Package: browsertools (via r-universe)

October 4, 2024

Title Running Specific JavaScript functions from R

Version 0.2.0

Description This package provides a series of js handlers for use in shiny applications.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports htmltools (>= 0.4.0), shiny (>= 1.4.0), rlang (>= 0.4.7), purrr (>= 0.3.4), jsonlite (>= 1.7.0)

Repository https://davidruvolo51.r-universe.dev

RemoteUrl https://github.com/davidruvolo51/browsertools

RemoteRef HEAD

RemoteSha 93315c72467d6cb318a6835718c3c4d5a14e1af2

Contents

`add_css` *Add CSS*

Description

Adds a css class(es) to an element using id or classname.

Usage

```
add_css(elem, css)
```

Arguments

`elem`	the id or class name of an html element
`css`	a string or character array containing css classes

Value

Adds a css class(es) to an element using id or classname.

References

https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList/add

See Also

`remove_css()`, `toggle_css()`

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
       tags$style(
          ".blue-text {
            color: blue;
          }",
         ".font-size-lg {
            font-size: 150%;
         }"
       )
    ),
    tags$main(
       tags$h2("Add CSS Example"),
       tags$p(
         id = "my-text-example",
         "Click the button below to change the text color to blue"
       ),
       tags$button(
          id = "addCSS",
          class = "shiny-bound-input action-button",
          "Add CSS"
       )
    )
  )
  server <- function(input, output, session) {
     observeEvent(input$addCSS, {
       browsertools::add_css(
         elem = "#my-text-example",
         css = c("blue-text", "font-size-lg")
       )
     })
  }
  shinyApp(ui, server)
}
```

---

as_js_object as_js_object

Description

Restructure a data.frame to JavaScript Object

Usage

```
as_js_object(x)
```

Arguments

x	a data frame object

Value

Restructures a data.frame to JavaScript Object

See Also

console_log(), console_table()

Examples

```
if (interactive()) {
 library(shiny)
 ui <- tagList(
     browsertools::use_browsertools(),
     tags$p("Open the browser's dev console to view the data")
 )
 server <- function(input, output, session) {
     n <- 10
     df <- data.frame(
         group = sample(letters, n),
         x = rnorm(n),
         y = rnorm(n),
         z = rnorm(n)
     )
     browsertools::console_table(data = df)
  }
  shinyApp(ui, server)
}
```

---

console_error console_error

Description

Send an error message to the brower's console

Usage

```
console_error(message)
```

Arguments

message	a string containing an error to display

Value

Sends an error message to the browser's console

References

https://developer.mozilla.org/en-US/docs/Web/API/Console/error

See Also

console_warn(), console_log(), console_table()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Display an Error Message in the Dev Console"),
    tags$p("Open the browser's dev console and click the button below."),
    tags$button(
       id = "sendError",
       class = "shiny-bound-input action-button",
       "Send Error"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$sendError, {
        browsertools::console_error(
            message = "This is an error message"
        )
    })
  }
  shinyApp(ui, server)
}
```

---

console_log console_log

Description

Send general information to the browser's console. For example a small array of values or a non-error message. Use `console_error` and `console_warn` for displaying issues.

Usage

```
console_log(message)
```

Arguments

message	a message to display

Value

Outputs an object to the browser's console

References

https://developer.mozilla.org/en-US/docs/Web/API/Console/log

See Also

console_error(), console_table(), console_warn()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Display Message in the Dev Console"),
    tags$p("Open the browser's dev console and click the button below."),
    tags$button(
       id = "sendMessage",
       class = "shiny-bound-input action-button",
       "Send Message"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$sendMessage, {
        browsertools::console_log(
            message = "This is a message"
        )
    })
  }
  shinyApp(ui, server)
}
```

---

console_table console_table

Description

Display data in the browser's console as a data table. By default, this function transforms input data object as a JavaScript function using the function `as_js_object` (also included in this package). Alternatively, you can pass in a list object and set `to_json` to `FALSE`.

Usage

```
console_table(data, to_json = TRUE)
```

Arguments

data	an object to display in the browser (data.frame, etc.)
to_json	If TRUE, data will be converted to a JS object

Value

Outputs an object to the browser's console

References

https://developer.mozilla.org/en-US/docs/Web/API/Console/table

See Also

as_js_object(), console_log()

Examples

```
if (interactive()) {
 library(shiny)
 ui <- tagList(
     browsertools::use_browsertools(),
     tags$p("Open the browser's dev console")
 )
 server <- function(input, output, session) {
     n <- 10
     df <- data.frame(
         group = sample(letters, n),
         x = rnorm(n),
         y = rnorm(n),
         z = rnorm(n)
     )
     browsertools::console_table(data = df)
  }
  shinyApp(ui, server)
}
```

---

console_warn console_warn

Description

Send a warning message to the console

Usage

