Package ‘semantic.dashboard’

October 14, 2022

Type Package

Title Dashboard with Fomantic UI Support for Shiny

Version 0.2.1

Description It offers functions for creating dashboard with Fomantic UI.

BugReports `https://github.com/Appsilon/semantic.dashboard/issues`

Encoding UTF-8

License MIT + file LICENSE

Imports utils, shiny (>= 0.12.1), shiny.semantic (>= 0.3.3),
htmltools, glue, checkmate

Suggests testthat, lintr, shinydashboard, covr, knitr, rmarkdown

RoxygenNote 7.1.2

VignetteBuilder knitr

NeedsCompilation no

Author Filip Stachura [aut],
Dominik Krzeminski [aut],
Krystian Igras [aut],
Michał Maj [ctb],
Michał Drzazga [ctb],
Developers Appsilon [cre],
Appsilon [cph]

Maintainer Developers Appsilon <`support+opensource@appsilon.com`>

Repository CRAN

Date/Publication 2021-11-09 18:50:02 UTC

R topics documented:

---

box *Create a box.*

Description

Create a box with additional UI elements.

Usage

```
box(
  ...,
  title = NULL,
  color = "",
  ribbon = TRUE,
  title_side = "top right",
  collapsible = TRUE,
  width = 8,
  id = NULL,
  collapse_icon = "minus",
  expand_icon = "plus"
)
```

Arguments

`...`	UI elements to include within the box.
`title`	Label of the box.
`color`	Color of the box. One of `c("", "red", "orange", "yellow","olive", "green", "teal", "blue", "violet", "purple", "pink", "brown", "grey", "black")`
`ribbon`	Should label be presented as ribbon.
`title_side`	Side of a label. One of `c("top", "bottom", "top left","top right", "bottom left", "bottom right")` if `ribbon = FALSE`, or one of `c("top left", "top right")` if `ribbon = TRUE`
`collapsible`	Should minimize button be added to label.
`width`	Width of the box.
`id`	ID of the box.
`collapse_icon`	Icon class to be used for collapsing (when `collapsible = TRUE`).
`expand_icon`	Icon class to be used for expanding (when `collapsible = TRUE`).

Value

A box that can be passed to `dashboardBody`

Examples

```
box(title = "Sample box", color = "blue", width = 11,
    "This is a box content"
)
```

`column` *Create a column.*

Description

Create a column with additional UI elements.

Usage

```
column(width, ...)
```

Arguments

`width`	Width of the column. Between 1 and 16.
`...`	UI elements to include within the column.

Value

A column that can be passed to `dashboardPage`

dashboard_body *Create a body of a dashboard.*

Description

Create a body of a dashboard with tabs and other additional UI elements.

Usage

```
dashboard_body(..., class = "")

dashboardBody(..., class = "")
```

Arguments

...	UI elements to include within the body.
class	CSS class to be applied to the container of dashboardBody. Note it's not the <body> tag.

Value

A tab that can be passed to dashboardPage

Functions

- dashboardBody: Create a body of a dashboard (alias for dashboard_body for compatibility with shinydashboard)

Examples

```
if(interactive()){

  library(shiny)
  library(semantic.dashboard)

  ui <- dashboardPage(
    dashboardHeader(color = "blue"),
    dashboardSidebar(side = "left", size = "thin", color = "teal",
                     sidebarMenu(
                       menuItem(tabName = "tab1", "Tab 1"),
                       menuItem(tabName = "tab2", "Tab 2"))),
    dashboardBody(tabItems(
      tabItem(tabName = "tab1", p("Tab 1")),
      tabItem(tabName = "tab2", p("Tab 2"))))
  )

  server <- function(input, output) {
  }
  shinyApp(ui, server)
}
```

dashboard_header *Create a header of a dashboard.*

Description

Create a header of a dashboard with other additional UI elements. Hint: use `shiny::tagList()` if you want to add multiple elements in `left` / `center` or `right`.

