

# Package ‘monaco’

July 23, 2025

**Type** Package

**Title** The 'Monaco' Editor as a HTML Widget

**Version** 0.2.2

**Description** A HTML widget rendering the 'Monaco' editor. The 'Monaco' editor is the code editor which powers 'VS Code'. It is particularly well developed for 'JavaScript'. In addition to the built-in features of the 'Monaco' editor, the widget allows to prettify multiple languages, to view the 'HTML' rendering of 'Markdown' code, and to view and resize 'SVG' images.

**URL** <https://github.com/stla/monaco>

**BugReports** <https://github.com/stla/monaco/issues>

**License** GPL-3

**Encoding** UTF-8

**Imports** htmlwidgets (>= 1.5.3), rstudioapi, tools, htmltools, shiny

**Suggests** sass

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Author** Stéphane Laurent [aut, cre],  
Microsoft Corporation [ctb, cph] ('Monaco Editor' library),  
James Long and contributors [ctb, cph] ('Prettier' library),  
Rich Harris [ctb, cph] ('svg-parser' library),  
Lionel Tzatzkin [ctb, cph] ('scale-that-svg' library),  
Andrei Kashcha [ctb, cph] ('panzoom' library),  
Vitaly Puzrin [ctb, cph] ('markdown-it' library),  
Alex Kocharin [ctb, cph] ('markdown-it' library),  
John Schlinkert [ctb, cph] ('word-wrap' library),  
jQuery contributors [ctb, cph] ('jQuery' library),  
Kyle Fox [ctb, cph] ('jQuery Modal' library),  
Tristan Edwards [ctb, cph] ('sweetalert2' library),  
Limon Monte [ctb, cph] ('sweetalert2' library)

**Maintainer** Stéphane Laurent <laurent\_step@outlook.fr>

**Repository** CRAN

**Date/Publication** 2022-05-18 09:20:02 UTC

Contents

|                              |          |
|------------------------------|----------|
| getMonacoLanguages . . . . . | 2        |
| getMonacoThemes . . . . .    | 2        |
| monaco . . . . .             | 2        |
| monaco-shiny . . . . .       | 5        |
| <b>Index</b>                 | <b>8</b> |

---

|                    |                         |
|--------------------|-------------------------|
| getMonacoLanguages | <i>Monaco languages</i> |
|--------------------|-------------------------|

---

Description

Get the list of available languages in the Monaco editor.

Usage

getMonacoLanguages()

---

|                 |                      |
|-----------------|----------------------|
| getMonacoThemes | <i>Monaco themes</i> |
|-----------------|----------------------|

---

Description

Get the list of available themes of the Monaco editor. All themes are dark, excepted "vs".

Usage

getMonacoThemes()

---

|        |                      |
|--------|----------------------|
| monaco | <i>Monaco editor</i> |
|--------|----------------------|

---

Description

Open the Monaco editor.

## Usage

```
monaco(  
  contents,  
  language = NULL,  
  theme = NULL,  
  tabSize = NULL,  
  fontSize = 14,  
  header = TRUE,  
  width = NULL,  
  height = NULL,  
  elementId = NULL  
)
```

## Arguments

|               |                                                                                                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| contents      | this can be the path to a file, NULL to open an empty editor, missing to open the file currently open in RStudio, or a character vector which corresponds to the lines of a file                               |
| language      | the language of the contents; if NULL and the contents are read from a file, the mode is guessed from the extension of the file; run <a href="#">getMonacoLanguages</a> to get the list of available languages |
| theme         | the theme of the editor; run <a href="#">getMonacoThemes</a> to get the list of available themes                                                                                                               |
| tabSize       | number of spaces for the indentation (usually 2 or 4); if NULL, it is set to the one used in RStudio                                                                                                           |
| fontSize      | font size in pixels                                                                                                                                                                                            |
| header        | logical, whether to display the header of the widget                                                                                                                                                           |
| width, height | dimensions; the default values are nice for usage in the RStudio viewer pane                                                                                                                                   |
| elementId     | a HTML id for the container; this is useless for common usage                                                                                                                                                  |