```
console_warn(message)
```

Arguments

message	a message to display

Value

Outputs a warning message to the console

References

\url{https://developer.mozilla.org/en-US/docs/Web/API/Console/warn}

See Also

console_error(), console_log(), console_table()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Display Warning Message in the Dev Console"),
    tags$p("Open the browser's dev console and click the button below."),
    tags$button(
       id = "sendWarning",
       class = "shiny-bound-input action-button",
       "Send Warning"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$sendWarning, {
        browsertools::console_warn(
            message = "This is a warning message"
        )
    })
  }
  shinyApp(ui, server)
}
```

---

debug debug

Description

Print JavaScript errors in the R console. This may be useful for debugging errors that may arise when using functions included in this pacakge. For example, if an element cannot be found using the add_css function, the debug function will print the corresponding error. (The correct selector for add_css is #txt.)

Usage

```
debug()
```

Value

Print JavaScript errors in the R console

See Also

print_elem()

Examples

```
if (interactive()) {
  ui <- tagList(
    tags$head(
      tags$style(
        ".blue-text {
          color: blue;
        }"
      )
    ),
    tags$h2("Browsertools demo"),
    tags$p(
      id = "txt",
      "JavaScript messages will print in the R Console"
    )
  )
  server <- function(input, output, session) {
    browsertools::debug()
    browsertools::add_css(
      elem = "#t",
      css = "blue-text"
    )
  }
  shinyApp(ui, server)
}
```

---

enable_attributes enable_attributes

Description

This function allows you to access the attributes of an html element as R objects. This may be useful if there are cases where you need an html attribute is important for running specific code. A potential case is rendering a dark and light chart depending on the CSS classes of an element. However, if you are using an HTML/JavaScript based library, chart styling should be done using CSS.

Usage

```
enable_attributes()
```

Details

This functions works by injecting an span element immediately after the target element. Using a custom input binding, the function looks for the parent element and restructures the html attributes into an object.

This function takes no arguments. To access the attributes in the shiny server, it is critical that the target element has an ID. Call the function after the last attribute.

Value

View and print attributes of an html element

See Also

print_elem()

Examples

```
if (interactive()) {
  ui <- tagList(
      browsertools::use_browsertools(),
      tags$p(
         id = "my-text-elem",
         class = "text-bold text-size-lg",
         `data-value` = "12345",
         enable_attributes(),
         "You can access this element's attributes in the server"
      ),
      tags$button(
          id = "getAttribs",
          class = "shiny-bound-input action-button",
          "Get Attributes"
      )
  )
  server <- function(input, output) {
       observeEvent(input$getAttribs, {
           print(input$`my-text-elem`)
       })
  }
  shinyApp(ui, server)
}
```

hidden hidden

Description

Hide an element or elements by default (i.e., at the time of page render)

Usage

```
hidden(...)
```

Arguments

...	Shiny tag element(s) to hide

Value

Hide elements by when rendered

Examples

```
if (interactive()) {
 hidden(tags$p("hello, world"), tags$p("this is a test"))
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Hidden Elements"),
    tags$p("This element is visible by default"),
    browsertools::hidden(
      tags$p(
        id = "myHiddenElement",
        "This element is hidden by default"
      )
    )
  )
  server <- function(input, output, session) {
  }
  shinyApp(ui, server)
}
```

hide_elem hide_elem

Description

Hides an html element by id or class name.

Usage

```
hide_elem(elem)
```

Arguments

elem	the id or class name of an html elem to hide

Value

hide an HTML element

See Also

show_elem(), toggle_elem(), hidden()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
      browsertools::use_browsertools(),
      tags$main(
          tags$h2("Hide Element Example"),
          tags$p(
              id = "my-text-example",
              "Click the button below to hide the message below."
          ),
          tags$button(
              id = "hideElement",
              class = "shiny-bound-input action-button",
              "Hide Element"
          ),
          tags$p(
              id = "myMessage",
              "Hide this message!"
          )
      )
  )
  server <- function(input, output, session) {
      observeEvent(input$hideElement, {
          browsertools::hide_elem(
              elem = "#myMessage"
```

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

---

inner_html inner_html

Description

Update the content of an element by id or class name

Usage

