Package 'shinyAce'

January 3, 2019

Type Package

Title Ace Editor Bindings for Shiny

Version 0.3.3

Date 2018-12-25

Description Ace editor bindings to enable a rich text editing environment within Shiny.

License MIT + file LICENSE

Depends R (>= 3.4.0)

Imports shiny (>= 1.0.5), jsonlite

Suggests testthat (>= 2.0.0), dplyr (>= 0.7.4)

BugReports https://github.com/trestletech/shinyAce/issues

Encoding UTF-8

RoxygenNote 6.1.0

NeedsCompilation no

Author Vincent Nijs [aut, cre],
Forest Fang [aut],
Trestle Technology, LLC [aut],
Jeff Allen [aut],
Institut de Radioprotection et de Surete Nucleaire [cph],
Ajax.org B.V. [ctb, cph] (Ace)

Maintainer Vincent Nijs <radiant@rady.ucsd.edu>

Repository CRAN

Date/Publication 2019-01-03 09:30:07 UTC

R topics documented:

`aceAutocomplete` *Enable Code Completion for an Ace Code Input*

Description

This function dynamically auto complete R code pieces using built-in function `utils:::.win32consoleCompletion`. Please see `rcompgen` for details.

Usage

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

Arguments

`inputId`	The id of the input object
`session`	The `session` object passed to function given to shinyServer

Details

You can implement your own code completer by listening to `input$<editorId>_shinyAce_hint` where <editorId> is the `aceEditor` id. The input contains

- `linebuffer`: Code/Text at current editing line
- `cursorPosition`: Current cursor position at this line

Value

An observer reference class object that is responsible for offering code completion. See `observe` for more details. You can use `suspend` or `destroy` to pause to stop dynamic code completion.

`aceEditor` *Render Ace*

Description

Render an Ace editor on an application page.

Usage

```
aceEditor(outputId, value, mode, theme, vimKeyBinding = FALSE,
  readOnly = FALSE, height = "400px", fontSize = 12,
  debounce = 1000, wordWrap = FALSE, showLineNumbers = TRUE,
  highlightActiveLine = TRUE, selectionId = NULL, cursorId = NULL,
  hotkeys = NULL, autoComplete = c("disabled", "enabled", "live"),
  autoCompleters = "", autoCompleteList = NULL, tabSize = 4,
  useSoftTabs = TRUE, showInvisibles = FALSE,
  setBehavioursEnabled = TRUE, autoScrollEditorIntoView = FALSE,
  maxLines = NULL, minLines = NULL)
```

Arguments

outputId	The ID associated with this element
value	The initial text to be contained in the editor.
mode	The Ace mode to be used by the editor. The mode in Ace is often the programming or markup language that you're using and determines things like syntax highlighting and code folding. Use the getAceModes function to enumerate all the modes available.
theme	The Ace theme to be used by the editor. The theme in Ace determines the styling and coloring of the editor. Use getAceThemes to enumerate all the themes available.
vimKeyBinding	If set to TRUE, Ace will enable vim-keybindings. Default value is FALSE.
readOnly	If set to TRUE, Ace will disable client-side editing. If FALSE (the default), it will enable editing.
height	A number (which will be interpreted as a number of pixels) or any valid CSS dimension (such as "50%", "200px", or "auto").
fontSize	Defines the font size (in px) used in the editor and should be an integer. The default is 12.
debounce	The number of milliseconds to debounce the input. This will cause the client to withhold update notifications until the user has stopped typing for this amount of time. If 0, the server will be notified of every keystroke as it happens.
wordWrap	If set to TRUE, Ace will enable word wrapping. Default value is FALSE.
showLineNumbers	If set to TRUE, Ace will show line numbers.
highlightActiveLine	If set to TRUE, Ace will highlight the active line.
selectionId	The ID associated with a change of selected text
cursorId	The ID associated with a cursor change.
hotkeys	A list whose names are ID names and whose elements are the shortcuts of keys. Shortcuts can either be a simple string or a list with elements 'win' and 'mac' that that specifies different shortcuts for win and mac (see example).
autoComplete	Enable/Disable auto code completion. Must be one of the following: "disabled" Disable Code Autocomplete

"enabled" Enable Basic Code Autocomplete. Autocomplete can be triggered using Ctrl-Space, Ctrl-Shift-Space, or Alt-Space.

"live" Enable Live Code Autocomplete. In addition to Basic Autocomplete, it will automatically trigger at each key stroke.

