Package 'shinyEditor'

February 8, 2026

Type Package

Title 'Ace' and 'Monaco' Editors Bindings for 'shiny' Application

Version 0.1.0

Date 2026-02-05

Maintainer zearoby <949386232@qq.com>

Description 'Ace' and 'Monaco' editor bindings to enable a rich text widget within 'shiny' application and provide more features, e.g. text comparison, spell checking and an extra 'SAS' code highlight mode.

License MIT + file LICENSE

URL https://github.com/zearoby/shinyEditor

BugReports https://github.com/zearoby/shinyEditor/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0.0)

Imports htmlwidgets, jsonlite, memoise, shinyjs, shiny, systemfonts, methods, yaml

Suggests devtools, testthat

Config/testthat/edition 3

NeedsCompilation no

Author zearoby [aut, cre]

Repository CRAN

Date/Publication 2026-02-08 17:00:03 UTC

Contents

aceDiffEditor 3
aceDiffEditor-shiny 4
aceEditor 4
aceEditor-shiny 6

appendAceCompleter . . . 7
check_output_id . . . 8
createAceDiffView . . . 9
createMonacoDiffView . . . 9
getAceCursorPosition . . . 10
getAceModes . . . 11
getAceSelectedText . . . 11
getAceSelectionRange . . . 12
getAceThemes . . . 12
getAceValue . . . 13
getMonacoCursorPosition . . . 13
getMonacoLanguages . . . 14
getMonacoSelectedText . . . 14
getMonacoSelectionRange . . . 15
getMonacoThemes . . . 15
getMonacoValue . . . 16
getPackageStatus . . . 16
getSystemFontFamilies . . . 17
monacoDiffEditor . . . 17
monacoDiffEditor-shiny . . . 18
monacoEditor . . . 19
monacoEditor-shiny . . . 21
onAceEditorReady . . . 21
onMonacoEditorReady . . . 22
removeAceCompleter . . . 23
removeAceDiffView . . . 23
removeMonacoDiffView . . . 24
setAceEnableAutocompletion . . . 24
setAceEnableSpellCheck . . . 25
setAceHighlightActiveLine . . . 26
setAceLineNumbersVisible . . . 27
setAceMode . . . 27
setAceNewLineMode . . . 28
setAceOption . . . 29
setAceOptions . . . 29
setAceReadOnly . . . 30
setAceShowInvisibles . . . 31
setAceStatusBarVisible . . . 31
setAceTheme . . . 32
setAceValue . . . 33
setMonacoLanguage . . . 33
setMonacoTheme . . . 34
setMonacoValue . . . 35
updateMonacoOption . . . 35
updateMonacoOptions . . . 36

Index **38**

aceDiffEditor *Render an Ace aceDiffEditor*

Description

Render an Ace diff editor on an application page.

Usage

```
aceDiffEditor(
  valueA,
  valueB,
  mode = "ace/mode/text",
  enableSpellCheck = FALSE,
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

valueA	character: Set text to first editor when initializing
valueB	character: Set text to second editor when initializing
mode	character: The Ace `shinyEditor::getAceModes()` to be used by the editor
enableSpellCheck	logical: Enable check typo of spelling
...	For more `EditorOption`, please refer to https://ace.c9.io/api/interfaces/ace.Ace.EditorOptions.html
width	integer, character: Width in pixels (optional, defaults to automatic sizing)
height	integer, character: Height in pixels (optional, defaults to automatic sizing)
elementId	character: An element id for the widget (a random character by default)

Value

Widget for shiny application

Examples

```
if(interactive()){
    shinyEditor::aceDiffEditor(valueA = "text1", valueB = "text2")
}
```

aceDiffEditor-shiny *Shiny bindings for aceDiffEditor*

Description

Output and render functions for using aceDiffEditor within Shiny applications and interactive Rmd documents.

Usage

```
aceDiffEditorOutput(outputId, width = "100%", height = "400px")

renderAceDiffEditor(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a aceDiffEditor
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

htmlwidgets::shinyWidgetOutput

htmlwidgets::shinyRenderWidget

aceEditor *Render an Ace editor*

Description

Render an Ace editor on an application page.

