Package: shinyEvents (via r-universe)

September 6, 2024

Type Package

Title Shiny wrapper with event handlers instead of reactivity

Version 2.3

Date 2018-02-28

Author Sebastian Kranz

Maintainer Sebastian Kranz <sebastian.kranz@uni-ulm.de>

Description Shiny is a great package. Yet, for more complex projects,
with much dynamic UI, I find it hard to write clean code with
its reactivity paradigm. This package wraps shiny into a more
traditional programming approach: - event handlers for input
changes and button clicks - explicit update functions to change
outputs - no explicit server function

License GPL >= 2.0

Depends shiny, restorepoint

RoxygenNote 5.0.0

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

RemoteUrl https://github.com/skranz/shinyEvents

RemoteRef master

RemoteSha 092ea663d7d23b731e26eeb3b865afbde6c513f0

Contents

aceHotkeyHandler *Add an handler to a hotkey in an aceEditor component*

Description

Add an handler to a hotkey in an aceEditor component

Usage

```
aceHotkeyHandler(id, fun, ..., app = getApp(),
  if.handler.exists = c("replace", "add", "skip")[1],
  session = getAppSession(app), no.authentication.required = FALSE)
```

Arguments

`id`	name of the button
`fun`	function that will be called if button is pressed. The function will be called with the following arguments: keyId: the id assigned to the hotkey editorId: the id of the aceEditor widget selection: if a text is selected, this selection text: the text of the aceEditor widget cursor: a list with the current cursor position: row and column with index starting with 0 session: the current session object
`...`	extra arguments that will be passed to fun when the event is triggered.

appendToHTML *Append HTML code to a DOM element*

Description

Append HTML code to a DOM element

Usage

```
appendToHTML(html, selector = "body", app = getApp())
```

appInitHandler — *Set a function that will be called when a new session of an app is initialized.*

Description

@param initHandler a function that takes parameters session, input, output and app. It will be called from the app$server function whenever a new session is created. It allows to initialize session specific variables and e.g. store them within the app object. The passed app object is already the local copy created for the new session.

Usage

```
appInitHandler(initHandler, app = getApp())
```

appReadyToRun — *set the app ready to run*

Description

set the app ready to run

Usage

```
appReadyToRun(app = getApp(), ui = app$ui)
```

bottomScript — *If app is not running, mark script to be added at the bottom and return NULL If app is already running return script directly*

Description

If app is not running, mark script to be added at the bottom and return NULL If app is already running return script directly

Usage

```
bottomScript(..., app = getApp())
```

buttonHandler — *A more efficient version of button handler via global eventId handler*

Description

A more efficient version of button handler via global eventId handler

Usage

```
buttonHandler(id, fun, ..., eventId = "buttonHandlerEvent",
  jscript = buttonHandlerJS(eventId), app = getApp(),
  no.authentication.required = FALSE)
```

Arguments

`id`	buttonId
`fun`	the handler fun that will be called when the button is clicked
`...`	additional arguments passed to the handler fun

callJS — *Call a javascript function or method with R arguments*

Description

Call a javascript function or method with R arguments

Usage

```
callJS(.fun, ..., .args = NULL, .app = getApp())
```

changeHandler — *Add an handler to an input that is called when the input value changes*

Description

Add an handler to an input that is called when the input value changes

Usage

```
changeHandler(id, fun, ..., app = getApp(), on.create = FALSE,
  if.handler.exists = c("replace", "add", "skip")[1],
  session = getAppSession(app), no.authentication.required = FALSE)
```

Arguments

`id`	name of the input element
`fun`	function that will be called if the input value changes. The function will be called with the arguments: 'id', 'value' and 'session'. One can assign the same handler functions to several input elements.
`...`	extra arguments that will be passed to fun when the event is triggered.