Usage

```
dashboard_header(
  ...,
  left = NULL,
  center = NULL,
  right = NULL,
  title = NULL,
  titleWidth = "thin",
  logo_align = "center",
  logo_path = "",
  color = "",
  inverted = FALSE,
  disable = FALSE,
  show_menu_button = TRUE,
  menu_button_label = "Menu",
  class = ""
)

dashboardHeader(
  ...,
  left = NULL,
  center = NULL,
  right = NULL,
  title = NULL,
  titleWidth = "thin",
  logo_align = "center",
  logo_path = "",
  color = "",
  inverted = FALSE,
  disable = FALSE,
  show_menu_button = TRUE,
  menu_button_label = "Menu",
  class = ""
)
```

Arguments

`...`	UI elements to include within the header. They will be displayed on the right side.
`left`	UI element to put on the left of the header. It will be placed after (to the right) the title and menu button (if they exist).
`center`	UI element to put in the center of the header.
`right`	UI element to put to the right of the header. It will be placed before elements defined in `...` (if there are any).
`title`	Dashboard title to be displayed in the upper left corner. If NULL, will not display any title field. Use "" for an empty title.
`titleWidth`	Title field width, one of `c(NULL, "very thin", "thin", "wide", "very wide")`
`logo_align`	Where should logo be placed. One of `c("left", "center")`
`logo_path`	Path or URL of the logo to be shown in the header.
`color`	Color of the sidebar / text / icons (depending on the value of ‘inverted‘ parameter. \ One of `c("", "red", "orange", "yellow", "olive", "green", "teal", "blue", "violet", "purple", "pink", "brown", "grey", "black")`
`inverted`	If FALSE sidebar will be white and text will be colored. \ If TRUE text will be white and background will be colored. Default is `FALSE`.
`disable`	If `TRUE`, don’t display the header.
`show_menu_button`	If `FALSE`, don’t display the menu button. Default is `TRUE`.
`menu_button_label`	Text of the menu button. Default is `"Menu"`.
`class`	CSS class to be applied to the container of `dashboardHeader`.

Value

A header that can be passed to `dashboardPage`

Functions

- `dashboardHeader`: Create a header of a dashboard (alias for `dashboard_header` for compatibility with `shinydashboard`)

Examples

```
if(interactive()) {

  library(shiny)
  library(semantic.dashboard)

  ui <- dashboardPage(
    dashboardHeader(color = "blue", inverted = TRUE),
    dashboardSidebar(side = "left", size = "thin", color = "teal",
                     sidebarMenu(
                       menuItem(tabName = "tab1", "Tab 1"),
                       menuItem(tabName = "tab2", "Tab 2"))),
    dashboardBody(tabItems(
      tabItem(tabName = "tab1", p("Tab 1")),
      tabItem(tabName = "tab2", p("Tab 2"))))
  )

  server <- function(input, output) {
  }

  shinyApp(ui, server)
}
```

`dashboard_page` *Create a dashboard.*

Description

Create a page with menu item sidebar and body containing tabs and other additional elements.

Usage

```
dashboard_page(
  header,
  sidebar,
  body,
  title = "",
  suppress_bootstrap = TRUE,
  theme = NULL,
  margin = TRUE,
  class = "",
  sidebar_and_body_container_class = ""
)

dashboardPage(
  header,
  sidebar,
  body,
  title = "",
  suppress_bootstrap = TRUE,
  theme = NULL,
  margin = TRUE,
  class = "",
  sidebar_and_body_container_class = ""
)
```

Arguments

`header`	Header of a dashboard.
`sidebar`	Sidebar of a dashboard.
`body`	Body of a dashboard.
`title`	Title of a dashboard.
`suppress_bootstrap`	There are some conflicts in CSS styles between FomanticUI and Bootstrap. For the time being it's better to suppress Bootstrap. If `TRUE` bootstrap dependency from `shiny` will be disabled.
`theme`	Theme name or path. For possible options see `semanticPage`.
`margin`	If `TRUE`, margin to be applied to the whole dashboard. Defaults to `TRUE`.
`class`	CSS class to be applied to the page container (<body> tag).
`sidebar_and_body_container_class`	CSS class to be applied to the `div` containing `dashboardSidebar` and `dashboardBody`.

Value

Dashboard.

Functions

- `dashboardPage`: Create a dashboard (alias for `dashboard_page` for compatibility with `shinydashboard`)

Examples

```
if(interactive()){

  library(shiny)
  library(semantic.dashboard)

  ui <- dashboardPage(
    dashboardHeader(color = "blue"),
    dashboardSidebar(side = "left", size = "thin", color = "teal",
                     sidebarMenu(
                       menuItem(tabName = "tab1", "Tab 1"),
                       menuItem(tabName = "tab2", "Tab 2"))),
    dashboardBody(tabItems(
      tabItem(tabName = "tab1", p("Tab 1")),
      tabItem(tabName = "tab2", p("Tab 2"))))
  )

  server <- function(input, output) {
  }

  shinyApp(ui, server)
}
```

`dashboard_sidebar` *Create a sidebar of a dashboard.*

Description

Create a pushable sidebar of a dashboard with menu items and other additional UI elements.

