Package: keys (via r-universe)

September 13, 2024

Title Keyboard Shortcuts for 'shiny'

Version 0.1.1.9000

Description Assign and listen to keyboard shortcuts in 'shiny' using the 'Mousetrap' Javascript library.

License Apache License (>= 2)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

Imports htmltools, shiny, jsonlite

URL https://github.com/r4fun/keys

BugReports https://github.com/r4fun/keys/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://r4fun.r-universe.dev

RemoteUrl https://github.com/r4fun/keys

RemoteRef HEAD

RemoteSha e1f77a03891ecdec7dbe2796f99dea4f342e6ebc

Contents

addKeys *Add a key binding from the server side*

Description

Add a key binding from the server side

Usage

```
addKeys(inputId, keys, session = shiny::getDefaultReactiveDomain())

removeKeys(keys, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	The input slot that will be used to access the value.
keys	A character vector of keys to bind. Examples include, `command`, `command+shift+a`, `up down left right`, and more.
session	The `session` object passed to function given to `shinyServer`. Default is `getDefaultReactiveDomain()`

keysInput *Create a keys input control*

Description

Create a key input that can be used to observe keys pressed by the user.

Usage

```
keysInput(inputId, keys, global = FALSE)
```

Arguments

inputId	The input slot that will be used to access the value.
keys	A character vector of keys to bind. Examples include, `command`, `command+shift+a`, `up down left right`, and more.
global	Should keys work anywhere? If TRUE, keys are triggered when inside a textInput.

Examples

```
## Not run:
ui <- fluidPage(
  keysInput("keys", c(
    "1",
    "2",
    "3",
    "command+shift+k",
    "up up down down left right left right b a enter"
  )),
)

server <- function(input, output, session) {
  observeEvent(input$keys, {
    print(input$keys)
  })
}

shinyApp(ui, server)

## End(Not run)
```

---

keysRecordInput *Create a keys recorder input control*

Description

Create a key input that can be used to record keys pressed by the user.

Usage

```
keysRecordInput(inputId)

recordKeys(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	The input slot that will be used to access the value.
session	The session object passed to function given to shinyServer. Default is getDefaultReactiveDomain()

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    useKeys(),
    keysRecordInput("recorder"),
```

```
  keysInput("keys", "command+shift+k"),
  actionButton("record", "Record keys")
)

server <- function(input, output, session) {
  observeEvent(input$record, {
    print("recording keys...")
    recordKeys("recorder")
  })
  observeEvent(input$recorder, {
    print("adding keys...")
    addKeys("keys", input$recorder)
  })
  observeEvent(input$keys, {
    print(input$keys)
  })
}

shinyApp(ui, server)
}
```

---

pauseKey *Pause or Unpause Keys*

Description

These functions allow to pause and unpause keyboard watching

Usage

```
pauseKey(session = shiny::getDefaultReactiveDomain())

unpauseKey(session = shiny::getDefaultReactiveDomain())
```

Arguments

session	The session object passed to function given to shinyServer. Default is getDefaultReactiveDomain()

Examples

```
if (interactive()){
library(shiny)
ui <- fluidPage(
  useKeys(),
  keysInput("keys", letters),
  actionButton("pause", "Pause"),
  actionButton("unpause", "Unpause")
)

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

```
  observeEvent(input$keys, {
    print(input$keys)
  })
  observeEvent(input$pause, {
    pauseKey()
  })
  observeEvent(input$unpause, {
    unpauseKey()
  })

}

shinyApp(ui, server)
}
```

---

useKeys *Use Keys in your application*

Description

This function adds the keys dependencies to your application

Usage

```
useKeys()
```

Value

An html singleton

Index