Usage

```
aceEditor(
  value,
  enableBasicAutocompletion = TRUE,
  enableSnippets = TRUE,
  enableLiveAutocompletion = TRUE,
  enableSpellCheck = FALSE,
```

```
  fontFamily = "Consolas",
  fontSize = 16,
  highlightActiveLine = TRUE,
  mode = "ace/mode/text",
  newLineMode = "auto",
  placeholder = NULL,
  printMarginColumn = 120,
  readOnly = FALSE,
  scrollPastEnd = 0.5,
  showInvisibles = TRUE,
  showLineNumbers = TRUE,
  showPrintMargin = TRUE,
  showStatusBar = TRUE,
  tabSize = 4,
  theme = "ace/theme/xcode",
  useSoftTabs = TRUE,
  wrap = FALSE,
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`value`	character: Set text to editor when initializing
`enableBasicAutocompletion`	logical: Enable basic code automatically completion when editing
`enableSnippets`	logical: Enable code snippets automatically completion when editing
`enableLiveAutocompletion`	logical: Enable live code automatically completion when editing
`enableSpellCheck`	logical: Enable check typo of spelling
`fontFamily`	character: System font name
`fontSize`	integer: Font size
`highlightActiveLine`	logical: Highlight the current line
`mode`	character: The Ace `shinyEditor::getAceModes()` to be used by the editor
`newLineMode`	character: Set the end of line character Valid values: windows, unix, auto
`placeholder`	character: A string to use a placeholder when the editor has no content
`printMarginColumn`	integer: The print margin column width
`readOnly`	logical: Set editor to readOnly
`scrollPastEnd`	integer: Scroll past end Valid values: 0 to 1, TRUE, FALSE
`showInvisibles`	logical: Show invisible characters

showLineNumbers
: logical: Show line number area

showPrintMargin
: logical: Show print margin

showStatusBar
: logical: Show statusBar

tabSize
: integer: Tab size

theme
: character: The Ace `shinyEditor::getAceThemes()` to be used by the editor

useSoftTabs
: logical: Replace tabs by spaces

wrap
: logical: If set to `TRUE`, Ace will enable word wrapping

...
: For more `EditorOption`, please refer to https://ace.c9.io/api/interfaces/ace.Ace.EditorOptions.html

width
: integer, character: Width in pixels (optional, defaults to automatic sizing)

height
: integer, character: Height in pixels (optional, defaults to automatic sizing)

elementId
: character: An element id for the widget (a random character by default)

Value

Widget for shiny application

Examples

```
if(interactive()){
    shinyEditor::aceEditor(value = "text")
}
```

---

aceEditor-shiny *Shiny bindings for aceEditor*

Description

Output and render functions for using aceEditor within Shiny applications and interactive Rmd documents.

Usage

```
aceEditorOutput(outputId, width = "100%", height = "400px")

renderAceEditor(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a aceEditor
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

htmlwidgets::shinyWidgetOutput

htmlwidgets::shinyRenderWidget

---

appendAceCompleter *Add completer*

Description

Add completer to editor.completers. Please refer to https://ace.c9.io/api/interfaces/ace.Ace.Completer.html

Usage

```
appendAceCompleter(
  outputId,
  id,
  completer,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The element id of the first editor
id	list: Completer id
completer	list: Completer list
session	environment: The Shiny session object for the editor (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::appendAceCompleter(
       outputId = "editor",
       id = "custom_completer",
       completer = list(
          list(value = "function", caption = "function", meta = "keyword"),
          list(value = "if", caption = "if", meta = "keyword"),
          list(value = "else", caption = "else", meta = "keyword"),
          list(value = "for", caption = "for", meta = "keyword"),
          list(value = "while", caption = "while", meta = "keyword"),
          list(value = "console.log()", caption = "console.log", meta = "function"),
        list(value = "myCustomFunction()", caption = "myCustomFunction", meta = "custom")
       )
    )
}
```

---

check_output_id *Check outputId*

Description

Check outputId is character and exist in shiny session

Usage

```
check_output_id(outputId)
```

Arguments

outputId	character: The id of the table to be manipulated

Examples

```
if(interactive()){
    check_output_id("table_id")
}
```

createAceDiffView *Create diff view*

Description

Create diff view for exist editors

Usage

```
createAceDiffView(
  editorAId,
  editorBId,
  sessionA = shiny::getDefaultReactiveDomain(),
  sessionB = shiny::getDefaultReactiveDomain()
)
```