```
inner_html(elem, content, append = FALSE, delay = NULL)
```

Arguments

elem	the ID or class name of an html element
content	an html string, Shiny tag, or Shiny tag list
append	an option that appends value of string or replaces it
delay	an optional arg to add a brief pause before sending the content to the html element. Ideal for content that is rendered by the server. Input is time in milliseconds.

Value

Update the content of an element by id or class name

References

https://developer.mozilla.org/en-US/docs/Web/API/Element/innerHTML

See Also

inner_text(), insert_adjacent_html()

Examples

```
if (interactive()) {
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Inner Html Example"),
    tags$p("Click the button to update the following message."),
    tags$p(
      id = "textToUpdate",
```

```
      "[This text will be updated]"
    ),
    tags$button(
      id = "updateText",
      class = "shiny-bound-input action-button",
      "Update Text"
    )
  )
  server <- function(input, output, session) {
    browsertools::inner_html(
      elem = "#textToUpdate",
      content = tags$strong("This is a bold message!")
    )
  }
  shinyApp(ui, server)
}
```

---

inner_text inner_text

Description

Modify the text of an element

Usage

```
inner_text(elem, content, append = FALSE, delay = NULL)
```

Arguments

elem	an element to select (e.g., ID, class, tag, etc.)
content	content to insert
append	an option that appends value of string or replaces it
delay	an optional arg that adds a brief pause before sending the content to the html element. Ideal for content that is rendered server-side. Input time is in milliseconds.

Value

Modify the text of an element

References

https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/innerText

See Also

inner_html(), insert_adjacent_html()

Examples

```
if (interactive()) {
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Inner Html Example"),
    tags$p("Click the button to update the following message."),
    tags$p(
      id = "textToUpdate",
      "[This text will be updated]"
    ),
    tags$button(
      id = "updateText",
      class = "shiny-bound-input action-button",
      "Update Text"
    )
  )
  server <- function(input, output, session) {
    browsertools::inner_text(
      elem = "#textToUpdate",
      content = "This is a new message!"
    )
  }
  shinyApp(ui, server)
}
```

---

insert_adjacent_html insert_adjacent_html

Description

Create and render a new html element(s) using shiny tags. Content is added to the document using a reference container (i.e., body, div, etc.). Use the argument `position` to specify where the content should be added.

Usage

```
insert_adjacent_html(id, content, position = "beforeend")
```

Arguments

`id`	an id of the parent element to render new elements into
`content`	An html string, shiny tag, or shiny tag list
`position`	a position relative to the reference element (i.e., `beforebegin`, `afterbegin`, `beforeend`, or `afterend`)

Value

Create a new html element using shiny tags as a child of an element

References

https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentHTML

See Also

inner_text(), inner_html()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$main(
      id = "main",
      tags$h1("Inner Adjacent HTML Example")
    )
  )
  server <- function(input, output, session) {
  insert_adjacent_html(
    id = "main",
    content = tagList(
        tags$h2("Hello, world!"),
        tags$p("How are you doing today?")
    ),
    position = "afterbegin"
  )
}
}
```

---

print_elem print_elem

Description

Find and print an HTML element in the R console. This function is designed to be used during application development rather than production applications as the corresponding JavaScript does not create a new instance when called. Use this function to debug the HTML markup of elements or simply to view a structure of an element. This may be useful when applications are running the viewer pane rather than in the browser.

Usage

```
print_elem(elem)
```

Arguments

elem	an element to select (i.e., ID, class, tag, etc.)

Value

Find and print an HTML element in the R console.

See Also

debug()

Examples

```
if (interactive()) {
   ui <- tagList(
       browsertools::use_browsertools(),
       tags$div(
         id = "mydiv",
         tags$h2("My Element"),
         tags$p(
           "This is an example element. The structure will be printed",
           "in the R console."
         )
       )
   )
   server <- function(input, output) {
     observe({
        print_elem(elem = "#mydiv")
     })
   }
}
```

---

refresh_page refresh_page

Description

Trigger a page refresh

Usage

```
refresh_page()
```

Value

Trigger a page refresh

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
     browsertools::refresh_page(),
     tags$h2("Refresh App Example"),
     tags$p("Click the button to refresh the app"),
     tags$button(
         id = "refreshApp",
         class = "shiny-bound-input action-button",
         "Resfresh App"
     )
  )
  server <- function(input, output, session) {
    observeEvent(input$refreshApp, {
      browsertools::refresh_page()
    })
  }
  shinyApp(ui, server)
}
```

`remove_css` `remove_css`

Description

Removes a css class(es) to an element using id or classname.

Usage