Usage

```
dashboard_sidebar(
  ...,
  side = "left",
  size = "thin",
  color = "",
  inverted = FALSE,
  closable = FALSE,
  pushable = TRUE,
  center = FALSE,
  visible = TRUE,
  disable = FALSE,
  overlay = FALSE,
  dim_page = FALSE,
  class = ""
)

dashboardSidebar(
  ...,
  side = "left",
  size = "thin",
  color = "",
  inverted = FALSE,
  closable = FALSE,
  pushable = TRUE,
  center = FALSE,
  visible = TRUE,
  disable = FALSE,
  overlay = FALSE,
  dim_page = FALSE,
  class = ""
)
```

Arguments

`...`	UI elements to include within the sidebar.
`side`	Placement of the sidebar. One of `c("left", "right", "top", "bottom")`
`size`	Size of the sidebar. One of `c("", "thin", "very thin", "wide", "very wide")`

`color`	Color of the sidebar / text / icons (depending on the value of ‘inverted‘ parameter. \ One of `c("", "red", "orange", "yellow", "olive", "green", "teal", "blue", "violet", "purple", "pink", "brown", "grey", "black")`
`inverted`	If FALSE sidebar will be white and text will be colored. \ If TRUE text will be white and background will be colored. Default is `FALSE`.
`closable`	If `TRUE` allow close sidebar by clicking in the body. Default to `FALSE`
`pushable`	If `TRUE` the menu button is active. Default to `TRUE`
`center`	Should label and icon be centerd on menu items. Default to `FALSE`
`visible`	Should sidebar be visible on start. Default to `TRUE`
`disable`	If `TRUE`, don't display the sidebar.
`overlay`	If `TRUE`, opened sidebar will cover the tab content. Otherwise it is displayed next to the content. Relevant only for sidebar positioned on left or right. Default to `FALSE`
`dim_page`	If `TRUE`, page content will be darkened when sidebr is open. Default to `FALSE`
`class`	CSS class to be applied to the container of `dashboardSidebar`.

Value

A sidebar that can be passed to `dashboardPage`

Functions

- `dashboardSidebar`: Create a sidebar of a dashboard (alias for `dashboard_sidebar` for compatibility with `shinydashboard`)

Examples

```
if(interactive()){

  library(shiny)
  library(semantic.dashboard)

  ui <- dashboardPage(
    dashboardHeader(color = "blue"),
    dashboardSidebar(side = "left", size = "thin", color = "teal",
                     sidebarMenu(
                       menuItem(tabName = "tab1", "Tab 1"),
                       menuItem(tabName = "tab2", "Tab 2"))),
    dashboardBody(tabItems(
      tabItem(tabName = "tab1", p("Tab 1")),
      tabItem(tabName = "tab2", p("Tab 2"))))
  )

  server <- function(input, output) {
  }

  shinyApp(ui, server)
}
```

`dropdown_menu` *Create a dropdown menu.*

Description

Create a dropdown menu with additional UI elements.

Usage

```
dropdown_menu(..., type = "messages", icon = NULL, show_counter = TRUE)

dropdownMenu(..., type = "messages", icon = NULL, show_counter = TRUE)
```

Arguments

`...`	UI elements to include within the dropdown menu.
`type`	Type of the displayed items.
`icon`	Icon of the dropdown menu. If not specyfied created based on `type` agrument.
`show_counter`	If true circular label with counter is going to be shown for dropdown.

Value

A dropdown menu that can be passed to `dashboardHeader`

Functions

- `dropdownMenu`: Create a dropdown menu (alias for `dropdown_menu` for compatibility with `shinydashboard`)

Examples

```
dropdownMenu(icon = icon("warning sign"), taskItem("Project progress...", 50.777, color = "red"))
dropdownMenu(type = "notifications", notificationItem("This is notification!", color = "red"))
```

`dropdown_menu_output` *Create a dropdown menu output.*

Description

UI-side function for dynamic dropdownMenu.

Usage

```
dropdown_menu_output(outputId)

dropdownMenuOutput(outputId)
```

Arguments

outputId	Id of the output.

