Challenge-8

Marzuki Nooranas

2023-10-09

## Code Along Example

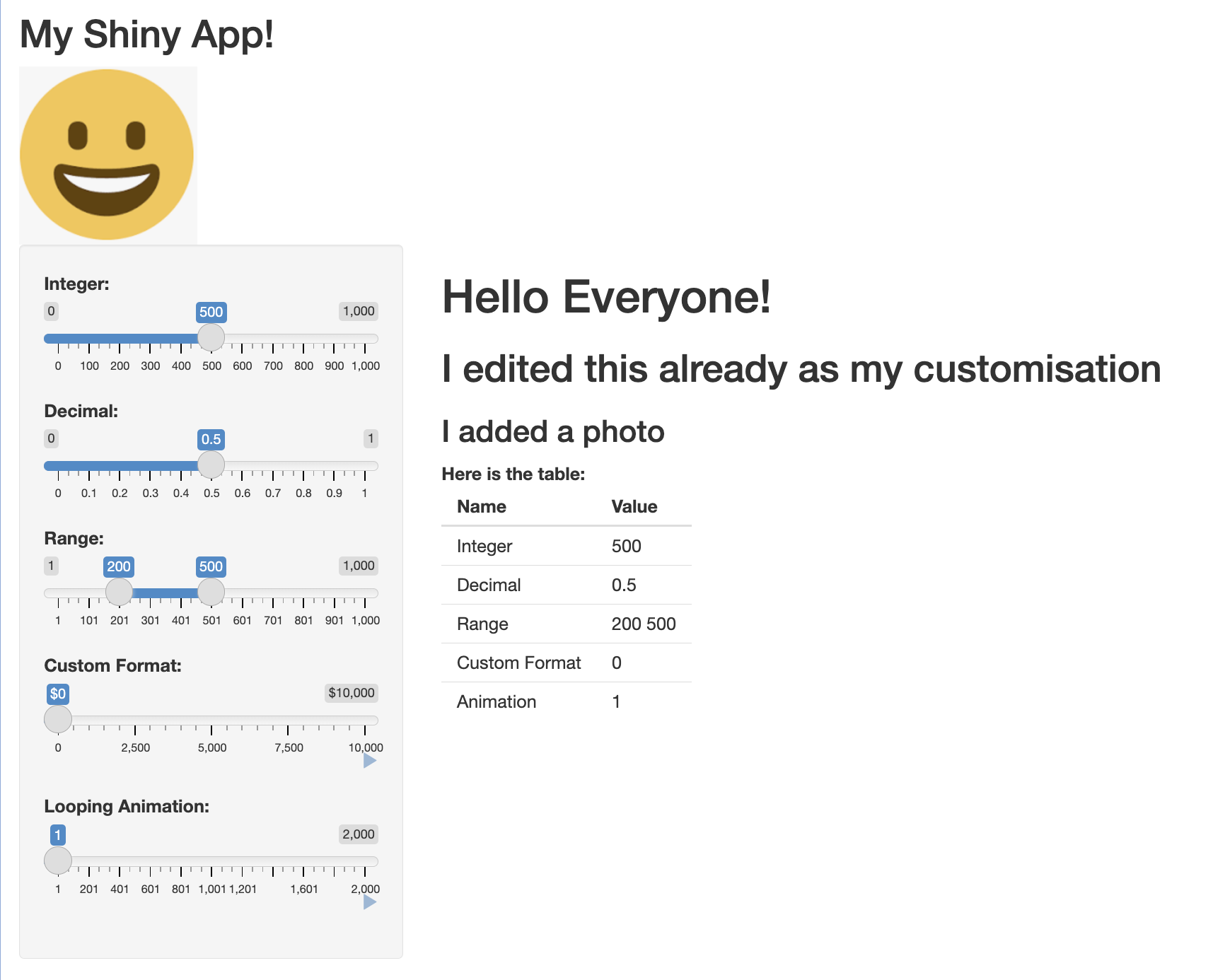
knitr::include\_graphics("/Users/marzuki/Desktop/NM2207/NM2207/Week-8/code\_along.png")



library(shiny)  
  
