Reactive apps with shiny

2019-08-29

UI

```
sliderInput(
  inputId = "alpha",
  label = "Alpha:",
  min = 0,
  max = 1,
  value = 0.5
)
```

Server

```
ggplot(...) +
  geom_point(alpha = input$alpha)
```

Inputs are reactive

```
# in server <- function(input, output) {}
output$scatterplot <- renderPlot({
  ggplot(
    data = movies,
    aes_string(x = input$x, y = input$y, color = input$z)
  ) +
    geom_point(alpha = input$alpha)
})
```

Inputs are reactive

```
# in server <- function(input, output) {}
output$scatterplot <- renderPlot({
  ggplot(
    data = movies,
    aes_string(x = input$x, y = input$y, color = input$z)
  ) +
    geom_point(alpha = input$alpha)
})
```

When the user changes the alpha **slider,** input$alpha **changes.**

Reactivity

Shiny watches reactive objects like input$alpha

Reactivity

Shiny watches reactive objects like input$alpha

Shiny **caches** and **updates** reactives

Reactivity

Shiny watches reactive objects like input$alpha

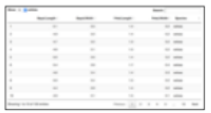
Shiny caches and updates reactives

render*() **functions are reactive contexts**

Render functions

Outputs - render*() and *Output() functions work together to add R output to the UI

works with

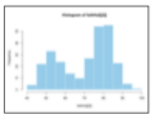


DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)

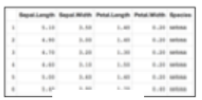


renderPlot(expr, width, height, res, …, env, quoted, func)

plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, inline, hoverDelayType, brush, clickId, hoverId)

renderPrint(expr, env, quoted, func, width)

verbatimTextOutput(outputId)



renderTable(expr,…, env, quoted, func)

tableOutput(outputId)

foo

renderText(expr, env, quoted, func)

textOutput(outputId, container, inline)

renderUI(expr, env, quoted, func)

&

uiOutput(outputId, inline, container, …)

htmlOutput(outputId, inline, container, …)

Your Turn 1

Start with movies_05.R

Add a new slider to the UI to select size **of the points from 0 to 5**

Add the value to geom_point()

Run the app and update the points

Your Turn 1 (solutions in movies_06.R)

```
sliderInput(
  inputId = "size",
  label = "Size:",
  min = 0,
  max = 5,
  value = 2
)
```

```
ggplot(...) +
  geom_point(alpha = input$alpha, size = input$size)
```

Making data reactive

```
checkboxGroupInput(
  inputId = "selected_type",
  label = "Select movie type(s)",
  choices = c("Documentary", "Feature Film", "TV Movie"),
  selected = "Feature Film"
)
```

Making data reactive

```
movies_subset <- reactive({
  req(input$selected_type)
  filter(
    movies, title_type %in% input$selected_type
  )
})
```

Making data reactive

```
movies_subset <- reactive({
  req(input$selected_type)
  filter(
    movies, title_type %in% input$selected_type
  )
})
```

Making data reactive

```
movies_subset <- reactive({
  req(input$selected_type)
  filter(
    movies, title_type %in% input$selected_type
  )
})
```

Making data reactive

```
movies_subset <- reactive({
  req(input$selected_type)
  filter(
    movies, title_type %in% input$selected_type
  )
})

movies_subset()
```

Making data reactive

```
movies_subset <- reactive({
  req(input$selected_type)
  filter(
    movies, title_type %in% input$selected_type
  )
})

movies_subset()
movies_subset()$title_type
```

Making data reactive

```
output$scatterplot <- renderPlot({
  ggplot(
    data = movies_subset(),
    aes_string(...),
  ) +
  ...
})
```

Making data reactive

```
# in mainPanel()
uiOutput("n")
```

```
# in server
output$n <- renderUI({
  types <- movies_subset()$title_type %>%
    factor(levels = input$selected_type)
  counts <- table(types)

  HTML(
    paste(
      "There are",
      counts,
      input$selected_type,
      "movies in this dataset. <br>"
    )
  )
})
```

