the JavaScript callback.

dock_view

Create a dock view widget

Description

Creates an interactive dock view widget that enables flexible layout management with draggable, resizable, and dockable panels. This is a wrapper around the dockview.dev JavaScript library, providing a powerful interface for creating IDE-like layouts in Shiny applications or R Markdown documents.

Usage

```
dock_view(
  panels = list(),
  ...,
  theme = c("light-spaced", "light", "abyss", "abyss-spaced", "dark", "vs", "dracula",
    "nord", "nord-spaced", "catppuccin-mocha", "catppuccin-mocha-spaced", "monokai",
    "solarized-light", "solarized-light-spaced", "github-dark", "github-dark-spaced",
    "github-light", "github-light-spaced"),
  edge_groups = list(),
  add_tab = new_add_tab_plugin(),
  close_icon = NULL,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

panels	An unnamed list of panel() .
...	Other options. See https://dockview.dev/docs/api/dockview/options/ .
theme	Theme. One of <code>c("light-spaced", "light", "abyss", "abyss-spaced", "dark", "vs", "dracula", "nord", "nord-spaced", "catppuccin-mocha", "catppuccin-mocha-spaced", "monokai", "solarized-light", "solarized-light-spaced", "github-dark", "github-dark-spaced", "github-light", "github-light-spaced")</code> .
edge_groups	Optional unnamed list of edge_group() objects to pin to the edges (left, right, top, bottom) of the dock at initialisation. A panel joins one by referencing its id with <code>position = list(referenceGroup = "<edge-group-id>")</code> , with no direction. See https://dockview.dev/docs/core/groups/edgeGroups .
add_tab	Globally controls the add tab behavior. List with enable and callback. Enable is a boolean, default to FALSE and callback is a JavaScript function passed with JS. See default_add_tab_callback() . By default, the callback sets a Shiny input <code>input[["<dock_ID>_panel-to-add"]]</code> so you can create observers with custom logic.
close_icon	Icon of the close button on each tab in "manual" remove mode (see new_remove_tab_plugin()), as a string of HTML or an <code>htmltools</code> tag such as <code>shiny::icon()</code> . The default, NULL, keeps Font Awesome's xmark. In "auto" mode the tab is dockview's own, close button included, so <code>close_icon</code> has no effect there.
width	Widget width.
height	Widget height.
elementId	When used outside Shiny.

Value

An HTML widget object.

Examples in Shinylive**example-1** [Open in Shinylive](#)

dock_view_plugins	<i>Create dock view plugins</i>
-------------------	---------------------------------

Description

Create plugins to enable additional functionality in dock view interfaces. Currently supports "add_tab" and "remove_tab" plugins.

Usage

```
new_dock_view_plugin(type, ...)

## S3 method for class 'add_tab'
new_dock_view_plugin(type, enable = FALSE, callback = NULL, ...)

## S3 method for class 'remove_tab'
new_dock_view_plugin(type, enable = FALSE, callback = NULL, mode = "auto", ...)

new_add_tab_plugin(enable = FALSE, callback = NULL, ...)

new_remove_tab_plugin(enable = FALSE, callback = NULL, mode = "auto", ...)
```

Arguments

type	Character string specifying the plugin type.
...	Additional plugin configuration arguments.
enable	Logical, whether the plugin functionality is enabled.
callback	Optional JavaScript function. If NULL and enable = TRUE, a default callback is used. See default_add_tab_callback() for the arguments an add_tab callback is called with.
mode	For remove_tab plugins only. One of "auto" or "manual".

Value

A dock view plugin object of class add_tab or remove_tab, depending on the chosen 'type'.

Examples

```
# Add tab plugin
new_dock_view_plugin("add_tab", enable = TRUE)
new_add_tab_plugin(enable = TRUE) # convenience function

# Remove tab plugin
```

```
new_dock_view_plugin("remove_tab", enable = TRUE, mode = "auto")
new_remove_tab_plugin(enable = TRUE, mode = "manual") # convenience function
```

dock_view_proxy	<i>Create a proxy object to modify an existing dockview instance</i>
-----------------	--

Description

This function creates a proxy object that can be used to update an existing dockview instance after it has been rendered in the UI. The proxy allows for server-side modifications of the graph without completely re-rendering it.

Usage