# Define UI for app that draws a histogram ----  
ui <- fluidPage(  
   
 # App title ----  
 titlePanel("Hello Everyone!"),  
   
 # Sidebar layout with input and output definitions ----  
 sidebarLayout(  
   
 # Sidebar panel for inputs ----  
 sidebarPanel(  
   
 # Input: Slider for the number of bins ----  
 sliderInput(inputId = "bins",  
 label = "Number of bins:",  
 min = 5,  
 max = 50,  
 value = 30)  
   
 ),  
   
 # Main panel for displaying outputs ----  
 mainPanel(  
   
 # Output: Histogram ----  
 plotOutput(outputId = "distPlot")  
   
 )  
 )  
)  
  
# Define server logic required to draw a histogram ----  
server <- function(input, output) {  
   
 # Histogram of the Old Faithful Geyser Data ----  
 # with requested number of bins  
 # This expression that generates a histogram is wrapped in a call  
 # to renderPlot to indicate that:  
 #  
 # 1. It is "reactive" and therefore should be automatically  
 # re-executed when inputs (input$bins) change  
 # 2. Its output type is a plot  
 output$distPlot <- renderPlot({  
   
 x <- faithful$waiting  
 bins <- seq(min(x), max(x), length.out = input$bins + 1)  
   
 hist(x, breaks = bins, col = "#75AADB", border = "yellow",  
 xlab = "Waiting time to next eruption (in mins)",  
 main = "Histogram of waiting times")  
   
 })  
   
}  
  
# Create Shiny app ----  
shinyApp(ui = ui, server = server)

## Challenge 8

knitr::include\_graphics("/Users/marzuki/Desktop/NM2207/NM2207/Week-8/challenge.png")



library(shiny)  
  
# Define UI for slider demo app ----  
ui <- fluidPage(  
   
 # App title ----  
 titlePanel("My Shiny App!"),  
 img(src = "face.png", height = 140, width = 140),  
   
 # Sidebar layout with input and output definitions ----  
 sidebarLayout(  
   
 # Sidebar to demonstrate various slider options ----  
 sidebarPanel(  
   
 # Input: Simple integer interval ----  
 sliderInput("integer", "Integer:",  
 min = 0, max = 1000,  
 value = 500),  
   
 # Input: Decimal interval with step value ----  
 sliderInput("decimal", "Decimal:",  
 min = 0, max = 1,  
 value = 0.5, step = 0.1),  
   
 # Input: Specification of range within an interval ----  
 sliderInput("range", "Range:",  
 min = 1, max = 1000,  
 value = c(200,500)),  
   
 # Input: Custom currency format for with basic animation ----  
 sliderInput("format", "Custom Format:",  
 min = 0, max = 10000,  
 value = 0, step = 2500,  
 pre = "$", sep = ",",  
 animate = TRUE),  
   
 # Input: Animation with custom interval (in ms) ----  
 # to control speed, plus looping  
 sliderInput("animation", "Looping Animation:",  
 min = 1, max = 2000,  
 value = 1, step = 10,  
 animate =  
 animationOptions(interval = 300, loop = TRUE))  
   
 ),  
   
 # Main panel for displaying outputs ----  
 mainPanel(  
 h1("Hello Everyone!"),  
 h2("I edited this already as my customisation"),  
 h3("I added a photo"),  
 strong("Here is the table:"),  
 # Output: Table summarizing the values entered ----  
 tableOutput("values")  
   
 )  
 )  
)  
  
# Define server logic for slider examples ----  
server <- function(input, output) {  
   
 # Reactive expression to create data frame of all input values ----  
 sliderValues <- reactive({  
   
 data.frame(  
 Name = c("Integer",  
 "Decimal",  
 "Range",  
 "Custom Format",  
 "Animation"),  
 Value = as.character(c(input$integer,  
 input$decimal,  
 paste(input$range, collapse = " "),  
 input$format,  
 input$animation)),  
 stringsAsFactors = FALSE)  
   
 })  
   
 # Show the values in an HTML table ----  
 output$values <- renderTable({  
 sliderValues()  
 })  
   
}  
  
# Create Shiny app ----  
shinyApp(ui, server)