Making data reactive

```
# in mainPanel()
uiOutput("n")
```

```
# in server
output$n <- renderUI({
  types <- movies_subset()$title_type %>%
    factor(levels = input$selected_type)
  counts <- table(types)

  HTML(
    paste(
      "There are",
      counts,
      input$selected_type,
      "movies in this dataset. <br>"
    )
  )
})
```

Making data reactive

```
# in mainPanel()
uiOutput("n")
```

```
# in server
output$n <- renderUI({
  types <- movies_subset()$title_type %>%
    factor(levels = input$selected_type)
  counts <- table(types)

  HTML(
    paste(
      "There are",
      counts,
      input$selected_type,
      "movies in this dataset. <br>"
    )
  )
})
```

Making data reactive

Open movies-07.R

Your Turn 2

Update the data table to use the reactive version of the data

Run the app

Your Turn 2 (solution: movies-08.R)

```
# in server
output$moviestable <- DT::renderDataTable({
  DT::datatable(
    data = movies_subset()[, 1:7],
    options = list(pageLength = 10),
    rownames = FALSE
  )
})
```

Your Turn 3

Create a new reactive data set called movies_sample**. Require** input$n_samp**. Return a sample of** movies_subset() **with** sample_n()

Use movies_sample() **as the data for your plot, table, and UI outputs.**

Create the corresponding UI input for input$n_samp **with** numericInput()**: set** min **to 1 and** max **to** nrows(movies)**. Set the default value to 50.**

Run the app

Your Turn 3 (solution: movies-09.R)

```
# in ui
numericInput(
  inputId = "n_samp",
  label = "Sample size:",
  min = 1, max = nrow(movies),
  value = 50
)
```

```
# in server
movies_sample <- reactive({
  req(input$n_samp)
  sample_n(movies_subset(), input$n_samp)
})
```

Your Turn 4

movies-10.R **tries to add a dynamic plot title, but it doesn't work. Fix it using** reactive().

Your Turn 4 (solution: movies-11.R)

```
# in server
pretty_plot_title <- reactive({
  str_to_title(input$plot_title)
})
```

```
# in server
output$scatterplot <- renderPlot({
  ggplot(...) +
    labs(
      ...,
      title = pretty_plot_title()
    )
})
```

Show of hands

Does updating the title update the data sample?

Reactivity is lazy

```
# in server
movies_subset <- reactive({
  req(input$selected_type)
  filter(movies, title_type %in% input$selected_type)
})
```

```
# in server
movies_sample <- reactive({
  req(input$n_samp)
  sample_n(movies_subset(), input$n_samp)
})
```

```
# in server
pretty_plot_title <- reactive({
  str_to_title(input$plot_title)
})
```

Reactivity is lazy

```
# in server
movies_subset <- reactive({
  req(input$selected_type)
  filter(movies, title_type %in% input$selected_type)
})
```

```
# in server
movies_sample <- reactive({
  req(input$n_samp)
  sample_n(movies_subset(), input$n_samp)
})
```

```
# in server
pretty_plot_title <- reactive({
  str_to_title(input$plot_title)
})
```

Reactivity is lazy

```
# in server
movies_subset <- reactive({
  req(input$selected_type)
  filter(movies, title_type %in% input$selected_type)
})
```

```
# in server
movies_sample <- reactive({
  req(input$n_samp)
  sample_n(movies_subset(), input$n_samp)
})
```

```
# in server
pretty_plot_title <- reactive({
  str_to_title(input$plot_title)
})
```

Reactivity is lazy

```
# in server
movies_subset <- reactive({
  req(input$selected_type)
  filter(movies, title_type %in% input$selected_type)
})
```

```
# in server
movies_sample <- reactive({
  req(input$n_samp)
  sample_n(movies_subset(), input$n_samp)
})
```

```
# in server
pretty_plot_title <- reactive({
  str_to_title(input$plot_title)
})
```

Resources

Shiny Website: A collection of articles on Shiny

Mastering Shiny: A Work-in-progress book from Hadley Wickham