```
dock_view_proxy(id, data = NULL, session = getDefaultReactiveDomain())
```

Arguments

id	Character string matching the ID of the dockview instance to be modified.
data	Unused parameter (for future compatibility).
session	The Shiny session object within which the graph exists. By default, this uses the current reactive domain.

Value

A proxy object of class "dock_view_proxy" that can be used with dockview proxy methods such as `add_panel()`, `remove_panel()`, etc. It contains:

- id: The ID of the dockview instance.
- session: The Shiny session object.

dockViewOutput	<i>Shiny bindings for dock_view</i>
----------------	-------------------------------------

Description

Output and render functions for using dock_view within Shiny applications and interactive Rmd documents.

Usage

```
dockViewOutput(outputId, width = "100%", height = "400px")

dock_view_output(outputId, width = "100%", height = "400px")

renderDockView(expr, env = parent.frame(), quoted = FALSE)

render_dock_view(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a dock_view
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

dockViewOutput and dock_view_output return a Shiny output function that can be used in the UI definition. renderDockView and render_dock_view return a Shiny render function that can be used in the server definition to render a dock_view element.

edge_group

Edge group

Description

Create an edge group, that is a **group** pinned to one of the four edges (left, right, top, bottom) of a `dock_view()`. Edge groups support tabs, drag-and-drop, overflow and the full group panel API. They cannot be maximised, floated or popped out.

Usage

```
edge_group(
  id,
  position = c("left", "right", "top", "bottom"),
  initial_size = NULL,
  minimum_size = NULL,
  maximum_size = NULL,
  collapsed = FALSE,
  collapsed_size = NULL,
  ...
)
```

Arguments

<code>id</code>	Edge group unique id. Used to reference the group from a <code>panel()</code> 's position = <code>list(referenceGroup = ...)</code> .
<code>position</code>	Edge on which to pin the group. One of "left", "right", "top", "bottom".
<code>initial_size</code>	Initial size of the edge group, in pixels.
<code>minimum_size</code>	Minimum size of the edge group, in pixels.
<code>maximum_size</code>	Maximum size of the edge group, in pixels.
<code>collapsed</code>	Whether the edge group is initially collapsed.
<code>collapsed_size</code>	Size of the edge group when collapsed, in pixels. Defaults to 35 on the dockview side.
<code>...</code>	Other options forwarded to <code>api.addEdgeGroup()</code>. See https://dockview.dev/docs/core/groups/edgeGroups. Use the JavaScript option names (camel-Case) for these extras. Two of those options need a dockview module that this package does not bundle, and have no effect here: <code>autoHide</code>, the pinnable tool-window behaviour, and <code>dockToEdgeGroups</code>, which reveals an edge by dragging onto it. Both ship in the commercially licensed dockview-enterprise. Setting either logs an error in the browser console naming the missing module. Everything else about an edge group is core, including collapsing. Clicking the active tab of a rail collapses it to <code>collapsed_size</code> and clicking again expands it, and it can be resized with the sash, or from the server with <code>set_edge_group_size()</code>. Note that collapsing and <code>set_edge_group_visible()</code> are different: a collapsed rail leaves its header strip standing, while an invisible one renders at zero and leaves the collapsed state untouched.

Details

Pass edge groups to `dock_view()` through `edge_groups`, or add them on the fly from the server with `add_edge_group()`.

An edge group's size, visibility, collapsed state and panel contents are part of the serialised layout, so `save_dock()` and `restore_dock()` round-trip a rail along with the rest of the dock.

Value

A list of class `dock_edge_group` with the camelCased options ready to be sent to the dockview JavaScript API. Contains at least:

- `id`: the edge group id (string).
- `position`: one of "left", "right", "top", "bottom".
- `options`: a list of options forwarded to `api.addEdgeGroup()`.

See Also

[add_edge_group\(\)](#), [remove_edge_group\(\)](#), [set_edge_group_visible\(\)](#), [set_edge_group_size\(\)](#).

`get_dock`*get dock*

Description

Edge groups are serialised beside the grid rather than inside it, keyed by edge rather than by group id. `get_edge_groups()` returns that raw record; the group-level helpers ([get_groups_ids\(\)](#), [get_groups_panels\(\)](#), [get_active_views\(\)](#)) fold edge groups in so a rail's panels are reported like any other group's.

Collapsed and invisible are different states: a collapsed rail keeps its header strip standing, while an invisible one renders at zero and keeps whatever collapsed state it had. Set it with [set_edge_group_collapsed\(\)](#).