Arguments

editorAId	character: The element id of the first editor
editorBId	character: The element id of the second editor
sessionA	environment: The Shiny session object for the first editor (from the server function of the Shiny app).
sessionB	environment: The Shiny session object for the second editor (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::createAceDiffView(editorAId = "editor1", editorBId = "editor2")
}
```

createMonacoDiffView *Create monaco diff view*

Description

Create monaco diff view for exist editors in an exist widget

Usage

```
createMonacoDiffView(
  editorAId,
  editorBId,
  elementId,
  sessionA = shiny::getDefaultReactiveDomain(),
  sessionB = shiny::getDefaultReactiveDomain()
)
```

Arguments

editorAId	character: The element id of the first editor
editorBId	character: The element id of the second editor
elementId	character: The element id of the exist widget to show monacoDiffEditor
sessionA	environment: The Shiny session object for the first editor (from the server function of the Shiny app)
sessionB	environment: The Shiny session object for the second editor (from the server function of the Shiny app)

Value

No return value, called for side effects

Examples

```
if(interactive()){
   shinyEditor::createMonacoDiffView(editorAId = "editor1", editorBId = "editor2")
}
```

---

getAceCursorPosition *Get cursor position in aceEditor*

Description

Get cursor position in aceEditor

Usage

```
getAceCursorPosition(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

List of cursor position

Examples

```
if(interactive()){
    shinyEditor::getAceCursorPosition(outputId = "editor")
}
```

---

getAceModes *Get all ace modes*

Description

Gets all of the available `modes` available in the installed version of ace editor. 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()
```

Value

List of all modes in Ace editor

---

getAceSelectedText *Get selected text in aceEditor*

Description

Get selected text in aceEditor

Usage

```
getAceSelectedText(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

`outputId`	character: The id of the editor
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

Character of selected text

Examples

```
if(interactive()){
    shinyEditor::getAceSelectedText(outputId = "editor")
}
```

`getAceSelectionRange` *Get selection range in aceEditor*

Description

Get selection range in aceEditor

Usage

```
getAceSelectionRange(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

`outputId`	character: The id of the editor
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

List of selection range

Examples

```
if(interactive()){
    shinyEditor::getAceSelectionRange(outputId = "editor")
}
```

`getAceThemes` *Get all ace themes*

Description

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

Usage

```
getAceThemes()
```

Value

List of all themes in Ace editor

getAceValue *Get value in aceEditor*

Description

Get value in aceEditor

Usage

```
getAceValue(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

Character of editor text

Examples

```
if(interactive()){
    shinyEditor::getAceValue(outputId = "editor")
}
```

getMonacoCursorPosition *Get cursor position in monacoEditor*

Description

Get cursor position in monacoEditor

Usage

```
getMonacoCursorPosition(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

List of cursor position

Examples

```
if(interactive()){
    shinyEditor::getMonacoCursorPosition(outputId = "editor")
}
```

---

getMonacoLanguages *Get all ace modes*

Description

Gets all of the available `modes` available in the installed version of ace editor. 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

```
getMonacoLanguages()
```

Value

List of all languages in Monaco editor

---

getMonacoSelectedText *Get selected text in monacoEditor*

Description

Get selected text in monacoEditor

Usage

```
getMonacoSelectedText(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

`outputId`	character: The id of the editor
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

Character of selected text

Examples

```
if(interactive()){
    shinyEditor::getMonacoSelectedText(outputId = "editor")
}
```

`getMonacoSelectionRange` *Get selection range in monacoEditor*

Description

Get selection range in monacoEditor

Usage

```
getMonacoSelectionRange(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

`outputId`	character: The id of the editor
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

List of selection range

Examples

```
if(interactive()){
    shinyEditor::getMonacoSelectionRange(outputId = "editor")
}
```

`getMonacoThemes` *Get all monaco themes*

Description

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

Usage

```
getMonacoThemes()
```

Value

List of all themes in Monaco editor

getMonacoValue *Get value in monacoEditor*

Description

Get value in monacoEditor

Usage

```
getMonacoValue(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

Character of editor text

Examples

```
if(interactive()){
    shinyEditor::getMonacoValue(outputId = "editor")
}
```

getPackageStatus *Get status of shinyEditor package*

Description

Get status of shinyEditor package.

