Package ‘savonliquide’

February 22, 2021

Title  Accessibility Toolbox for ‘R’ Users
Version  0.2.0
Description  Provides a toolbox that allows the user to implement accessibility related concepts.
License  MIT + file LICENSE
Encoding  UTF-8
LazyData  true
RoxygenNote  7.1.1
Suggests  testthat, spelling
URL  https://github.com/feddelegrand7/savonliquide
BugReports  https://github.com/feddelegrand7/savonliquide/issues
Imports  glue, htmltools, httr, crayon
Language  en-US
NeedsCompilation  no
Author  Mohamed El Fodil Ihaddaden [aut, cre]
Maintainer  Mohamed El Fodil Ihaddaden <ihaddaden.fodeil@gmail.com>
Repository  CRAN
Date/Publication  2021-02-22 20:40:03 UTC

R topics documented:

add_description .................................................. 2
check_contrast .................................................... 3
check_contrast_raw .............................................. 4
create_invisible_anchor ....................................... 4
describe_using .................................................. 5
make_skiplinks ................................................... 6
make_tabable ...................................................... 7

Index  8
add_description

Add a description to an HTML element

Description

Add a description to an HTML element

Usage

add_description(element, descID, description, visible = FALSE)

Arguments

- element: an HTML element to describe
- descID: the ID of the div that will describe the HTML element
- description: the description of the HTML element
- visible: should the description be visible? Defaults to FALSE

Value

an HTML element with a description attached to it

Examples

if (interactive()) {
  ui <- fluidPage(
    h2("Using a screen reader
    hit <Tab> or <Shift + Tab> to
    navigate between the buttons
    and stop at button 5 to see the difference"),
    actionButton(
      inputId = "inp1",
      label = "button 1"
    ),
    actionButton(
      inputId = "inp2",
      label = "button 2"
    ),
    actionButton(
      inputId = "inp3",
      label = "button 3"
    ),
    actionButton(
      inputId = "inp4",
      label = "button 4"
    ),
check_contrast

$$\text{ActionButton(}
\text{inputId = "inp5",}
\text{label = "button 5"}
) \%>\%
$$

$$\text{add_description(}
\text{description = "hello this is a button}
\text{when you click it you'll have a}
\text{thing, when you don't click it you'll}
\text{have another thing",}
\text{descID = "chkoup"}
$$

$$)$$

$$\text{server <- function(input, output, session) {}}$$

$$\text{shinyApp(ui, server)}$$

---

**check_contrast**  | **Color Contrast Accessibility Report**

**Description**

returns a report from the Contrast Checker API about color contrast for accessibility

**Usage**

`check_contrast(fg_col, bg_col)`

**Arguments**

`fg_col` | the Foreground Color

`bg_col` | the Background Color

**Value**

Color Contrast Report

**Examples**

`check_contrast(fg_col = "#21EA06", bg_col = "#483D3D")`
check_contrast_raw  \hspace{1cm} \textit{Color Contrast Accessibility Report in a Raw Format}

\textbf{Description}

returns a report from the Contrast Checker API about color contrast for accessibility in a list format so that the information provided can be extracted and piped into other functions.

\textbf{Usage}

\begin{verbatim}
check_contrast_raw(fg_col, bg_col)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{fg\_col} \hspace{1cm} the Foreground Color
  \item \texttt{bg\_col} \hspace{1cm} the Background Color
\end{itemize}

\textbf{Value}

Color Contrast Report in a raw format

\textbf{Examples}

\begin{verbatim}
check_contrast_raw(fg\_col = \"#21EA06\", bg\_col = \"#483D3D\")
\end{verbatim}

create_invisible_anchor  \hspace{1cm} \textit{Create an HTML invisible anchor}

\textbf{Description}

Make an element invisible so that it can only be read by screen readers

\textbf{Usage}

\begin{verbatim}
create_invisible_anchor(id, text, href = NULL)
\end{verbatim}

\textbf{Arguments}

\begin{itemize}
  \item \texttt{id} \hspace{1cm} id of the anchor
  \item \texttt{text} \hspace{1cm} text of the anchor
  \item \texttt{href} \hspace{1cm} of the anchor. Defaults to NULL.
\end{itemize}
describe_using

Value
an invisible HTML anchor element

Description
Describe an HTML element by another one

Usage
describe_using(element, descID)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>element</td>
<td>the HTML element to describe</td>
</tr>
<tr>
<td>descID</td>
<td>one or a vector of many HTML elements' &lt;IDs&gt; that will be used to describe the 'element' parameter</td>
</tr>
</tbody>
</table>

Value
an HTML element described by another HTML element

Examples

```r
if (interactive()) {
  ui <- fluidPage(
    h2("Using a screen reader
        hit Tab and Shift + Tab to
        navigate between the buttons
        and stop at button 2 to see the difference"),

    div(
      id = "paragraph",
      p("The following paragraph tag will be used as a descriptor")
    ),

    actionButton(
      inputId = "inp1",
      label = "button 1"
    ),
    actionButton(
      inputId = "inp2",
      label = "button 2"
    )
  )
  describe_using(
    descID = "paragraph"
  )
}"
```
make_skiplinks

Transform an HTML element to a Skip Link

Description

Transform an HTML element to a Skip Link

Usage

make_skiplinks(element, skip_to, bg_color = "#002240", col = "#FFFFFF")

Arguments

- element: the element to use as a Skip Link
- skip_to: the HTML element to skip to
- bg_color: the background color of the element to use as a Skip Link
- col: the color of the element to use as a Skip Link

Value

a Skip Link HTML element

Examples

if (interactive()) {
  ui <- fluidPage(
    tags$a("do you want to be redirected to google.com ?", id = "skip-link"),
  )

  make_skiplinks(
    skip_to = "https://google.com",
    bg_color = "red",
    col = "white"
  ),

  h1("accessibility is not a detail")
  )

  server <- function(input, output, session) {}

  shinyApp(ui, server)
}
**make_tabable**

Make HTML elements tabable

**Usage**

```r
make_tabable(element, tab_index = 0)
```

**Arguments**

- `element`: the HTML element to be tabable (if not by default)
- `tab_index`: takes either 0, a negative or a positive value according to the required state of the element. 0 will make the element tabable with its relative order defined by the platform convention. A negative value will make the element untabable. A positive value will make the element tabable and its relative order defined by the provided value.

**Value**

a tabable HTML element

**Examples**

```r
if (interactive()) {
  ui <- fluidPage(
    textInput(inputId = "inp1", label = "input"),
    div(h1("Not tabable")) %>%
      make_tabable(tab_index = -1),
    div(h2("Tabable ! with priority")) %>%
      make_tabable(tab_index = 1),
    div(h2("Simply Tabable")) %>%
      make_tabable(tab_index = 0)
  )

  server <- function(input, output, session) {}

  shinyApp(ui = ui, server = server)
}
```
Index

add_description, 2
check_contrast, 3
check_contrast_raw, 4
create_invisible_anchor, 4
describe_using, 5
make_skiplinks, 6
make_tabable, 7