`classEventHandler` *An event handler for objects with given class*

Description

An event handler for objects with given class

Usage

```
classEventHandler(class, fun, event = "change", css.locator = "",
  inner.js.code = NULL, shiny.value.code = NULL, eventId = paste0(class,
  "_class_", event, "_event"), class.prefix = ".", stop.propagation = FALSE,
  ...)
```

`customEventHandler` *A custom event handler. Need to write correct css.locator*

Description

A custom event handler. Need to write correct css.locator

Usage

```
customEventHandler(eventId, fun, css.locator, event = "change",
  inner.js.code = NULL, shiny.value.code = NULL,
  extra.shiny.value.code = "", id = NULL, stop.propagation = FALSE, ...)
```

`dsetUI` *Directly setUI , also works for hidden UI*

Description

Directly setUI , also works for hidden UI

Usage

```
dsetUI(id, ui, selector = paste0("#", id, collapse = ", "), app = getApp(),
  ...)
```

`evalJS` *Evaluate arbitrary java script code in the client's web browser*

Description

Evaluate arbitrary java script code in the client's web browser

Usage

```
evalJS(js, ..., .args = list(...), app = getApp())
```

`eventsApp` *Generate an empty shiny events app*

Description

Generate an empty shiny events app

Usage

```
eventsApp(set.as.default = TRUE, verbose = TRUE, single.instance = FALSE,
  add.events = getDefaultAppEvents(), no.events = FALSE,
  need.authentication = FALSE, adapt.ui = TRUE)
```

`getApp` *get the current app object*

Description

If the app is already running, gets by default the local app copy corresponding to the current session

Usage

```
getApp(session = NULL)
```

`getAppSession` *Get the session associated with the app object*

Description

Get the session associated with the app object

Usage

```
getAppSession(app = NULL)
```

`getCurrentSession` *Get the current session object*

Description

Get the current session object

Usage

```
getCurrentSession()
```

`getInputValue` *Get an input value from the current session*

Description

Get an input value from the current session

Usage

```
getInputValue(id, session = getAppSession(app), app = getApp())
```

`hasWidgetValueChanged` *Checks whether the value of an input item has been changed (internal function)*

Description

Checks whether the value of an input item has been changed (internal function)

Usage

```
hasWidgetValueChanged(id, new.value, on.create = FALSE, app = getApp())
```

`idEventHandler` *An event handler for an object with given id*

Description

An event handler for an object with given id

Usage

```
idEventHandler(id, fun, event = "change", css.locator = "",
  inner.js.code = NULL, shiny.value.code = NULL, eventId = paste0(id,
  "_id_", event, "_event"), stop.propagation = FALSE, ...)
```

`ids2sel` *Transform a vector of ids to a jQuery selector string*

Description

Transform a vector of ids to a jQuery selector string

Usage

```
ids2sel(ids)
```

`imageClickHandler` *Handler for an image click*

Description

Handler for an image click

Usage

```
imageClickHandler(id, fun, ..., eventId = "imageClickEvent", app = getApp(),
  no.authentication.required = FALSE)
```

Arguments

`id`	id of the HTML img object
`fun`	the handler fun that will be called when the image is clicked
`...`	additional arguments passed to the handler fun

`initialQueryDispatch` *Can be called inside initApp handler fun is a function that gets an argument query and can do some initial dispatch depending on the query. For some reason we need to use the observer trick to get access to the query object. This means dispatch takes place after other commands in the initApp handler.*

Description

Can be called inside initApp handler fun is a function that gets an argument query and can do some initial dispatch depending on the query. For some reason we need to use the observer trick to get access to the query object. This means dispatch takes place after other commands in the initApp handler.

Usage

```
initialQueryDispatch(fun, app = getApp(), ...)
```

`moveBottomScripts` *Given a tag object, extract out any children of tags$head and return them separate from the body.*

Description

Given a tag object, extract out any children of tags$head and return them separate from the body.

Usage

```
moveBottomScripts(ui, reset.app = FALSE)
```

`prependToHTML` *Prpend HTML code to a DOM element*

Description

Prpend HTML code to a DOM element

Usage

```
prependToHTML(html, selector = "body", app = getApp())
```

`runEventsApp` *run shiny events app*

Description

run shiny events app

Usage

```
runEventsApp(app = getApp(), ui = NULL, ...)
```

`selectChangeHandler` *Add an handler to an input or select that is called when the input value changes*

Description

Add an handler to an input or select that is called when the input value changes

Usage

```
selectChangeHandler(id, fun, ..., eventId = "selectChangeHandlerEvent",
  jscript = selectChangeHandlerJS(eventId), app = getApp())
```

Arguments

`id`	name of the input element
`fun`	function that will be called if the input value changes. The function will be called with the arguments: 'id', 'value' and 'session'. One can assign the same handler functions to several input elements.
`...`	extra arguments that will be passed to fun when the event is triggered.

