Package 'bscui'

March 31, 2024

Type Package

Title Build SVG Custom User Interface

Version 0.1.5

Description Render SVG as interactive figures to display contextual
information, with selectable and clickable user interface elements.
These figures can be seamlessly integrated into 'rmarkdown' and 'Quarto'
documents, as well as 'shiny' applications, allowing manipulation of elements
and reporting actions performed on them.
Additional features include pan, zoom in/out functionality, and the ability
to export the figures in SVG or PNG formats.

URL <https://patzaw.github.io/bscui/>, <https://github.com/patzaw/bscui/>

BugReports <https://github.com/patzaw/bscui/issues>

Depends R (>= 4.1)

Imports htmlwidgets, webshot2

Suggests knitr, rmarkdown, here, xml2, dplyr, readr, stringr, glue,
scales, shiny, reactable, reactable.extras

License GPL-3

Encoding UTF-8

VignetteBuilder knitr

RoxygenNote 7.3.1

NeedsCompilation no

Author Patrice Godard [aut, cre, cph]
(<<https://orcid.org/0000-0001-6257-9730>>)

Maintainer Patrice Godard <patrice.godard@gmail.com>

Repository CRAN

Date/Publication 2024-03-31 18:20:02 UTC

R topics documented:

---

add_bscui_element   *Add an SVG element to the UI*

Description

Add an SVG element to the UI

Usage

```
add_bscui_element(proxy, id, svg_txt, ui_type = NULL, title = NULL)
```

Arguments

proxy	a bscui_Proxy object
id	the identifier of the element to add (will replace the id attribute of the provided svg if any)
svg_txt	a character with SVG code of one element and its children
ui_type	either "selectable", "button" or "none". If NULL (default), the element won't be available as UI
title	a description of the element to display on mouseover event

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

bscui   *Build SVG Custom User Interface*

Description

Build SVG Custom User Interface

Usage

```
bscui(
  svg_txt,
  sanitize_attributes = TRUE,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

svg_txt   a character with SVG code

sanitize_attributes
                 logical indicating if '<' and '>' characters in element attributes must be replaced
                 by text

width, height   widget width: must be a valid CSS unit (like '100\%', '400px', 'auto') or a
                 number, which will be coerced to a string and have 'px' appended.

elementId   hmtl identifier of the widget

Value

An htmlwidget object

Examples

```
##################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
    "examples", "Animal_cells.svg.gz",
    package="bscui"
))
info <- readr::read_tsv(system.file(
    "examples", "uniprot_cellular_locations.txt.gz",
    package="bscui"
), col_types=strrep("c", 6)) |>
    mutate(id = sub("-", "", `Subcellular location ID`))

##################################@
### Building the figure ----

figure <- bscui(svg) |>
    set_bscui_ui_elements(
        info |>
            mutate(
                ui_type = "selectable",
                title = Name
            ) |>
            select(id, ui_type, title)
    ) |>
    set_bscui_styles(
        info |>
            filter(Name == "Cytosol") |>
            mutate(fill = "#FF7F7F") |>
            select(id, fill)
    ) |>
    set_bscui_attributes(
        info |>
            filter(Name == "Cytoskeleton") |>
            mutate(display = "none") |>
            select(id, display)
    ) |>
    set_bscui_selection("SL0188") |>
    set_bscui_options(zoom_min=1, clip=TRUE)
figure

##################################@
### Saving the figure ----

if(interactive()){
    ## Temporary directory to save example file
```

```
    tdir <- tempdir()

    ## Interactive html file
    f_path <- file.path(tdir, "figure.html")
    figure |> htmlwidgets::saveWidget(file=f_path)
    cat(f_path)

    ## PNG image
    f_path <- file.path(tdir, "figure.png")
    figure |>
        set_bscui_options(show_menu = FALSE) |>
        export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

---

bscui-shiny *'shiny' bindings for bscui*

Description

Output and render functions for using bscui within 'shiny' applications.

Usage

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

renderBscui(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 bscui
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.

