免费电子书

学习

# shiny

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#shiny

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: shiny

It is an unofficial and free shiny ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official shiny.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# 1:

。

。 。

## Examples

Shinyshinyapps.io 。

1. **Shiny**R / RStudioCRAN`install.packages("shiny")` RStudio Github `devtools::install_github("rstudio/shiny")` 。 GithubShinyCRAN。

1. “”。 Web。
2. 。 。
3. Web。 UI。
4. 。 “”“”“”“”。 。

`shiny UI server` 。 UI“Hello world”。

**UI.R**

UIdiv。

```
library(shiny)

# Define UI for application print "Hello world"
shinyUI(

  # Create bootstrap page
  fluidPage(

    # Paragraph "Hello world"
    p("Hello world"),

    # Create button to print "Hello world" from server
    actionButton(inputId = "Print_Hello", label = "Print_Hello World"),

    # Create position for server side text
    textOutput("Server_Hello")

  )
)
```

**Server.R**

。

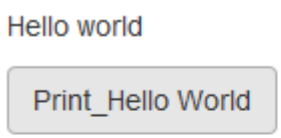
```
# Define server logic required to print "Hello World" when button is clicked
shinyServer(function(input, output) {
```

```
  # Create action when actionButton is clicked
  observeEvent(input$Print_Hello,{

    # Change text of Server_Hello
    output$Server_Hello = renderText("Hello world from server side")
  })


})
```

1. `runApp('your dir path')`
2. uiserver`shinyApp(ui,server)`





shinyApp`plotOutput`UI`renderPlot。 ggPlot`

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  plotOutput('myPlot'),
  plotOutput('myGgPlot')
)

server <- function(input, output, session){
  output$myPlot = renderPlot({
    hist(rnorm(1000))
  })
  output$myGgPlot <- renderPlot({
    ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point()
  })
}

shinyApp(ui, server)
```

DT DTJavaScriptDataTablesR。

```
library(shiny)
library(DT)
```

```
ui <- fluidPage(
  dataTableOutput('myTable')
)

server <- function(input, output, session){
  output$myTable <- renderDataTable({
    datatable(iris)
  })
}

shinyApp(ui, server)
```

https://riptutorial.com/zh-CN/shiny/topic/2667/

# 2: Javascript API

- session $ sendCustomMessage
- Shiny.addCustomMessageHandler *JS*
- Shiny.onInputChange

## Examples

RJS。

```
library(shiny)
runApp(
  list(
    ui = fluidPage(
      tags$script(
        "Shiny.addCustomMessageHandler('message', function(params) { alert(params); });"
      ),
      actionButton("btn","Press Me")
    ),
    server = function(input, output, session) {
      observeEvent(input$btn,{
        randomNumber <- runif(1,0,100)
        session$sendCustomMessage("message",list(paste0(randomNumber," is a random number!")))
      })
    }
  )
)
```

`Rsession$sendCustomMessagejavascriptShiny.addCustomMessageHandler`。

`session$sendCustomMessageRjavascriptShiny.addCustomMessageHandlerRjavascript`。

`Rjavascript`JSON

JSR。 javascript`Shiny.onInputChange`

```
library(shiny)
runApp(
  list(
    ui = fluidPage(
      # create password input
      HTML('<input type="password" id="passwordInput">'),
      # use jquery to write function that sends value to
      # server when changed
      tags$script(
        '$("#passwordInput").on("change",function() {
          Shiny.onInputChange("myInput",this.value);
        })'
      ),
      # show password
      verbatimTextOutput("test")
    ),
    server = function(input, output, session) {
```

```
      # read in then show password
      output$test <- renderPrint(
        input$myInput
      )
    }
  )
)
```

id `passwordInput`。 UIJavascript`passwordInputShiny.onInputChange`。

`Shiny.onInputChange input$*name* input$*name*`

`input$*name*`Shiny。

Javascript API https://riptutorial.com/zh-CN/shiny/topic/3149/javascript-api

# 3: reactivereactiveValueeventReactiveShiny

**reactivereactiveValueeventReactive**Shiny。。

**observeobserveEvent**。。

。

## Examples

。$ texttext_reactivetext_reactiveinput $ user_text。$ user_text$ texttext_reactive。input $ user_text。