Usage

```
getPackageStatus()
```

Value

List of package name, package version, package date, Ace editor version, Monaco editor version

Examples

```
shinyEditor::getPackageStatus()
```

getSystemFontFamilies *Get system font families*

Description

Get system font families

Usage

```
getSystemFontFamilies()
```

Value

List of system font families

monacoDiffEditor *Render an Ace monacoDiffEditor*

Description

Render an Ace diff editor on an application page.

Usage

```
monacoDiffEditor(
  valueA,
  valueB,
  language = "plaintext",
  ignoreTrimWhitespace = FALSE,
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

valueA	character: Set text to first editor when initializing
valueB	character: Set text to second editor when initializing
language	character: The initial language of the auto created model in the editor. To not automatically create a model, use model: null
ignoreTrimWhitespace	logical: Compute the diff by ignoring leading/trailing whitespace Defaults to false

...	For more arguments, please refer to https://microsoft.github.io/monaco-editor/typedoc/interfaces/editor_editor_api.editor.IDiffEditorOptions.html
width	integer, character: Width in pixels (optional, defaults to automatic sizing)
height	integer, character: Height in pixels (optional, defaults to automatic sizing)
elementId	character: An element id for the widget (a random character by default)

Value

Widget for shiny application

Examples

```
if(interactive()){
    shinyEditor::monacoDiffEditor(valueA = "text1", valueB = "text2")
}
```

---

monacoDiffEditor-shiny

Shiny bindings for monacoDiffEditor

Description

Output and render functions for using monacoDiffEditor within Shiny applications and interactive Rmd documents.

Usage

```
monacoDiffEditorOutput(outputId, width = "100%", height = "400px")

renderMonacoDiffEditor(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a monacoDiffEditor
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

htmlwidgets::shinyWidgetOutput

htmlwidgets::shinyRenderWidget

`monacoEditor` *Render an monaco editor*

Description

Render an Monaco editor on an application page.

Usage

```
monacoEditor(
  value,
  fontFamily = "Consolas",
  fontSize = 16,
  insertSpaces = TRUE,
  language = "plaintext",
  lineNumbers = "on",
  placeholder = NULL,
  readOnly = FALSE,
  renderWhitespace = "boundary",
  rulers = list(80, 100, 120),
  scrollBeyondLastLine = TRUE,
  showStatusBar = TRUE,
  tabSize = 4,
  theme = "vs",
  wordWrap = "off",
  automaticLayout = TRUE,
  ...,
  width = NULL,
  height = NULL,
  elementId = NULL
)
```

Arguments

`value`	character: Set text to editor when initializing
`fontFamily`	character: The font family
`fontSize`	integer: The font size
`insertSpaces`	logical: Insert spaces when pressing Tab. This setting is overridden based on the file contents when detectIndentation is on. Defaults to true.
`language`	character: The initial language of the auto created model in the editor. To not automatically create a model, use model: null.
`lineNumbers`	character, integer: Control the rendering of line numbers. If it is a function, it will be invoked when rendering a line number and the return value will be rendered. Otherwise, if it is a truthy, line numbers will be rendered normally (equivalent of using an identity function). Otherwise, line numbers will not be rendered. Defaults to on.

placeholder — character: Sets a placeholder for the editor. If set, the placeholder is shown if the editor is empty.

readOnly — logical: Should the editor be read only. See also domReadOnly. Defaults to false.

renderWhitespace — character: Enable rendering of whitespace. Defaults to 'selection'. Valid values: "all" | "none" | "boundary" | "selection" | "trailing"

rulers — list: Render vertical lines at the specified columns. Defaults to empty list.

scrollBeyondLastLine — logical: Enable that scrolling can go one screen size after the last line. Defaults to true.

showStatusBar — logical: Show statusBar

tabSize — integer: The number of spaces a tab is equal to. This setting is overridden based on the file contents when detectIndentation is on. Defaults to 4.