Details

The [bscuiProxy()](#) function can be used to allow user interface dynamic updates.

Value

An output or render function that enables the use of the widget within 'shiny' applications.

See Also

bscuiProxy()

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

bscuiProxy   *Manipulate an existing bscui instance in a 'shiny' app*

Description

Manipulate an existing bscui instance in a 'shiny' app

Usage

```
bscuiProxy(shinyId, session = shiny::getDefaultReactiveDomain())
```

Arguments

shinyId   single-element character vector indicating the 'shiny' output ID of the UI to
             modify

session   the 'shiny' session object to which the UI belongs; usually the default value will
             suffice

Details

This function creates a proxy object that can be used to manipulate an existing bscui instance in a
'shiny' app using different methods:

- update_bscui_ui_elements: change type and title of elements
- update_bscui_styles: set style of UI elements
- update_bscui_attributes set attributes of a UI element
- update_bscui_selection: chose selected elements
- click_bscui_element: trigger a single or double click on a UI element
- order_bscui_elements: change elements order (e.g. move them forward)
- add_bscui_element: add an SVG element to the UI
- remove_bscui_elements: remove SVG elements from the UI
- get_bscui_svg: get the displayed SVG in R session

Value

A bscui_Proxy object with an "id" and a "session" slot.

See Also

bscui-shiny

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

click_bscui_element   *Trigger a click event on a clickable element*

Description

Trigger a click event on a clickable element

Usage

```
click_bscui_element(proxy, element_id, dbl_click = FALSE)
```

Arguments

proxy	a bscui_Proxy object
element_id	element identifier on which the click will be triggered
dbl_click	logical indicating the type of click (default: FALSE => single click is triggered)

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
```

```
      bscui(from_shiny[[n]]) |> print()
   }
}
```

export_bscui_to_image   *Save a bscui widget to an image file*

Description

Save a bscui widget to an image file

Usage

```
export_bscui_to_image(
  widget,
  file,
  selector = ".bscui",
  zoom = 1,
  quiet = TRUE,
  ...
)
```

Arguments

widget	a bscui object
file	name of output file. Should end with an image file type (.png, .jpg, .jpeg, or .webp) or .pdf.
selector	(webshot2::webshot()) one or more CSS selectors specifying a DOM element to set the clipping rectangle to (default: ".bscui").
zoom	(webshot2::webshot()) If TRUE (default), status updates via console messages are suppressed.
quiet	(webshot2::webshot()) a number specifying the zoom factor.
...	additional parameters for webshot2::webshot() suppressed.

Value

Invisibly returns the normalized path to the image. The character vector will have a class of "webshot".

See Also

webshot2::webshot()

Examples

```
###################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
    "examples", "Animal_cells.svg.gz",
    package="bscui"
))
info <- readr::read_tsv(system.file(
    "examples", "uniprot_cellular_locations.txt.gz",
    package="bscui"
), col_types=strrep("c", 6)) |>
    mutate(id = sub("-", "", `Subcellular location ID`))

###################################@
### Building the figure ----

figure <- bscui(svg) |>
    set_bscui_ui_elements(
        info |>
            mutate(
                ui_type = "selectable",
                title = Name
            ) |>
            select(id, ui_type, title)
    ) |>
    set_bscui_styles(
        info |>
            filter(Name == "Cytosol") |>
            mutate(fill = "#FF7F7F") |>
            select(id, fill)
    ) |>
    set_bscui_attributes(
        info |>
            filter(Name == "Cytoskeleton") |>
            mutate(display = "none") |>
            select(id, display)
    ) |>
    set_bscui_selection("SL0188") |>
    set_bscui_options(zoom_min=1, clip=TRUE)
figure

###################################@
### Saving the figure ----

if(interactive()){
    ## Temporary directory to save example file
```

```
    tdir <- tempdir()

    ## Interactive html file
    f_path <- file.path(tdir, "figure.html")
    figure |> htmlwidgets::saveWidget(file=f_path)
    cat(f_path)