## Examples

```
# in RStudio, `monaco()` opens the current file:  
monaco()  
  
# opens a new, empty JavaScript file:  
monaco(NULL, language = "javascript")  
  
# opens an existing file:  
monaco(system.file("exampleFiles", "JavaScript.js", package = "monaco"))  
  
# try the SVG viewer; you can zoom and pan the image:  
monaco(system.file("exampleFiles", "react.svg", package = "monaco"))  
  
# a dirty Markdown file, try to prettify it:  
monaco(system.file("exampleFiles", "Markdown.md", package = "monaco"))  
  
# opens two editors side-by-side:
```

```

library(monaco)
library(htmltools)

ed1 <- monaco(
  system.file("exampleFiles", "JavaScript.js", package = "monaco")
)
ed2 <- monaco(
  system.file("exampleFiles", "react.svg", package = "monaco")
)

if(interactive()){
  browsable(
    div(
      div(ed1, style="position: fixed; left: 1vw; right: 51vw;"),
      div(ed2, style="position: fixed; left: 51vw; right: 1vw;")
    )
  )
}

# stacks two editors:
library(monaco)
library(htmltools)

ed1 <- monaco(
  system.file("exampleFiles", "JavaScript.js", package = "monaco"),
  height = "calc(50vh - 40px)"
)
ed2 <- monaco(
  system.file("exampleFiles", "react.svg", package = "monaco"),
  height = "calc(50vh - 40px)"
)

if(interactive()){
  browsable(
    tagList(
      tags$style(HTML(
        ".editor {",
        "  position: fixed;",
        "  left: 1vw;",
        "  width: 98vw;",
        "}"
      )),
      div(
        div(ed1, class = "editor", style = "bottom: calc(50vh + 5px);"),
        div(ed2, class = "editor", style = "top: calc(50vh + 5px);")
      )
    )
  )
}

```

## Description

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

## Usage

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

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

## Arguments

|               |                                                                                                                      |
|---------------|----------------------------------------------------------------------------------------------------------------------|
| outputId      | output variable to read from                                                                                         |
| width, height | CSS measurements like "100%", "400px", "auto", or a number, which will be coerced to a string and have "px" appended |
| expr          | an expression that creates a Monaco editor with <a href="#">monaco</a>                                               |
| env           | the environment in which to evaluate expr                                                                            |
| quoted        | logical, whether expr is a quoted expression                                                                         |

## Examples

```
library(monaco)
library(shiny)

ui <- fluidPage(
  monacoOutput("ed", height = "400px")
)

server <- function(input, output){
  output[["ed"]] <- renderMonaco({
    monaco(
      system.file("exampleFiles", "JavaScript.js", package = "monaco")
    )
  })
}

if(interactive()){
  shinyApp(ui, server)
}

# Customizing the input range, using the 'sass' package ####
```

```

library(monaco)
library(shiny)
library(sass)

ui <- fluidPage(

  uiOutput("style"),

  titlePanel("Customized range input"),

  fluidRow(
    column(
      width = 4,
      actionButton("sass", "Compile to CSS", class = "btn-primary btn-block")
    ),
    column(
      width = 8,
      tags$input(type = "range", min = 0, max = 10, step = 0.1)
    )
  ),

  br(),

  fluidRow(
    column(
      width = 6,
      monacoOutput("scss", height = "75vh")
    ),
    column(
      width = 6,
      monacoOutput("css", height = "75vh")
    )
  )
)

server <- function(input, output){

  output[["scss"]] <- renderMonaco({
    monaco(
      system.file(
        "htmlwidgets", "customRangeInput", "customRangeInput.scss",
        package = "monaco"
      ),
      header = FALSE
    )
  })

  css <- eventReactive(input[["sass"]], {
    sass(input[["scss"]])
  })

  output[["css"]] <- renderMonaco({

```

```
    monaco(css(), language = "css", header = FALSE)
  })

  output[["style"]] <- renderUI({
    tags$head(tags$style(HTML(input[["css"]])))
  })
}

if(interactive()){
  shinyApp(ui, server)
}
```

# Index

`getMonacoLanguages`, [2](#), [3](#)

`getMonacoThemes`, [2](#), [3](#)

`monaco`, [2](#), [5](#)

`monaco-shiny`, [5](#)

`monacoOutput` (`monaco-shiny`), [5](#)

`renderMonaco` (`monaco-shiny`), [5](#)