theme — character: Initial theme to be used for rendering. The current out-of-the-box available themes are: 'vs' (default), 'vs-dark', 'hc-black', 'hc-light. You can create custom themes via monaco.editor.defineTheme. To switch a theme, use monaco.editor.setTheme. NOTE: The theme might be overwritten if the OS is in high contrast mode, unless autoDetectHighContrast is set to false.

wordWrap — character: Control the wrapping of the editor. When wordWrap = "off", the lines will never wrap. When wordWrap = "on", the lines will wrap at the viewport width. When wordWrap = "wordWrapColumn", the lines will wrap at wordWrapColumn. When wordWrap = "bounded", the lines will wrap at min(viewport width, wordWrapColumn). Defaults to "off". Valid values: "off" | "on" | "wordWrapColumn" | "bounded"

automaticLayout — logical: Enable that the editor will install a ResizeObserver to check if its container dom node size has changed. Defaults to TRUE.

... — For more arguments, please refer to https://microsoft.github.io/monaco-editor/docs.html#interfaces/editor_editor_api.editor.IStandaloneEditorConstructionOptions.html

width — integer, character: Width in pixels (optional, defaults to automatic sizing)

height — integer, character: Height in pixels (optional, defaults to automatic sizing)

elementId — character: An element id for the widget (a random character by default)

Value

Widget for shiny application

Examples

```
if(interactive()){
    shinyEditor::monacoEditor(value = "text")
}
```

monacoEditor-shiny *Shiny bindings for monacoEditor*

Description

Output and render functions for using monacoEditor within Shiny applications and interactive Rmd documents.

Usage

```
monacoEditorOutput(outputId, width = "100%", height = "400px")

renderMonacoEditor(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
width, height	Must be a valid CSS unit (like '100%', '400px', 'auto') or a number, which will be coerced to a string and have 'px' appended.
expr	An expression that generates a monacoEditor
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

htmlwidgets::shinyWidgetOutput

htmlwidgets::shinyRenderWidget

onAceEditorReady *Fires upon aceEditor initialisation*

Description

Fires upon aceEditor initialisation

Usage

```
onAceEditorReady(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

TRUE

Examples

```
if(interactive()){
    shinyEditor::onAceEditorReady(outputId = "editor")
}
```

---

onMonacoEditorReady *Fires upon monacoEditor initialisation*

Description

Fires upon monacoEditor initialisation

Usage

```
onMonacoEditorReady(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

TRUE

Examples

```
if(interactive()){
    shinyEditor::onMonacoEditorReady(outputId = "editor")
}
```

removeAceCompleter *Remove completer*

Description

Remove completer in editor.completers. Please refer to https://ace.c9.io/api/interfaces/ace.Ace.Completer.html

Usage

```
removeAceCompleter(outputId, id, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The element id of the first editor
id	list: Completer id
session	environment: The Shiny session object for the editor (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::removeAceCompleter(outputId = "editor", id = "custom_completer")
}
```

removeAceDiffView *Remove ace diff view*

Description

Remove ace diff view for exist editors

Usage

```
removeAceDiffView(outputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The element id of the first editor
session	environment: The Shiny session object for the editor (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::removeAceDiffView(outputId = "editor")
}
```

`removeMonacoDiffView` *Remove monaco diff view*

Description

Remove monaco diff view for exist editors

Usage

```
removeMonacoDiffView(elementId)
```

Arguments

`elementId`	character: The element id of the monacoDiffEditor

Value

No return value, called for side effects

Examples

```
if(interactive()){
   shinyEditor::removeMonacoDiffView(elementId = "editor1")
}
```

`setAceEnableAutocompletion`
Enable or disable code completion

Description

Enable or disable code completion in aceEditor

Usage

```
setAceEnableAutocompletion(
  outputId,
  enable,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
enable	logical: TRUE or FALSE
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceEnableAutocompletion(outputId = "editor", enable = TRUE)
}
```

---

setAceEnableSpellCheck

Enable or disable spell check

Description

Enable or disable spell check

Usage

```
setAceEnableSpellCheck(
  outputId,
  enable = TRUE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
enable	logical: Set spell check TRUE or FALSE
session	environment: The Shiny session object for the editor (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceEnableSpellCheck(outputId = "editor", enable = TRUE)
}
```

`setAceHighlightActiveLine` *Highlight the current line*

Description

Determines whether or not the current line should be highlighted.

Usage