```
remove_css(elem, css)
```

Arguments

`elem`	the id or class name of an html element
`css`	a string or character array containing css classes

Value

Removes a css class(es) to an element using id or classname.

References

https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList/remove

See Also

`add_css()`, `toggle_css()`

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
       tags$style(
          ".blue-text {
            color: blue;
          }"
       )
    ),
    tags$main(
       tags$h2("Remove CSS Example"),
       tags$p(
         id = "my-text-example",
         class = "blue-text",
         "Click the button below to remove the blue color from the text"
       ),
       tags$button(
          id = "removeCSS",
          class = "shiny-bound-input action-button",
          "Remove CSS"
       )
    )
  )
  server <- function(input, output, session) {
     observeEvent(input$removeCSS, {
       browsertools::remove_css(
         elem = "#my-text-example",
         css = "blue-text"
       )
     })
  }
  shinyApp(ui, server)
}
```

`remove_element` `remove_element`

Description

Removes an html element from the DOM

Usage

```
remove_element(elem)
```

Arguments

`elem`	an ID of the element to remove

Value

Removes an html element from the DOM

See Also

inner_html(), insert_adjacent_html()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Remove Element Example"),
    tags$p(
      id = "textToRemove",
      "Click the button below to remove this message."
    ),
    tags$button(
      id = "removeMessage",
      class = "shiny-bound-input action-button",
      "Remove Message"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$removeMessage, {
      browsertools::remove_element(
        elem = "#textToRemove"
      )
    })
  }
  shinyApp(ui, server)
}
```

---

remove_element_attribute

remove_element_attribute

Description

Removes an html attribute from an element

Usage

```
remove_element_attribute(elem, attr)
```

Arguments

elem	an ID of the html element to be modified
attr	the name of the attribute to remove

Value

Remove an attribute from an html element

See Also

set_element_attribute(), print_elem()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
      tags$style(
        "[aria-hidden='true'] {
             position: absolute;
             width: 1px;
             height: 1px;
             margin: -1px;
             clip: rect(0, 0, 0, 0);
             clip: rect(0 0 0 0);
             overflow: hidden;
             white-space: nowrap;
        }"
      )
    ),
    tags$h2("Remove Element Attribute Example"),
    tags$p(
      "Click the button below to remove an attribute of a hidden element"
    ),
    tags$button(
      id = "removeAttr",
      class = "shiny-bound-input action-button",
      "Remove Attribute"
    ),
    tags$p(
      id = "hidden-text",
      `aria-hidden` = "true",
      "This is a hidden element"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$removeAttr, {
      browsertools::remove_element_attribute(
        elem = "#hidden-text",
```

```
        attr = "aria-hidden"
      )
    })
  }
  shinyApp(ui, server)
}
```

---

scroll_to scroll_to

Description

Scrolls the window to the top of the page, user defined coordinates, or a specific element. The default behavior of this function is to scroll to the top of the page (x: 0, y: 0). You can also use an element's selector path instead.

Usage

```
scroll_to(x = 0, y = 0, elem = NULL)
```

Arguments

x	amount (in pixels) to scroll along the horizontal axis starting from the top left (default: 0)
y	amount (in pixels) to scroll along the vertical axis starting from the top left (default: 0)
elem	a selector path of an element that you would like to scroll to (i.e., id, class, tag, etc.). Using elem will override any coordinates.

Value

Scrolls the window to the top of the page or a user defined coordinates

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
        tags$style(
            "#spacing{
                height: 1200px;
              }"
        )
    ),
    tags$h2("Document Scrolling"),
    tags$p("Click the button at the bottom of the page"),
```

```
    tags$div(
        id = "spacing",
        `aria-hidden` = "true"
    ),
    tags$button(
       id = "appScroll",
       class = "shiny-bound-input action-button",
       "Scroll to Top"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$appScroll, {
        browsertools::scroll_to()
    })
  }
  shinyApp(ui, server)
}
```

`set_document_title` `set_document_title`

Description

Set or change the document title

Usage

```
set_document_title(title, append = FALSE)
```

Arguments

`title`	a string containing a title of the document
`append`	if TRUE the title will be added to the current title

Value

Set or change the document title

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Setting the Document Title"),
    tags$p("Enter a new title for the document"),
    tags$input(
      id = "titleInput",
      class = "shiny-bound-input",
```

```
      type = "text"
    ),
    tags$button(
       id = "submit",
       class = "shiny-bound-input action-button",
       "Submit"
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$submit, {
       browsertools::set_document_title(
         title = as.character(input$titleInput)
       )
    })
  }
  shinyApp(ui, server)
}
```

`set_element_attribute` `set_element_attribute`

Description

Set an attribute of an html element

Usage

```
set_element_attribute(elem, attr, value)
```

Arguments

`elem`	the id or class name of an html element
`attr`	the name of the attribute to update
`value`	the value to add to the attribute