```
library(shiny)

ui <- fluidPage(
  headerPanel("Example reactive"),

  mainPanel(

    # input field
    textInput("user_text", label = "Enter some text:", placeholder = "Please enter some
text."),

    # display text output
    textOutput("text"))
)

server <- function(input, output) {

  # reactive expression
  text_reactive <- reactive({
    input$user_text
  })

  # text output
  output$text <- renderText({
    text_reactive()
  })
}

shinyApp(ui = ui, server = server)
```

### eventReactive

eventReactives

```
eventReactive( event {
code to run
})
```

eventReactives“ ”。。

eventReactive。$ user_texteventReactiveeventReactiveactionButton$ submit。text_reactive$ text $ user_text。

```
library(shiny)

ui <- fluidPage(
  headerPanel("Example eventReactive"),

  mainPanel(

    # input field
    textInput("user_text", label = "Enter some text:", placeholder = "Please enter some
text."),

    # submit button
    actionButton("submit", label = "Submit"),

    # display text output
    textOutput("text"))
)

server <- function(input, output) {

  # reactive expression
  text_reactive <- eventReactive( input$submit, {
    input$user_text
  })

  # text output
  output$text <- renderText({
    text_reactive()
  })
}

shinyApp(ui = ui, server = server)
```

## reactiveValues

reactiveValues。

reactiveValues“。”。reactiveValues。reactiveValues。

```
library(shiny)

ui <- fluidPage(
  headerPanel("Example reactiveValues"),

  mainPanel(

    # input field
    textInput("user_text", label = "Enter some text:", placeholder = "Please enter some
text."),
    actionButton("submit", label = "Submit"),

    # display text output
    textOutput("text"))
)
```

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

  # observe event for updating the reactiveValues
  observeEvent(input$submit,
               {
    text_reactive$text <- input$user_text
  })

  # reactiveValues
  text_reactive <- reactiveValues(
    text = "No text has been submitted yet."
  )

  # text output
  output$text <- renderText({
    text_reactive$text
  })
}

shinyApp(ui = ui, server = server)
```

## observeEvent

observeEvent。

```
observeEvent( event {
code to run
})
```

observeEvent*“event”*。“ ”。

```
library(shiny)

ui <- fluidPage(
  headerPanel("Example reactive"),

  mainPanel(

    # action buttons
    actionButton("button1","Button 1"),
    actionButton("button2","Button 2")
  )
)

server <- function(input, output) {

  # observe button 1 press.
  observeEvent(input$button1, {
    # The observeEvent takes no dependency on button 2, even though we refer to the input in
the following line.
    input$button2
    showModal(modalDialog(
      title = "Button pressed",
      "You pressed one of the buttons!"
    ))
  })
```

```
}

shinyApp(ui = ui, server = server)
```

◦ reactiveValues◦

observeNULLNULL◦ observeEventNULL◦

```
library(shiny)

ui <- fluidPage(
  headerPanel("Example reactive"),

  mainPanel(

    # action buttons
    actionButton("button1","Button 1"),
    actionButton("button2","Button 2")
  )
)

server <- function(input, output) {

  # observe button 1 press.
  observe({
    input$button1
    input$button2
    showModal(modalDialog(
      title = "Button pressed",
      "You pressed one of the buttons!"
    ))
  })
}

shinyApp(ui = ui, server = server)
```

reactivereactiveValueeventReactiveShiny https://riptutorial.com/zh-CN/shiny/topic/10787/reactive-reactivevalueeventreactive-shiny

# 4: MCVE

Shiny。 。 I / O ID。

MCVE。

## Examples

MCVEShiny。 `shinyApp`。

```
library(shiny)

ui <- fluidPage(
  checkboxInput('checkbox', 'click me'),
  verbatimTextOutput('text')
)

server <- function(input, output, session){
  output$text <- renderText({
    isolate(input$checkbox)
  })
}

shinyApp(ui, server)
```