```
setAceHighlightActiveLine(
  outputId,
  visible,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`outputId`	character: The id of the editor
`visible`	logical: TRUE or FALSE
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceHighlightActiveLine(outputId = "editor", visible = TRUE)
}
```

`setAceLineNumbersVisible` *Show or hide line number area in aceEditor*

Description

Show or hide line number area in aceEditor

Usage

```
setAceLineNumbersVisible(
  outputId,
  visible,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`outputId`	character: The id of the editor
`visible`	logical: TRUE or FALSE
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceLineNumbersVisible(outputId = "editor", visible = TRUE)
}
```

`setAceMode` *Set new mode*

Description

Set a new mode for the EditSession.

Usage

```
setAceMode(outputId, mode, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
mode	character: The mode of the code language
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceMode(outputId = "editor", mode = "r")
}
```

---

setAceNewLineMode *Set the new line mode to aceEditor*

Description

Set the new line mode to aceEditor

Usage

```
setAceNewLineMode(
  outputId,
  newLineMode,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
newLineMode	character: The new line mode in aceEditor Valid values: windows, unix, auto
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceNewLineMode(outputId = "editor", newLineMode = "windows")
}
```

setAceOption *Set an option to aceEditor*

Description

Set an option to aceEditor

Usage

```
setAceOption(
  outputId,
  name,
  value,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
name	character: Option name. Refer to https://ace.c9.io/api/interfaces/ace.Ace.EditorOptions.html
value	character, integer, logical: Option value. Refer to https://ace.c9.io/api/interfaces/ace.Ace.EditorOptions.html
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceOption(outputId = "editor", name = "tabSize", value = 3)
}
```

setAceOptions *Set options to aceEditor*

Description

Set options to aceEditor

Usage

```
setAceOptions(outputId, options, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
options	list: Editor options. Refer to https://ace.c9.io/api/interfaces/ace.Ace.EditorOptions.html
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceOptions(outputId = "editor", options = list())
}
```

---

setAceReadOnly *Set readOnly for aceEditor*

Description

If readOnly is true, then the editor is set to read-only mode, and none of the content can change.

Usage

```
setAceReadOnly(outputId, readOnly, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
readOnly	logical: TRUE or FALSE
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceReadOnly(outputId = "editor", readOnly = TRUE)
}
```

`setAceShowInvisibles` *Show or hide invisible characters in aceEditor*

Description

Show or hide invisible characters in aceEditor

Usage

```
setAceShowInvisibles(
  outputId,
  visible,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`outputId`	character: The id of the editor
`visible`	logical: TRUE or FALSE
`session`	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceShowInvisibles(outputId = "editor", visible = TRUE)
}
```

`setAceStatusBarVisible` *Show or hide the statusBar*

Description

Show or hide the statusBar of aceEditor

Usage

```
setAceStatusBarVisible(
  outputId,
  visible,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
visible	logical: TRUE or FALSE
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceStatusBarVisible(outputId = "editor", visible = TRUE)
}
```

---

setAceTheme *Set new theme*

Description

Set a new theme for the editor. theme should exist, like ace/theme/github

Usage

```
setAceTheme(outputId, theme, session = shiny::getDefaultReactiveDomain())
```

Arguments

outputId	character: The id of the editor
theme	character: The theme of the aceEditor
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceTheme(outputId = "editor", theme = "ace/theme/github")
}
```

setAceValue *Replace text with new text*

Description

Replace all the lines in the current Document with the value of text.