    ## PNG image
    f_path <- file.path(tdir, "figure.png")
    figure |>
        set_bscui_options(show_menu = FALSE) |>
        export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

---

get_bscui_svg   *Get the displayed SVG*

Description

Get the displayed SVG

Usage

```
get_bscui_svg(proxy)
```

Arguments

proxy   a bscui_Proxy object

Value

the provided proxy object

Examples

```
if(interactive()){
    from_shiny <- new.env()
    shiny::runApp(system.file(
        "examples", "shiny-anatomogram", package = "bscui"
    ))
    for(n in names(from_shiny)){
        bscui(from_shiny[[n]]) |> print()
    }
}
```

---

order_bscui_elements   *Change element order in the SVG*

Description

Change element order in the SVG

Usage

```
order_bscui_elements(
  proxy,
  element_ids,
  where = c("front", "back", "forward", "backward")
)
```

Arguments

proxy	a `bscui_Proxy` object
element_ids	the identifiers of the element to move
where	where to move the elements (default: "front")

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
       "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

remove_bscui_elements   *Remove SVG elements from the UI*

Description

Remove SVG elements from the UI

Usage

```
remove_bscui_elements(proxy, element_ids)
```

Arguments

proxy	a [bscui_Proxy](#) object
element_ids	the identifiers of the elements to remove

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

set_bscui_attributes   *Set attributes of elements of a bscui widget*

Description

Set attributes of elements of a bscui widget

Usage

```
set_bscui_attributes(
  widget,
  element_attributes,
  to_ignore = NULL,
  targeted_tags = widget$x$structure_shapes,
  append = FALSE
)
```

Arguments

widget	a [bscui](#) object
element_attributes	
	a data frame with an **id** column providing the element identifier and one column per attribute name.
to_ignore	identifiers of elements to ignore: if those elements are children of elements to update they won't be updated
targeted_tags	targeted_tags affected tag names (by default: structure_shapes of the scui object)
append	if TRUE the value will be concatenate with the existing value

Value

The modified [bscui](#) object

Examples

```
###################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
    "examples", "Animal_cells.svg.gz",
    package="bscui"
))
info <- readr::read_tsv(system.file(
    "examples", "uniprot_cellular_locations.txt.gz",
    package="bscui"
), col_types=strrep("c", 6)) |>
    mutate(id = sub("-", "", `Subcellular location ID`))

###################################@
### Building the figure ----

figure <- bscui(svg) |>
    set_bscui_ui_elements(
        info |>
            mutate(
                ui_type = "selectable",
                title = Name
            ) |>
            select(id, ui_type, title)
    ) |>
    set_bscui_styles(
        info |>
            filter(Name == "Cytosol") |>
            mutate(fill = "#FF7F7F") |>
            select(id, fill)
    ) |>
    set_bscui_attributes(
        info |>
            filter(Name == "Cytoskeleton") |>
            mutate(display = "none") |>
            select(id, display)
    ) |>
    set_bscui_selection("SL0188") |>
    set_bscui_options(zoom_min=1, clip=TRUE)
figure

###################################@
```

```
### Saving the figure ----

if(interactive()){
    ## Temporary directory to save example file
    tdir <- tempdir()

    ## Interactive html file
    f_path <- file.path(tdir, "figure.html")
    figure |> htmlwidgets::saveWidget(file=f_path)
    cat(f_path)

    ## PNG image
    f_path <- file.path(tdir, "figure.png")
    figure |>
        set_bscui_options(show_menu = FALSE) |>
        export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

---

set_bscui_options   *Set options of bscui widget*

Description

Set options of bscui widget

Usage