Value

A dropdown menu that can be passed to dashboardHeader

Functions

- dropdownMenuOutput: Create a dropdown menu output (alias for dropdown_menu_output for compatibility with shinydashboard)

Examples

```
## Not run:
dropdownMenuOutput("dropdown")

output$dropdown <- renderDropdownMenu({
  dropdownMenu(messageItem("Michał", "Test message", color = "teal"),
              messageItem("Marek", "Another test!", icon = "warning", color = "red"))
})

## End(Not run)
```

---

get_dashboard_dependencies

Get the semantic.dashboard dependencies

Description

To add dependencies in the future follow the htmlDependency help.

Usage

```
get_dashboard_dependencies()
```

Value

semantic.dashboard dependencies

`icon` *Create Semantic UI icon tag (alias for* `icon` *for compatibility with* `shinydashboard`*)*

Description

This creates an icon tag using Semantic UI styles.

Usage

```
icon(type, ...)
```

Arguments

`type`	A name of an icon. Look at http://semantic-ui.com/elements/icon.html for all possibilities.
`...`	Other arguments to be added as attributes of the tag (e.g. style, class etc.)

Examples

```
icon("dog")
```

`light_semantic_palette` *Semantic light colors https://github.com/Semantic-Org/Semantic-UI/blob/master/src/themes/default/globals/site.variables*

Description

Semantic light colors https://github.com/Semantic-Org/Semantic-UI/blob/master/src/themes/default/globals/site.variables

Usage

```
light_semantic_palette
```

Format

An object of class `character` of length 13.

menu_item *Create a menu item.*

Description

Create a menu item corresponding to a tab.

Usage

```
menu_item(
  text,
  ...,
  icon = NULL,
  tabName = NULL,
  href = NULL,
  newtab = TRUE,
  selected = FALSE
)

menuItem(
  text,
  ...,
  icon = NULL,
  tabName = NULL,
  href = NULL,
  newtab = TRUE,
  selected = FALSE
)

menuSubItem(
  text,
  ...,
  icon = NULL,
  tabName = NULL,
  href = NULL,
  newtab = TRUE,
  selected = FALSE
)
```

Arguments

`text`	Text to show for the menu item.
`...`	This may consist of menuSubItems.
`icon`	Icon of the menu item. (Optional)
`tabName`	Id of the tab. Not compatible with href.
`href`	A link address. Not compatible with tabName.

`newtab`	If href is supplied, should the link open in a new browser tab?
`selected`	If TRUE, this menuItem will start selected.

Value

A menu item that can be passed `sidebarMenu`

Functions

- `menuItem`: Create a menu item (alias for `manu_item` for compatibility with `shinydashboard`)
- `menuSubItem`: Create a menu item (alias for `manu_item` for compatibility with `shinydashboard`)

Examples

```
menuItem(tabName = "plot_tab", text = "My plot", icon = icon("home"))
```

`menu_item_output` *Create a menu item output.*

Description

UI-side function for dynamic manuItem.

Usage

```
menu_item_output(outputId)

menuItemOutput(outputId)
```

Arguments

`outputId`	Id of the output.

Value

A menu item that can be passed to `sidebarMenu`

Functions

- `menuItemOutput`: Create a menu item output (alias for `menu_item_output` for compatibility with `shinydashboard`)

message_item *Create a message item.*

Description

Create a message item.

Usage

```
message_item(from, message, ..., icon = "user")

messageItem(from, message, ..., icon = "user")
```

Arguments

from	Who the message is from.
message	Text of the message.
...	Additional UI elements to include within the dropdown menu.
icon	Additional icon.

Value

A message item that can be passed to `dropdownMenu`

Functions

- `messageItem`: Create a message item (alias for `message_item` for compatibility with `shinydashboard`)

Examples

```
messageItem("Marek", "Another test!", icon = "warning")
```

notification_item *Create a notification item.*

Description

Create a notification item.

Usage

```
notification_item(text, icon = "warning", color = "")

notificationItem(text, icon = "warning", color = "")
```

Arguments

`text`	Text of the notification.
`icon`	Additional icon.
`color`	Color of the notification item. One of `c("", "red", "orange", "yellow", "olive", "green", "teal", "blue","violet", "purple", "pink", "brown", "grey", "black")`

Value

A notification item that can be passed to `dropdownMenu`

Functions

- `notificationItem`: Create a notification item (alias for `notification_item` for compatibility with `shinydashboard`)

Examples

```
notificationItem("This is notification!", color = "red")
```

`render_dropdown_menu` *Create a dropdown menu output.*

Description

Server-side function for dynamic dropdownMenu.