Usage

```
setAceValue(
  outputId,
  value,
  clearChangedHistory = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
value	character: The text of the editor
clearChangedHistory	logical: Clear undo/redo history
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setAceValue(outputId = "editor", value = "text")
}
```

setMonacoLanguage *Set language*

Description

Set language to monaco editor

Usage

```
setMonacoLanguage(
  outputId,
  language,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
language	character: The highlight of code
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setMonacoLanguage(outputId = "editor", language = "text")
}
```

---

setMonacoTheme *Set new theme*

Description

Set a new theme for the editor. theme should exist, like `vs-dark`

Usage

```
setMonacoTheme(theme)
```

Arguments

theme	character: The theme of the monacoEditor

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setMonacoTheme(theme = "vs")
}
```

setMonacoValue *Replace text with new text*

Description

Replace all the lines in the current Document with the value of text.

Usage

```
setMonacoValue(
  outputId,
  value,
  clearChangedHistory = FALSE,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
value	character: The text of the editor
clearChangedHistory	logical: Clear undo/redo history
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::setMonacoValue(outputId = "editor", value = "text")
}
```

updateMonacoOption *Update an option to monacoEditor*

Description

Update an option to monacoEditor

Usage

```
updateMonacoOption(
  outputId,
  name,
  value,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
name	character: Option name. Refer to https://microsoft.github.io/monaco-editor/docs.html#interfaces/editor_editor_api.editor.IEditorOptions.html
value	character, integer, logical: Option value. Refer to https://microsoft.github.io/monaco-editor/docs.html#interfaces/editor_editor_api.editor.IEditorOptions.html
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::updateMonacoOption(outputId = "editor", name = "tabSize", value = 3)
}
```

---

updateMonacoOptions *Update options to monacoEditor*

Description

Update options to monacoEditor

Usage

```
updateMonacoOptions(
  outputId,
  options,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

outputId	character: The id of the editor
options	list: monaco editor options. Refer to https://microsoft.github.io/monaco-editor/docs.html#interfaces/editor_editor_api.editor.IEditorOptions.html
session	environment: The Shiny session object (from the server function of the Shiny app).

Value

No return value, called for side effects

Examples

```
if(interactive()){
    shinyEditor::updateMonacoOptions(outputId = "editor", options = list())
}
```

Index

aceDiffEditor, 3
aceDiffEditor-shiny, 4
aceDiffEditorOutput *(aceDiffEditor-shiny)*, 4
aceEditor, 4
aceEditor-shiny, 6
aceEditorOutput *(aceEditor-shiny)*, 6
appendAceCompleter, 7

character, *3*, *5–37*
check_output_id, 8
createAceDiffView, 9
createMonacoDiffView, 9

environment, *7*, *9–16*, *21–23*, *25–37*

getAceCursorPosition, 10
getAceModes, 11
getAceSelectedText, 11
getAceSelectionRange, 12
getAceThemes, 12
getAceValue, 13
getMonacoCursorPosition, 13
getMonacoLanguages, 14
getMonacoSelectedText, 14
getMonacoSelectionRange, 15
getMonacoThemes, 15
getMonacoValue, 16
getPackageStatus, 16
getSystemFontFamilies, 17

integer, *3*, *5*, *6*, *18–20*, *29*, *36*

list, *7*, *20*, *23*, *30*, *37*
logical, *3*, *5*, *6*, *17*, *19*, *20*, *25–27*, *29–33*, *35*, *36*

monacoDiffEditor, 17
monacoDiffEditor-shiny, 18
monacoDiffEditorOutput *(monacoDiffEditor-shiny)*, 18
monacoEditor, 19
monacoEditor-shiny, 21
monacoEditorOutput *(monacoEditor-shiny)*, 21

onAceEditorReady, 21
onMonacoEditorReady, 22

removeAceCompleter, 23
removeAceDiffView, 23
removeMonacoDiffView, 24
renderAceDiffEditor *(aceDiffEditor-shiny)*, 4
renderAceEditor *(aceEditor-shiny)*, 6
renderMonacoDiffEditor *(monacoDiffEditor-shiny)*, 18
renderMonacoEditor *(monacoEditor-shiny)*, 21

setAceEnableAutocompletion, 24
setAceEnableSpellCheck, 25
setAceHighlightActiveLine, 26
setAceLineNumbersVisible, 27
setAceMode, 27
setAceNewLineMode, 28
setAceOption, 29
setAceOptions, 29
setAceReadOnly, 30
setAceShowInvisibles, 31
setAceStatusBarVisible, 31
setAceTheme, 32
setAceValue, 33
setMonacoLanguage, 33
setMonacoTheme, 34
setMonacoValue, 35

updateMonacoOption, 35
updateMonacoOptions, 36