```
set_bscui_options(
  widget,
  show_menu,
  menu_width,
  zoom_min,
  zoom_max,
  zoom_step,
  clip,
  default_png_scale,
  selection_color,
  selection_opacity,
  selection_width,
  hover_color,
  hover_opacity,
  hover_width,
  structure_shapes,
  dblclick_timeout,
  hover_timeout,
  width,
  height
)
```

Arguments

`widget`	a [bscui](#) object
`show_menu`	if TRUE (default) control menu will be available
`menu_width`	css width value (default: "30px")
`zoom_min`	smallest zoom value (default: 0.5)
`zoom_max`	largest zoom value (default: 20)
`zoom_step`	zooming step: the larger the faster (default: 1.1)
`clip`	if TRUE (default: FALSE), when the current zoom is 1, the viewBox is automatically set to its original state (the drawing cannot be moved)
`default_png_scale`	default value for scaling PNG export (default: 1)
`selection_color`	color used to highlight selection (default: "orange")
`selection_opacity`	opacity of selection highlight (default: 0.5)
`selection_width`	the additional stroke width to apply on selection (default: 4)
`hover_color`	a list of colors used to highlight hovered elements (default: `list(button="yellow"`, `selectable="cyan"`, `none="transparent")`)
`hover_opacity`	opacity of hovered highlight (default: 0.5)
`hover_width`	the additional stroke width to apply on hover (default: 4)
`structure_shapes`	SVG shapes to considered as concrete drawing (default: `c("rect"`, `"circle"`, `"ellipse"`, `"line"`, `"polyline"`, `"polygon"`, `"path")`: "text" excluded)
`dblclick_timeout`	minimum time in ms between 2 independant clicks (default: 250)
`hover_timeout`	time in ms before update hovered element (default: 100)
`width, height`	widget width: must be a valid CSS unit (like `'100\%'`, `'400px'`, `'auto'`) or a number, which will be coerced to a string and have `'px'` appended.

Value

The modified [bscui](#) object

Examples

```
#################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)
```

```
svg <- xml2::read_xml(system.file(
   "examples", "Animal_cells.svg.gz",
   package="bscui"
))
info <- readr::read_tsv(system.file(
   "examples", "uniprot_cellular_locations.txt.gz",
   package="bscui"
), col_types=strrep("c", 6)) |>
   mutate(id = sub("-", "", `Subcellular location ID`))

#################################@
### Building the figure ----

figure <- bscui(svg) |>
   set_bscui_ui_elements(
      info |>
         mutate(
            ui_type = "selectable",
            title = Name
         ) |>
         select(id, ui_type, title)
   ) |>
   set_bscui_styles(
      info |>
         filter(Name == "Cytosol") |>
         mutate(fill = "#FF7F7F") |>
         select(id, fill)
   ) |>
   set_bscui_attributes(
      info |>
         filter(Name == "Cytoskeleton") |>
         mutate(display = "none") |>
         select(id, display)
   ) |>
   set_bscui_selection("SL0188") |>
   set_bscui_options(zoom_min=1, clip=TRUE)
figure

#################################@
### Saving the figure ----

if(interactive()){
   ## Temporary directory to save example file
   tdir <- tempdir()

   ## Interactive html file
   f_path <- file.path(tdir, "figure.html")
   figure |> htmlwidgets::saveWidget(file=f_path)
   cat(f_path)

   ## PNG image
   f_path <- file.path(tdir, "figure.png")
   figure |>
```

```
        set_bscui_options(show_menu = FALSE) |>
        export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