`uiserver`。

```
library(shiny)

shinyApp(
  fluidPage(
    checkboxInput('checkbox', 'click me'),
    verbatimTextOutput('text')
  ),
  function(input, output, session){
    output$text <- renderText({
      isolate(input$checkbox)
    })
  }
)

shinyApp(ui, server)
```

。 。 MCVE。

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  plotOutput('plot')
)
```

```
server <- function(input, output, session){
  df <- data.frame(treatment = rep(letters[1:3], times = 3),
                   context = rep(LETTERS[1:3], each = 3),
                   effect = runif(9,0,1))
  df$treat.con <- paste(df$treatment,df$context, sep = ".")
  df$treat.con <- reorder(df$treat.con, -df$effect, )
  output$plot = renderPlot({
    myPlot <- ggplot(df, aes(x = treat.con, y = effect)) +
       geom_point() +
       facet_wrap(~context,
                  scales = "free_x",
                  ncol = 1)
  })
}

shinyApp(ui, server)
```

---

### Plot

```
library(shiny)
library(ggplot2)

ui <- fluidPage(
  plotOutput('plot')
)

server <- function(input, output, session){
  output$plot = renderPlot({
    myPlot <- ggplot(mtcars, aes(mpg, wt)) + geom_point()
  })
}

shinyApp(ui, server)
```

MCVE https://riptutorial.com/zh-CN/shiny/topic/10653/mcve---

# 5:

## Examples

### fileInput.RData

.RData。 `loadget`。 “”。

```
library(shiny)

# Define two datasets and store them to disk
x <- rnorm(100)
save(x, file = "x.RData")
rm(x)
y <- rnorm(100, mean = 2)
save(y, file = "y.RData")
rm(y)

# Define UI
ui <- shinyUI(fluidPage(
  titlePanel(".RData File Upload Test"),
  mainPanel(
    fileInput("file", label = ""),
    actionButton(inputId="plot","Plot"),
    plotOutput("hist"))
  )
)

# Define server logic
server <- shinyServer(function(input, output) {

  observeEvent(input$plot,{
    if ( is.null(input$file)) return(NULL)
    inFile <- input$file
    file <- inFile$datapath
    # load the file into new environment and get it from there
    e = new.env()
    name <- load(file, envir = e)
    data <- e[[name]]

    # Plot the data
    output$hist <- renderPlot({
      hist(data)
    })
  })
})

# Run the application
shinyApp(ui = ui, server = server)
```

### csvShiny

csvShiny。 。 radioButton。

```
library(shiny)
library(DT)

# Define UI
ui <- shinyUI(fluidPage(

  fileInput('target_upload', 'Choose file to upload',
            accept = c(
              'text/csv',
              'text/comma-separated-values',
              '.csv'
            )),
  radioButtons("separator","Separator: ",choices = c(";",",","","":"), selected=";",inline=TRUE),
  DT::dataTableOutput("sample_table")
)
)

# Define server logic
server <- shinyServer(function(input, output) {

  df_products_upload <- reactive({
    inFile <- input$target_upload
    if (is.null(inFile))
      return(NULL)
    df <- read.csv(inFile$datapath, header = TRUE,sep = input$separator)
    return(df)
  })

  output$sample_table<- DT::renderDataTable({
    df <- df_products_upload()
    DT::datatable(df)
  })

}
)

# Run the application
shinyApp(ui = ui, server = server)
```

https://riptutorial.com/zh-CN/shiny/topic/7576/

| S. No | | Contributors |
|---|---|---|
| 1 | | Batanichek, Bogdan Rau, chrki, Community, Florian, Gregor de Cillia, jsb, Mathias711, micstr, sigmabeta |
| 2 | Javascript API | Carl, Tomás Barcellos |
| 3 | reactive<br>reactiveValue<br>eventReactiveShiny | Florian |
| 4 | MCVE | Florian, Gregor de Cillia |
| 5 | | Florian, symbolrush |