Value

Set an attribute of an html element

See Also

`remove_element_attribute()`, `print_elem()`

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
      tags$style(
        "[aria-hidden='true'] {
             position: absolute;
             width: 1px;
             height: 1px;
             margin: -1px;
             clip: rect(0, 0, 0, 0);
             clip: rect(0 0 0 0);
             overflow: hidden;
             white-space: nowrap;
        }"
      )
    ),
    tags$h2("Set Element Attribute Example"),
    tags$p(
      "Click the button below to set an attribute of an element"
    ),
    tags$button(
      id = "setAttr",
      class = "shiny-bound-input action-button",
      "Set Attribute"
    ),
    tags$p(
      id = "mytext",
      `aria-hidden` = "true",
      "This element will be modified."
    )
  )
  server <- function(input, output, session) {
    observeEvent(input$removeAttr, {
      browsertools::set_element_attribute(
        elem = "#mytext",
        attr = "aria-hidden",
        value = "true"
      )
    })
  }
  shinyApp(ui, server)
}
```

---

show_elem show_elem

Description

Shows an html element by id or class name.

Usage

```
show_elem(elem)
```

Arguments

elem	the id or class name of an html elem

Value

Show a hidden html element

See Also

hide_elem(), hidden(), toggle_elem()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
      browsertools::use_browsertools(),
      tags$main(
          tags$h2("Show Element Example"),
          tags$p(
              id = "my-text-example",
              "Click the button below to show a hidden element."
          ),
          tags$button(
              id = "showElement",
              class = "shiny-bound-input action-button",
              "Show Element"
          ),
          browsertools::hidden(
              tags$p(
                  id = "hiddenMessage",
                  "You found the hidden message!"
              )
          )
      )
  )
  server <- function(input, output, session) {
      observeEvent(input$showElement, {
          browsertools::show_elem(
              elem = "#hiddenMessage"
          )
      })
  }
  shinyApp(ui, server)
```

```
}
```

---

toggle_css toggle_css

Description

Toggles a css state of an html element

Usage

```
toggle_css(elem, css)
```

Arguments

elem	the id or class name of an html element
css	a string or character array containing css classes

Value

Toggles a css state of an html element

References

https://developer.mozilla.org/en-US/docs/Web/API/DOMTokenList/toggle

See Also

remove_css(), add_css()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$head(
       tags$style(
          ".blue-text {
            color: blue;
          }"
       )
    ),
    tags$main(
       tags$h2("Toggle CSS Example"),
       tags$p(
         id = "my-text-example",
         "Click the button below to toggle the text color."
```

```
        ),
        tags$button(
           id = "toggleCSS",
           class = "shiny-bound-input action-button",
           "Toggle CSS"
        )
      )
    )
    server <- function(input, output, session) {
       observeEvent(input$toggleCSS, {
         browsertools::toggle_css(
           elem = "#my-text-example",
           css = "blue-text"
         )
       })
    }
    shinyApp(ui, server)
  }
```

---

toggle_elem toggle_elem

Description

Toggle the visibility of an html element

Usage

```
toggle_elem(elem)
```

Arguments

elem	the id or class name of an html element

Value

Toggle the visibility of an html element

See Also

hide_elem(), show_elem()

Examples

```
if (interactive()) {
  library(shiny)
  ui <- tagList(
      browsertools::use_browsertools(),
      tags$main(
```

```
            tags$h2("Toggle Element Example"),
            tags$p(
                id = "my-text-example",
                "Click the button below to toggle the hidden element."
            ),
            tags$button(
                id = "toggleElement",
                class = "shiny-bound-input action-button",
                "Toggle Element"
            ),
            browsertools::hidden(
                tags$p(
                    id = "hiddenMessage",
                    "You found the hidden message!"
                )
            )
        )
    )
    server <- function(input, output, session) {
        observeEvent(input$toggleElement, {
            browsertools::toggle_elem(
                elem = "#hiddenMessage"
            )
        })
    }
    shinyApp(ui, server)
}
```

`use_browsertools` `use_browsertools`

Description

Function that loads the browsertools JavaScript file into your shiny app. This function is required and should be called at the top of the Shiny app. If you are using other ShinyUI layouts, you may need to wrap your app in `tagList`. (The size of the js file is approximately 5.1kb.)

Usage

```
use_browsertools()
```

Value

Function that loads all assets into your shiny app

Examples

```
if (interactive()) {
  ui <- tagList(
    browsertools::use_browsertools(),
    tags$h2("Hello, world!")
  )
  server <- function(input, output, session) {
  }
  shinyApp(ui, server)
}
```

Index