---

set_bscui_selection *Pre-select UI elements in a bscui widget*

Description

Pre-select UI elements in a bscui widget

Usage

```
set_bscui_selection(widget, selected)
```

Arguments

widget	a [bscui](#) object
selected	identifiers of pre-selected identifiers

Value

The modified [bscui](#) object

Examples

```
##################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
    "examples", "Animal_cells.svg.gz",
    package="bscui"
))
info <- readr::read_tsv(system.file(
    "examples", "uniprot_cellular_locations.txt.gz",
    package="bscui"
), col_types=strrep("c", 6)) |>
    mutate(id = sub("-", "", `Subcellular location ID`))


##################################@
### Building the figure ----

figure <- bscui(svg) |>
```

```
    set_bscui_ui_elements(
        info |>
            mutate(
                ui_type = "selectable",
                title = Name
            ) |>
            select(id, ui_type, title)
    ) |>
    set_bscui_styles(
        info |>
            filter(Name == "Cytosol") |>
            mutate(fill = "#FF7F7F") |>
            select(id, fill)
    ) |>
    set_bscui_attributes(
        info |>
            filter(Name == "Cytoskeleton") |>
            mutate(display = "none") |>
            select(id, display)
    ) |>
    set_bscui_selection("SL0188") |>
    set_bscui_options(zoom_min=1, clip=TRUE)
figure

#################################@
### Saving the figure ----

if(interactive()){
    ## Temporary directory to save example file
    tdir <- tempdir()

    ## Interactive html file
    f_path <- file.path(tdir, "figure.html")
    figure |> htmlwidgets::saveWidget(file=f_path)
    cat(f_path)

    ## PNG image
    f_path <- file.path(tdir, "figure.png")
    figure |>
        set_bscui_options(show_menu = FALSE) |>
        export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

---

set_bscui_styles   *Set styles of elements of a bscui widget*

Description

Set styles of elements of a bscui widget

Usage

```
set_bscui_styles(
  widget,
  element_styles,
  to_ignore = NULL,
  targeted_tags = widget$x$structure_shapes,
  append = FALSE
)
```

Arguments

widget	a bscui object
element_styles	NULL or a data frame with an **id** column providing the element identifier and one column per style name. Column names should correspond to a style name in camel case (e.g., "strokeOpacity").
to_ignore	identifiers of elements to ignore: if those elements are children of elements to update they won't be updated
targeted_tags	targeted_tags affected tag names (by default: structure_shapes of the scui object)
append	if TRUE the value will be concatenate with the existing value

Value

The modified bscui object

Examples

```
###################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
   "examples", "Animal_cells.svg.gz",
   package="bscui"
))
info <- readr::read_tsv(system.file(
   "examples", "uniprot_cellular_locations.txt.gz",
   package="bscui"
), col_types=strrep("c", 6)) |>
   mutate(id = sub("-", "", `Subcellular location ID`))

###################################@
### Building the figure ----

figure <- bscui(svg) |>
   set_bscui_ui_elements(
```

```
        info |>
           mutate(
               ui_type = "selectable",
               title = Name
           ) |>
           select(id, ui_type, title)
    ) |>
    set_bscui_styles(
        info |>
           filter(Name == "Cytosol") |>
           mutate(fill = "#FF7F7F") |>
           select(id, fill)
    ) |>
    set_bscui_attributes(
        info |>
           filter(Name == "Cytoskeleton") |>
           mutate(display = "none") |>
           select(id, display)
    ) |>
    set_bscui_selection("SL0188") |>
    set_bscui_options(zoom_min=1, clip=TRUE)
figure

###################################@
### Saving the figure ----

if(interactive()){
    ## Temporary directory to save example file
    tdir <- tempdir()

    ## Interactive html file
    f_path <- file.path(tdir, "figure.html")
    figure |> htmlwidgets::saveWidget(file=f_path)
    cat(f_path)

    ## PNG image
    f_path <- file.path(tdir, "figure.png")
    figure |>
       set_bscui_options(show_menu = FALSE) |>
       export_bscui_to_image(file=f_path, zoom=2)
    cat(f_path)
}
```

set_bscui_ui_elements   *Set UI elements of a bscui widget*

Description

Set UI elements of a bscui widget

Usage

```
set_bscui_ui_elements(widget, ui_elements)
```

Arguments

widget   a [bscui](#) object

ui_elements   NULL or a data frame with the following columns:

- **id**: the element identifier
- **ui_type**: either "selectable" (several elements can be selected), "button" (action will be triggered on click), "none" (no ui)
- **title**: a description of the element to display on mouseover event

Value

The modified [bscui](#) object

Examples

```
##################################@
### Preparing data ----

library(bscui)
library(xml2)
library(readr)
library(dplyr)

svg <- xml2::read_xml(system.file(
   "examples", "Animal_cells.svg.gz",
   package="bscui"
))
info <- readr::read_tsv(system.file(
   "examples", "uniprot_cellular_locations.txt.gz",
   package="bscui"
), col_types=strrep("c", 6)) |>
   mutate(id = sub("-", "", `Subcellular location ID`))