Usage

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

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

Arguments

`expr`	dropdownMenu.
`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

A dynamic dropdown menu that can be assigned to output.

Functions

- `renderDropdownMenu`: Create a dropdown menu output (alias for `render_dropdown_menu` for compatibility with `shinydashboard`)

Examples

```
## Not run:
dropdownMenuOutput("dropdown")

output$dropdown <- renderDropdownMenu({
  dropdownMenu(messageItem("Michał", "Test message", color = "teal"),
               messageItem("Marek", "Another test!", icon = "warning", color = "red"))
})

## End(Not run)
```

`render_menu` *Create a menu output.*

Description

Server-side function for dynamic sidebarMenu.

Usage

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

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

Arguments

`expr`	menu.
`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

A dynamic menu that can be assigned to output.

Functions

- `renderMenu`: Create a menu output (alias for `render_menu` for compatibility with `shinydashboard`)

`render_value_box` *Create a value box output.*

Description

Server-side function for dynamic valueBox.

Usage

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

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

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

Arguments

`expr`	ValueBox.
`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

A dynamic valueBox that can be assigned to output.

Functions

- `renderValueBox`: Create a value box output (alias for `render_value_box`)
- `renderInfoBox`: Create a value box output (alias for `render_value_box`)

Examples

```
## Not run:
valueBoxOutput("value_box")

output$value_box <- renderValueBox({
  valueBox(
    value = 33.45,
    subtitle = "Simple valuebox",
    icon = icon("bar chart"),
    color = "purple",
    width = 5)
})

## End(Not run)
```

`semantic.dashboard` *semantic.dashboard*

Description

semantic.dashboard

`semantic_palette` *Semantic colors https://github.com/Semantic-Org/Semantic-UI/blob/master/src/themes/default/globals/site.variables*

Description

Semantic colors https://github.com/Semantic-Org/Semantic-UI/blob/master/src/themes/default/globals/site.variables

Usage

```
semantic_palette
```

Format

An object of class `character` of length 13.

`sidebar_menu` *Create a sidebar menu.*

Description

Create a sidebar menu with menu items.

Usage

```
sidebar_menu(..., id = "uisidebar")

sidebarMenu(..., id = "uisidebar")
```

Arguments

`...`	Menu items.
`id`	The sidebar id class also used for update input on server side. Default is `uisidebar`

Details

It's possible to set selected menu item by setting 'selected = TRUE' in 'menuItem'.

Value

A sidebar menu that can be passed `dashboardSidebar`

Functions

- `sidebarMenu`: Create a sidebar menu (alias for `sidebar_menu` for compatibility with `shinydashboard`)

Examples

```
sidebarMenu(
  menuItem(tabName = "plot_tab", text = "My plot", icon = icon("home")),
 menuItem(tabName = "table_tab", text = "My table", icon = icon("smile"), selected = TRUE)
  )
```

`sidebar_menu_output` *Create a sidebar menu output.*

Description

UI-side function for dynamic sidebarMenu.

Usage

```
sidebar_menu_output(outputId)

sidebarMenuOutput(outputId)
```

Arguments

`outputId`	Id of the output.

Value

A sidebar menu that can be passed to `dashboardSidebar`

Functions

- `sidebarMenuOutput`: Create a sidebar menu output (alias for `sidebar_menu_output` for compatibility with `shinydashboard`)

tab_box *Create a tab box.*

Description

Create a tab box with additional UI elements.

Usage

```
tab_box(
  tabs,
  title = NULL,
  color = "",
  ribbon = TRUE,
  title_side = "top right",
  collapsible = TRUE,
  width = 8,
  id = NULL,
  ...
)

tabBox(
  tabs,
  title = NULL,
  color = "",
  ribbon = TRUE,
  title_side = "top right",
  collapsible = TRUE,
  width = 8,
  id = NULL,
  ...
)
```

Arguments

`tabs`	Tabs to include within the box.
`title`	Label of the box.
`color`	Color of the box. One of `c("", "red", "orange", "yellow","olive", "green", "teal", "blue", "violet", "purple", "pink", "brown", "grey", "black")`
`ribbon`	Should label be presented as ribbon.
`title_side`	Side of a label. One of `c("top", "bottom", "top left","top right", "bottom left", "bottom right")` if `ribbon = FALSE`, or one of `c("top left", "top right")` if `ribbon = TRUE`
`collapsible`	Should minimize button be added to label.
`width`	Width of the box.