Usage

```
get_dock(dock)

get_panels(dock)

get_panels_ids(dock)

get_active_group(dock)

get_grid(dock)

get_edge_groups(dock)

is_edge_group_visible(dock, position)

is_edge_group_collapsed(dock, position)

grid_shape(dock)

get_groups(dock)

get_groups_ids(dock)

get_groups_panels(dock)

get_active_views(dock)

get_active_panel(dock)

save_dock(dock)

restore_dock(dock, data)
```

Arguments

dock	Dock proxy created with <code>dock_view_proxy()</code> .
position	Edge position. One of "left", "right", "top", "bottom".
data	Data representing a serialised dock object.

Value

`get_dock` returns a list of 3 elements:

- `grid`: a list representing the dock layout.
- `panels`: a list having the same structure as `panel()` composing the dock.
- `activeGroup`: the current active group (a string).

Each other function allows to deep dive into the returned value of `get_dock()`. `get_panels()` returns the `panels` element of `get_dock()`. `get_panels_ids()` returns a character vector containing all panel ids from `get_panels()`. `get_active_group()` extracts the `activeGroup` component of `get_dock()` as a string. `get_active_views()` is a convenience function that returns the active view in each group. `get_active_panel()` is a convenience function that returns the active panel in the active group. `get_grid()` returns the `grid` element of `get_dock()` which is a list. `grid_shape()` returns the structural shape of `get_grid()`: the branch / leaf nesting, orientation, group ids, panel membership and order, with all absolute geometry (size, width, height) dropped. Pixel geometry is volatile – and in a headless render not even a valid partition – so snapshotting structure alone keeps the layout coverage that matters without the flake. `get_groups()` returns a list of panel groups from `get_grid()`. `get_groups_ids()` returns a character vector of groups ids from `get_groups()`. `get_groups_panels()` returns a list of character vector containing the ids of each panel within each group. `save_dock()` and `restore_dock()` are used for their side effect to allow to respectively serialise and restore a dock object.

TRUE or FALSE, or NULL when no edge group is pinned to position (or the dock has not yet published a state).

TRUE or FALSE, or NULL when no edge group is pinned to position (or the dock has not yet published a state).

Note

Only works with server side functions like `add_panel`. Don't call it from the UI.

is_add_tab_plugin	<i>Check if object is an add tab plugin</i>
-------------------	---

Description

Check if object is an add tab plugin

Usage

```
is_add_tab_plugin(x)
```

Arguments

x An object to test.

Value

Logical value.

is_dock_view_plugin	<i>Check if object is a dock view plugin of specific type</i>
---------------------	---

Description

Check if object is a dock view plugin of specific type

Usage

is_dock_view_plugin(x, type)

Arguments

x An object to test.
type Character string specifying plugin type ("add_tab" or "remove_tab").

Value

Logical value indicating whether the object is the specified plugin type.

is_edge_group	<i>Check whether an object is an edge group</i>
---------------	---

Description

Check whether an object is an edge group

Usage

is_edge_group(x)

Arguments

x An object to test.

Value

Logical value indicating whether x was created with [edge_group\(\)](#).

is_remove_tab_plugin	<i>Check if object is a remove tab plugin</i>
----------------------	---

Description

Check if object is a remove tab plugin

Usage

```
is_remove_tab_plugin(x)
```

Arguments

x	An object to test.
---	--------------------

Value

Logical value.

panel	<i>Dock panel</i>
-------	-------------------

Description

Create a panel for use within a `dock_view()` widget. Panels are the main container components that can be docked, dragged, resized, and arranged within the dockview interface.

Usage

```
panel(  
  id,  
  title,  
  content,  
  active = TRUE,  
  remove = new_remove_tab_plugin(),  
  style = list(padding = "10px", overflow = "auto", height = "100%", margin = "10px"),  
  ...  
)
```

Arguments

id	Panel unique id.
title	Panel title.
content	Panel content. Can be a list of Shiny tags.
active	Is active?
remove	List with two fields: enable and mode. Enable is a boolean and mode is one of manual, auto (default to auto). In auto mode, dockview JS removes the panel when it is closed and all its content. If you need more control over the panel removal, set it to manual so you can explicitly call <code>remove_panel()</code> and perform other tasks. On the server side, a shiny input is available <code>input[["<dock_ID>_panel-to-remove"]]</code> so you can create observers with custom logic.
style	List of CSS style attributes to apply to the panel content. See defaults
...	Other options passed to the API. See https://dockview.dev/docs/api/dockview/panelApi/ . If you pass position, it must be a list with 2 fields: - referencePanel: reference panel id. - direction: one of above, below, left, right or within (above, below, left, right put the panel in a new group, while within puts the panel after its reference panel in the same group). Position is relative to the reference panel target.