By default, only local completer is used where all aforementioned code pieces will be considered as candidates. Use autoCompleteList for static completions and aceAutocomplete for dynamic R code completions.

autoCompleters List of completers to enable. If set to NULL, all completers will be disabled. Select one or more of "snippet", "text", "static", and "keyword" to control which completers to use. Default option is an empty character vector which does not effect default completion options

autoCompleteList
A named list that contains static code completions candidates. This can be especially useful for Non-Standard Evaluation (NSE) functions such as those in dplyr and ggvis. Each element in list should be a character array whose words will be listed under the element key. For example, to suggests column names from mtcars and airquality, you can use list(mtcars = colnames(mtcars), airquality = colna

tabSize Set tab size. Default value is 4

useSoftTabs Replace tabs by spaces. Default value is TRUE

showInvisibles Show invisible characters (e.g., spaces, tabs, newline characters). Default value is FALSE

setBehavioursEnabled
Determines if the auto-pairing of special characters, like quotation marks, parenthesis, or brackets should be enabled. Default value is TRUE.

autoScrollEditorIntoView
If TRUE, expands the size of the editor window as new lines are added

maxLines Maximum number of lines the editor window will expand to when autoScrollEditorIntoView is TRUE

minLines Minimum number of lines in the editor window when autoScrollEditorIntoView is TRUE

Author(s)

Jeff Allen <jeff@trestletech.com>

Examples

```
## Not run:
 aceEditor(
   outputId = "myEditor",
   value = "Initial text for editor here",
   mode = "r",
   theme = "ambiance"
 )

 aceEditor(
   outputId = "myCodeEditor",
```

```
  value = "# Enter code",
  mode = "r",
  hotkeys = list(
    helpKey = "F1",
    runKey = list(
      win = "Ctrl-R|Ctrl-Shift-Enter",
      mac = "CMD-ENTER|CMD-SHIFT-ENTER"
    )
  ),
  wordWrap = TRUE, debounce = 10
)

aceEditor(
  outputId = "mySmartEditor",
  value = "plot(wt ~ mpg, data = mtcars)",
  mode = "r",
  autoComplete = "live",
  autoCompleteList = list(mtcars = colnames(mtcars))
)

## End(Not run)
```

`getAceModes` *Get available modes*

Description

Gets all of the available `modes` available in the installed version of shinyAce. Modes are often the programming or markup language which will be used in the editor and determine things like syntax highlighting and code folding.

Usage

```
getAceModes()
```

Author(s)

Jeff Allen <jeff@trestletech.com>

getAceThemes *Get available themes*

Description

Gets all of the available `themes` available in the installed version of shinyAce. Themes determine the styling and colors used in the editor.

Usage

```
getAceThemes()
```

Author(s)

Jeff Allen <jeff@trestletech.com>

jsQuote *Escape a JS String*

Description

Escape a String to be sent to JavaScript

Usage

```
jsQuote(text)
```

Arguments

`text`	The text to escape

Author(s)

Jeff Allen <jeff@trestletech.com>

updateAceEditor *Update Ace Editor*

Description

Update the styling or mode of an aceEditor component.

Usage

```
updateAceEditor(session, editorId, value, theme, readOnly, mode, fontSize,
  wordWrap, useSoftTabs, tabSize, showInvisibles, border = c("normal",
  "alert", "flash"), autoComplete = c("disabled", "enabled", "live"),
  autoCompleters = c("snippet", "text", "keyword", "static", "rlang"),
  autoCompleteList = NULL)
```

Arguments

session	The Shiny session to whom the editor belongs
editorId	The ID associated with this element
value	The initial text to be contained in the editor.
theme	The Ace theme to be used by the editor. The theme in Ace determines the styling and coloring of the editor. Use getAceThemes to enumerate all the themes available.
readOnly	If set to TRUE, Ace will disable client-side editing. If FALSE (the default), it will enable editing.
mode	The Ace mode to be used by the editor. The mode in Ace is often the programming or markup language that you're using and determines things like syntax highlighting and code folding. Use the getAceModes function to enumerate all the modes available.
fontSize	If set, will update the font size (in px) used in the editor. Should be an integer.
wordWrap	If set to TRUE, Ace will enable word wrapping. Default value is FALSE.
useSoftTabs	Replace tabs by spaces. Default value is TRUE
tabSize	Set tab size. Default value is 4
showInvisibles	Show invisible characters (e.g., spaces, tabs, newline characters). Default value is FALSE
border	Set the border 'normal', 'alert', or 'flash'.
autoComplete	Enable/Disable code completion. See aceEditor for details.
autoCompleters	List of completers to enable. If set to NULL, all completers will be disabled.
autoCompleteList	If set to NULL, existing static completions list will be unset. See aceEditor for details.

Author(s)

Jeff Allen <jeff@trestletech.com>

Examples

```
## Not run:
 shinyServer(function(input, output, session) {
   observe({
     updateAceEditor(session, "myEditor", "Updated text for editor here",
       mode = "r", theme = "ambiance")
   })
 }

## End(Not run)
```

Index