##################################@
### Building the figure ----

figure <- bscui(svg) |>
   set_bscui_ui_elements(
      info |>
         mutate(
            ui_type = "selectable",
            title = Name
         ) |>
         select(id, ui_type, title)
   ) |>
   set_bscui_styles(
      info |>
```

```
        filter(Name == "Cytosol") |>
        mutate(fill = "#FF7F7F") |>
        select(id, fill)
   ) |>
   set_bscui_attributes(
       info |>
        filter(Name == "Cytoskeleton") |>
        mutate(display = "none") |>
        select(id, display)
   ) |>
   set_bscui_selection("SL0188") |>
   set_bscui_options(zoom_min=1, clip=TRUE)
figure

###################################@
### Saving the figure ----

if(interactive()){
   ## Temporary directory to save example file
   tdir <- tempdir()

   ## Interactive html file
   f_path <- file.path(tdir, "figure.html")
   figure |> htmlwidgets::saveWidget(file=f_path)
   cat(f_path)

   ## PNG image
   f_path <- file.path(tdir, "figure.png")
   figure |>
       set_bscui_options(show_menu = FALSE) |>
       export_bscui_to_image(file=f_path, zoom=2)
   cat(f_path)
}
```

---

update_bscui_attributes

Update the attributes of bscui elements in 'shiny' app

Description

Update the attributes of bscui elements in 'shiny' app

Usage

```
update_bscui_attributes(
  proxy,
  element_attributes,
  to_ignore = NULL,
  targeted_tags = NULL
)
```

Arguments

proxy   a `bscui_Proxy` object

element_attributes

         a data frame with an **id** column providing the element identifier and one column per attribute name.

to_ignore   of elements to ignore: if those elements are children of elements to update they won't be updated. This parameter is not taken into account when there is no "id" column in the element_styles data frame.

targeted_tags   affected tag names. If NULL (default), the structure_shapes of the `bscui` object

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

update_bscui_selection

Replace current selection with given element identifiers

Description

Replace current selection with given element identifiers

Usage

```
update_bscui_selection(proxy, element_ids)
```

Arguments

proxy   a `bscui_Proxy` object

element_ids   element identifiers to add to the selection; empty clear the selection

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
       "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

update_bscui_styles   *Update the style of bscui elements in 'shiny' app*

Description

Update the style of bscui elements in 'shiny' app

Usage

```
update_bscui_styles(
  proxy,
  element_styles,
  to_ignore = NULL,
  targeted_tags = NULL,
  append = FALSE
)
```

Arguments

proxy   a `bscui_Proxy` object

element_styles   a data frame with an "id" column and one column per style to apply. If the "id" column is missing, then the modifications apply to the svg selected elements.

to_ignore   of elements to ignore: if those elements are children of elements to update they won't be updated. This parameter is not taken into account when there is no "id" column in the element_styles data frame.

targeted_tags   affected tag names. If NULL (default), the structure_shapes of the `bscui` object

append   if TRUE the value will be concatenate with the existing value

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

---

update_bscui_ui_elements

Update the type and title of bscui ui elements in 'shiny' app

---

Description

Update the type and title of bscui ui elements in 'shiny' app

Usage

```
update_bscui_ui_elements(proxy, ui_elements)
```

Arguments

proxy   a bscui_Proxy object

ui_elements   NULL or a data frame with the following columns:

- **id**: the element identifier
- **ui_type**: either "selectable" (several elements can be selected), "button" (action will be triggered on click), "none" (no ui)
- **title**: a description of the element to display on mouseover event

Value

the provided proxy object

Examples

```
if(interactive()){
   from_shiny <- new.env()
   shiny::runApp(system.file(
      "examples", "shiny-anatomogram", package = "bscui"
   ))
   for(n in names(from_shiny)){
      bscui(from_shiny[[n]]) |> print()
   }
}
```

Index