Value

A list representing a panel object to be consumed by `dock_view`:

- id: unique panel id (string).
- title: panel title (string).
- content: panel content (`shiny.tag.list` or single `shiny.tag`).
- active: whether the panel is active or not (boolean).
- ...: extra parameters to pass to the API.

update_dock_view	<i>Update options for dockview instance</i>
------------------	---

Description

This does not rerender the widget, just update options like global theme.

Usage

```
update_dock_view(dock, options)
```

Arguments

dock	Dock proxy created with <code>dock_view_proxy()</code> .
options	List of options for the <code>dock_view</code> instance.

Value

This function is called for its side effect. It sends a message to JavaScript through the current websocket connection, leveraging the shiny session object.

Index

add_edge_group, [2](#)
add_edge_group(), [11](#)
add_panel, [3](#), [13](#)

default_add_tab_callback, [5](#)
default_add_tab_callback(), [7](#), [8](#)
default_remove_tab_callback, [6](#)
dock_view, [6](#), [16](#)
dock_view(), [10](#), [11](#), [15](#)
dock_view_output (dockViewOutput), [9](#)
dock_view_plugins, [8](#)
dock_view_proxy, [9](#)
dock_view_proxy(), [2](#), [4](#), [13](#), [16](#)
dockViewOutput, [9](#)

edge_group, [10](#)
edge_group(), [2](#), [3](#), [7](#), [14](#)

get_active_group (get_dock), [12](#)
get_active_group(), [13](#)
get_active_panel (get_dock), [12](#)
get_active_panel(), [13](#)
get_active_views (get_dock), [12](#)
get_active_views(), [12](#), [13](#)
get_dock, [12](#)
get_dock(), [13](#)
get_edge_groups (get_dock), [12](#)
get_edge_groups(), [3](#)
get_grid (get_dock), [12](#)
get_grid(), [13](#)
get_groups (get_dock), [12](#)
get_groups(), [13](#)
get_groups_ids (get_dock), [12](#)
get_groups_ids(), [12](#), [13](#)
get_groups_panels (get_dock), [12](#)
get_groups_panels(), [12](#), [13](#)
get_panels (get_dock), [12](#)
get_panels(), [13](#)
get_panels_ids (get_dock), [12](#)
get_panels_ids(), [13](#)

grid_shape (get_dock), [12](#)
grid_shape(), [13](#)

is_add_tab_plugin, [13](#)
is_dock_view_plugin, [14](#)
is_edge_group, [14](#)
is_edge_group_collapsed (get_dock), [12](#)
is_edge_group_collapsed(), [3](#)
is_edge_group_visible (get_dock), [12](#)
is_edge_group_visible(), [3](#)
is_remove_tab_plugin, [15](#)

JS, [7](#)

move_group (add_panel), [3](#)
move_group2 (add_panel), [3](#)
move_panel (add_panel), [3](#)

new_add_tab_plugin (dock_view_plugins),
[8](#)
new_dock_view_plugin
 (dock_view_plugins), [8](#)
new_remove_tab_plugin
 (dock_view_plugins), [8](#)
new_remove_tab_plugin(), [7](#)

panel, [4](#), [15](#)
panel(), [4](#), [5](#), [7](#), [11](#), [13](#)

remove_edge_group (add_edge_group), [2](#)
remove_edge_group(), [11](#)
remove_panel (add_panel), [3](#)
render_dock_view (dockViewOutput), [9](#)
renderDockView (dockViewOutput), [9](#)
restore_dock (get_dock), [12](#)
restore_dock(), [3](#), [11](#), [13](#)

save_dock (get_dock), [12](#)
save_dock(), [11](#), [13](#)
select_panel (add_panel), [3](#)

set_edge_group_collapsed
 (add_edge_group), [2](#)
set_edge_group_collapsed(), [12](#)
set_edge_group_size (add_edge_group), [2](#)
set_edge_group_size(), [11](#)
set_edge_group_visible
 (add_edge_group), [2](#)
set_edge_group_visible(), [11](#)
set_panel_title (add_panel), [3](#)
set_size (add_panel), [3](#)
set_size(), [3](#)
shiny::icon(), [7](#)

update_dock_view, [16](#)