`setApp` *set the current app*

Description

set the current app

Usage

```
setApp(app)
```

`setAppUI` *set the main ui object for the app*

Description

set the main ui object for the app

Usage

```
setAppUI(ui, app = getApp())
```

`setDataTable` *Update an dataTableOutput object.*

Description

Can be used instead of renderDataTable. Similar to updateDataTable but no need to provide session object

Usage

```
setDataTable(id, value, app = getApp(), ...)
```

`setDownloadHandler` *Shiny events version of downloadHandler*

Description

Shiny events version of downloadHandler

Usage

```
setDownloadHandler(id, filename, content, contentType = NA, ...,
  app = getApp())
```

Arguments

`id`	name of the downloadButton or downloadLink
`filename`	A string of the filename, including extension, that the user's web browser should default to when downloading the file; or a function that returns such a string.
`content`	A function that takes a single argument file that is a file path (string) of a nonexistent temp file, and writes the content to that file path.
`contentType`	A string of the download's content type, for example "text/csv" or "image/png". If NULL or NA, the content type will be guessed based on the filename extension, or application/octet-stream if the extension is unknown.

setHtmlAttribute *Set attributes of HTML elements*

Description

Set attributes of HTML elements

Usage

```
setHtmlAttribute(id = NULL, attr, class = NULL,
  selector = paste0(c(sc("#", id), sc(".", class)), collapse = ", "),
  app = getApp())
```

setHtmlCSS *Set css style of HTML elements*

Description

Set css style of HTML elements

Usage

```
setHtmlCSS(id = NULL, attr, class = NULL, selector = paste0(c(sc("#", id),
  sc(".", class)), collapse = ", "), app = getApp())
```

setHtmlHide *Hide HTML elements*

Description

Hide HTML elements

Usage

```
setHtmlHide(id = NULL, class = NULL, display = "none",
  selector = paste0(c(sc("#", id), sc(".", class)), collapse = ", "))
```

setHtmlShow *Show HTML elements*

Description

Show HTML elements

Usage

```
setHtmlShow(id = NULL, class = NULL, display = "block",
  visibility = "visible", selector = paste0(c(sc("#", id), sc(".", class)),
  collapse = ", "))
```

setImage *Update an output object. Can be used instead of renderImage*

Description

Similar to updateImage but no need to provide session object

Usage

```
setImage(id, value, app = getApp(), ...)
```

setPlot *update an plotOutput object. Can be used instead of renderPlot.*

Description

Similar to updatePlot but no need to provide session object

Usage

```
setPlot(id, expr, app = getApp(), update.env = parent.frame(),
  quoted = FALSE, ...)
```

setPrint — *Update an textOutput object. Can be used instead of renderPrint*

Description

Similar to updatePrint but no need to provide session object

Usage

```
setPrint(id, expr, app = getApp(), ...)
```

setRHandsontable — *Update an RHandsontable object. Can be used instead of renderRHandsontable*

Description

Similar to updateRHandsontable but no need to provide session object

Usage

```
setRHandsontable(id, value, app = getApp(), ...)
```

setTable — *Update an tableOutput object. Can be used instead of renderTable*

Description

Similar to updateTable but no need to provide session object

Usage

```
setTable(id, value, app = getApp(), ...)
```

setText — *Update an textOutput object. Can be used instead of renderText*

Description

Similar to updateText but no need to provide session object

Usage

```
setText(id, value, app = getApp(), ...)
```

setUI — *Update an uiOutput object. Can be used instead of renderUI*

Description

Similar to updateUI but no need to provide session object

Usage

```
setUI(id, ui, app = getApp(), ...)
```

singletonBottomScript — *If app is not running, mark script to be added at the bottom and return NULL If app is already running return script directly*

Description

If app is not running, mark script to be added at the bottom and return NULL If app is already running return script directly