`id`	ID of the box.
`...`	other elements of the box.

Value

A box that can be passed to `dashboardBody`

Functions

- `tabBox`: Create a tab box (alias for `tab_box` for compatibility with `shinydashboard`)

Examples

```
tabBox(title = "Sample tab box", color = "blue", width = 5,
       tabs = list(
         list(menu = "First Tab", content = "This is first tab"),
         list(menu = "Second Tab", content = "This is second tab")
       ))
```

`tab_item` *Create a tab*

Description

Create a tab panel with additional UI elements.

Usage

```
tab_item(tabName, ..., fluid = TRUE)

tabItem(tabName, ..., fluid = TRUE)
```

Arguments

`tabName`	Id of the tab.
`...`	UI elements to include within the tab.
`fluid`	Controls whether tab width should be 100% (TRUE) or limited by Foomantic UI breakpoints (FALSE).

Value

A tab that can be passed to `dashboardBody`

Functions

- `tabItem`: Create a tab (alias for `tab_item` for compatibility with `shinydashboard`)

Examples

```
tab_item(tabName = "tab1", "Tab 1")
```

tab_items *Create a panel with tabs.*

Description

Create a panel with tabs.

Usage

```
tab_items(...)

tabItems(...)
```

Arguments

`...`	Tabs.

Value

A panel with tabs that can be passed to `dashboardBody`

Functions

- `tabItems`: Create a panel with tabs (alias for `tab_items` for compatibility with `shinydashboard`)

Examples

```
tabItems(
 tabItem(tabName = "tab1", "Tab 1"),
 tabItem(tabName = "tab2", "Tab 2"))
```

task_item *Create a task item.*

Description

Create a task item.

Usage

```
task_item(text, value, color = "")

taskItem(text, value, color = "")
```

Arguments

`text`	Progress bar label.
`value`	Progress bar value.
`color`	Color of the task item. One of `c("", "red", "orange","yellow", "olive", "green", "teal", "blue", "violet", "purple", "pink","brown", "grey", "black")`

Value

A task item that can be passed to `dropdownMenu`

Functions

- `taskItem`: Create a task item (alias for `taks_item` for compatibility with `shinydashboard`)

Examples

```
taskItem("Project progress...", 50.777, color = "red")
```

`validate_tab_name` *Valid tab name should not containt dot character '.'.*

Description

Valid tab name should not containt dot character '.'.

Usage

```
validate_tab_name(name)
```

Arguments

`name`	Tab name to validate.

value_box *Create a valueBox.*

Description

Create a valueBox with additional UI elements.

Usage

```
value_box(subtitle, value, icon = NULL, color = "blue", width = 5, size = "")

valueBox(subtitle, value, icon = NULL, color = "blue", width = 5, size = "")

infoBox(subtitle, value, icon = NULL, color = "blue", width = 5, size = "")
```

Arguments

`subtitle`	Label of the valueBox.
`value`	Value of the valueBox.
`icon`	Icon of the valueBox.
`color`	Color of the valueBox. One of `c("", "red", "orange", "yellow","olive", "green", "teal", "blue", "violet", "purple", "pink", "brown", "grey", "black")`
`width`	Width of the valueBox.
`size`	Size of value. One of `c("mini", "tiny", "small", "", "large", "huge")`. Default is "".

Value

A valueBox that can be passed to `dashboardBody`

Functions

- `valueBox`: Create a valueBox (alias for `value_box`)
- `infoBox`: Create a valueBox (alias for `value_box`)

Examples

```
valueBox("Unread Mail", 44, icon("mail"), color = "blue", width = 5, size = "tiny")
```

`value_box_output` *Create a value box output.*

Description

UI-side function for dynamic valueBox.

Usage

```
value_box_output(outputId, width = 5)

valueBoxOutput(outputId, width = 5)

infoBoxOutput(outputId, width = 5)
```

Arguments

`outputId`	Id of the output.
`width`	Width of the valueBox.

Value

A value box that can be passed to `dashboardBody`

Functions

- `valueBoxOutput`: Create a valueBox output (alias for `value_box_output`)
- `infoBoxOutput`: Create a valueBox output (alias for `value_box_output`)

Examples

```
## Not run:
valueBoxOutput("value_box")

output$value_box <- renderValueBox({
  valueBox(
    value = 33.45,
    subtitle = "Simple valuebox",
    icon = icon("bar chart"),
    color = "purple",
    width = 5)
})

## End(Not run)
```

Index