Usage

```
singletonBottomScript(..., app = getApp())
```

svgClickHandler — *Handler for an image click*

Description

Handler for an image click

Usage

```
svgClickHandler(id, fun, ..., eventId = if (stop.propagation) "svgClickEvent"
  else "svgClickEventWithPropagation", class = "clickable_svg",
  app = getApp(), no.authentication.required = FALSE,
  stop.propagation = TRUE)
```

Arguments

`id`	id of the HTML img object
`fun`	the handler fun that will be called when the image is clicked
`...`	additional arguments passed to the handler fun

`timerHandler` *Add an handler that triggers every intervalMs milliseconds*

Description

Add an handler that triggers every intervalMs milliseconds

Usage

```
timerHandler(id, intervalMs, fun, ..., app = getApp(), on.create = FALSE,
  if.handler.exists = c("replace", "add", "skip")[1], verbose = FALSE,
  session = getAppSession(app))
```

Arguments

`id`	name of the input element
`fun`	function that will be called if the input value changes. The function will be called with the arguments: 'id', 'value' and 'session'. One can assign the same handler functions to several input elements.
`...`	extra arguments that will be passed to fun when the event is triggered.

`updateDataTable` *Update an dataTableOutput object. Can be used instead of render-DataTable*

Description

Update an dataTableOutput object. Can be used instead of renderDataTable

Usage

```
updateDataTable(session = NULL, id, value, app = getApp(session), ...)
```

updateDownloadHandler *Shiny events version of downloadHandler*

Description

Shiny events version of downloadHandler

Usage

```
updateDownloadHandler(session = NULL, id, filename, content,
  contentType = NA, ..., app = getApp(session))
```

Arguments

id	name of the downloadButton or downloadLink
filename	A string of the filename, including extension, that the user's web browser should default to when downloading the file; or a function that returns such a string.
content	A function that takes a single argument file that is a file path (string) of a nonexistent temp file, and writes the content to that file path.
contentType	A string of the download's content type, for example "text/csv" or "image/png". If NULL or NA, the content type will be guessed based on the filename extension, or application/octet-stream if the extension is unknown.

updateImage *Update an output object. Can be used instead of renderImage*

Description

Update an output object. Can be used instead of renderImage

Usage

```
updateImage(session = NULL, id, value, app = getApp(session), ...)
```

updatePlot *update an plotOutput object. Can be used instead of renderPlot.*

Description

update an plotOutput object. Can be used instead of renderPlot.

Usage

```
updatePlot(session = NULL, id, expr, app = getApp(session),
  update.env = parent.frame(), quoted = FALSE)
```

updatePrint — *Update an textOutput object. Can be used instead of renderPrint*

Description

Update an textOutput object. Can be used instead of renderPrint

Usage

```
updatePrint(session = NULL, id, expr, app = getApp(session), ...)
```

updateRHandsontable — *Update an RHandsontable object. Can be used instead of render-RHandsontable*

Description

Update an RHandsontable object. Can be used instead of renderRHandsontable

Usage

```
updateRHandsontable(session = NULL, id, value, app = getApp(session), ...)
```

updateTable — *Update an tableOutput object. Can be used instead of renderTable*

Description

Update an tableOutput object. Can be used instead of renderTable

Usage

```
updateTable(session = NULL, id, value, app = getApp(session), ...)
```

updateText — *Update an textOutput object. Can be used instead of renderText*

Description

Update an textOutput object. Can be used instead of renderText

Usage

```
updateText(session = NULL, id, value, app = getApp(session), ...)
```

`updateUI` *Update an uiOutput object. Can be used instead of renderUI*

Description

Update an uiOutput object. Can be used instead of renderUI

Usage

```
updateUI(session, id, ui, app = getApp(session), ...)
```

`viewApp` *view shiny events app in RStudio viewer*

Description

view shiny events app in RStudio viewer

Usage

```
viewApp(app = getApp(), ui = NULL, launch.browser = rstudio::viewer, ...)
```

`wasAceHotkeyPressed` *Checks whether a button has been pressed again (internal function)*

Description

Checks whether a button has been pressed again (internal function)

Usage

```
wasAceHotkeyPressed(keyId, value, app = getApp())
```

Index