Shiny Apps with R

Subha Srinivasagan

Nutrient Management Spear Program, Cornell University

Overview

1. Shiny Apps – Overview
2. User Interface (UI)
3. Server (Reactive Programming)
4. Adding Image
5. Additing Button
6. Deploying Shiny Apps
7. Resources
8. Example – Simple Calculator

1. R Shiny Overview

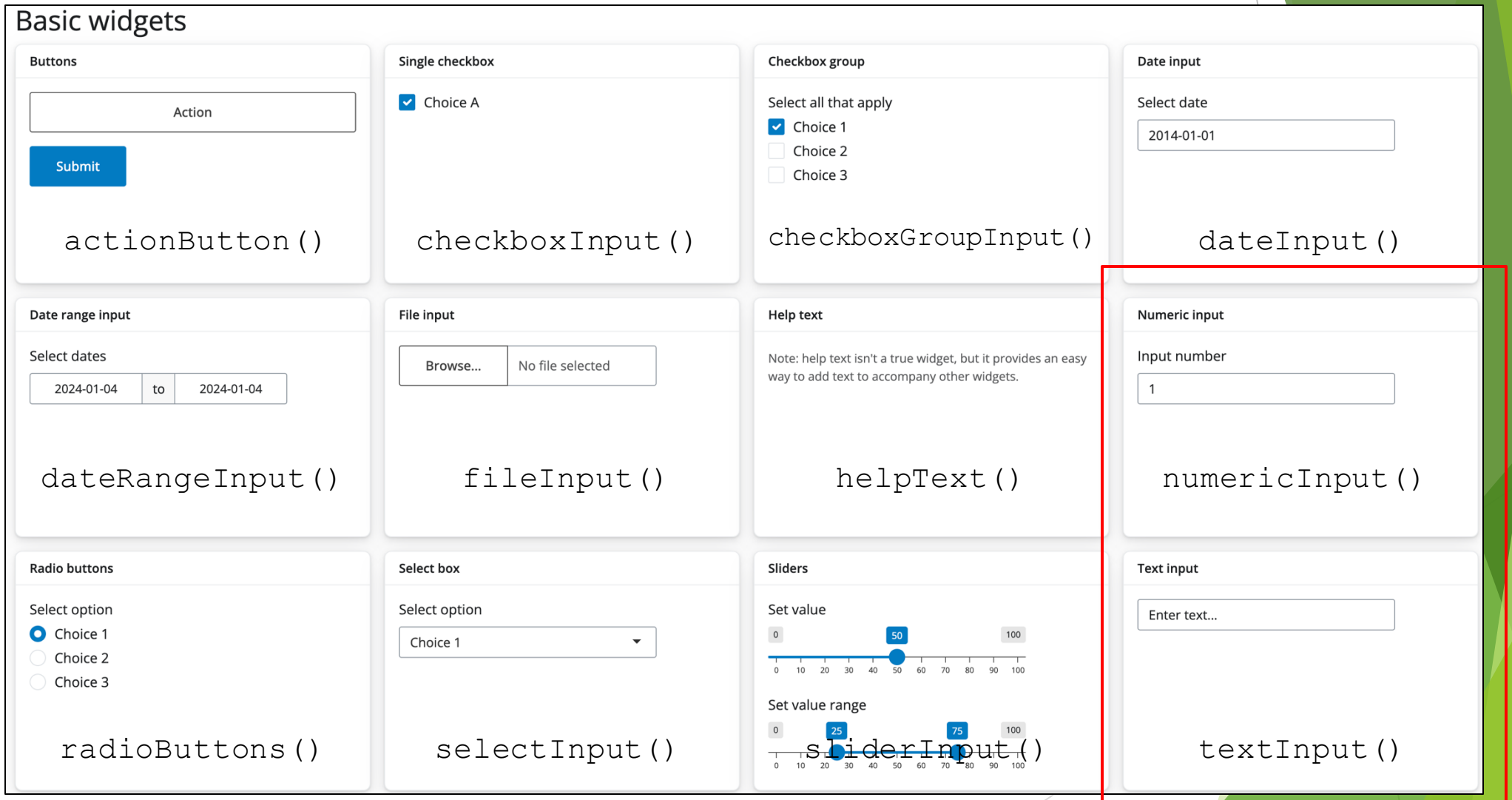
- Rstudio product – Shiny
- Interactive Web application framework for R
- Not code heavy – HTML, CSS, and JavaScript
- Three basic components
  - User-interface
  - Server function
  - A call to Shiny app function

```
# Define user interface for application
ui <- fluidPage(
)

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

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

2. User-interface (UI)

- Application’s front-end appearance
- Collection of inputs and outputs
  - Collects inputs from the users
  - Creates a placeholder to show the outputs
- Inputs and outputs carry a reference ID that is used in server

2. User-interface (UI)

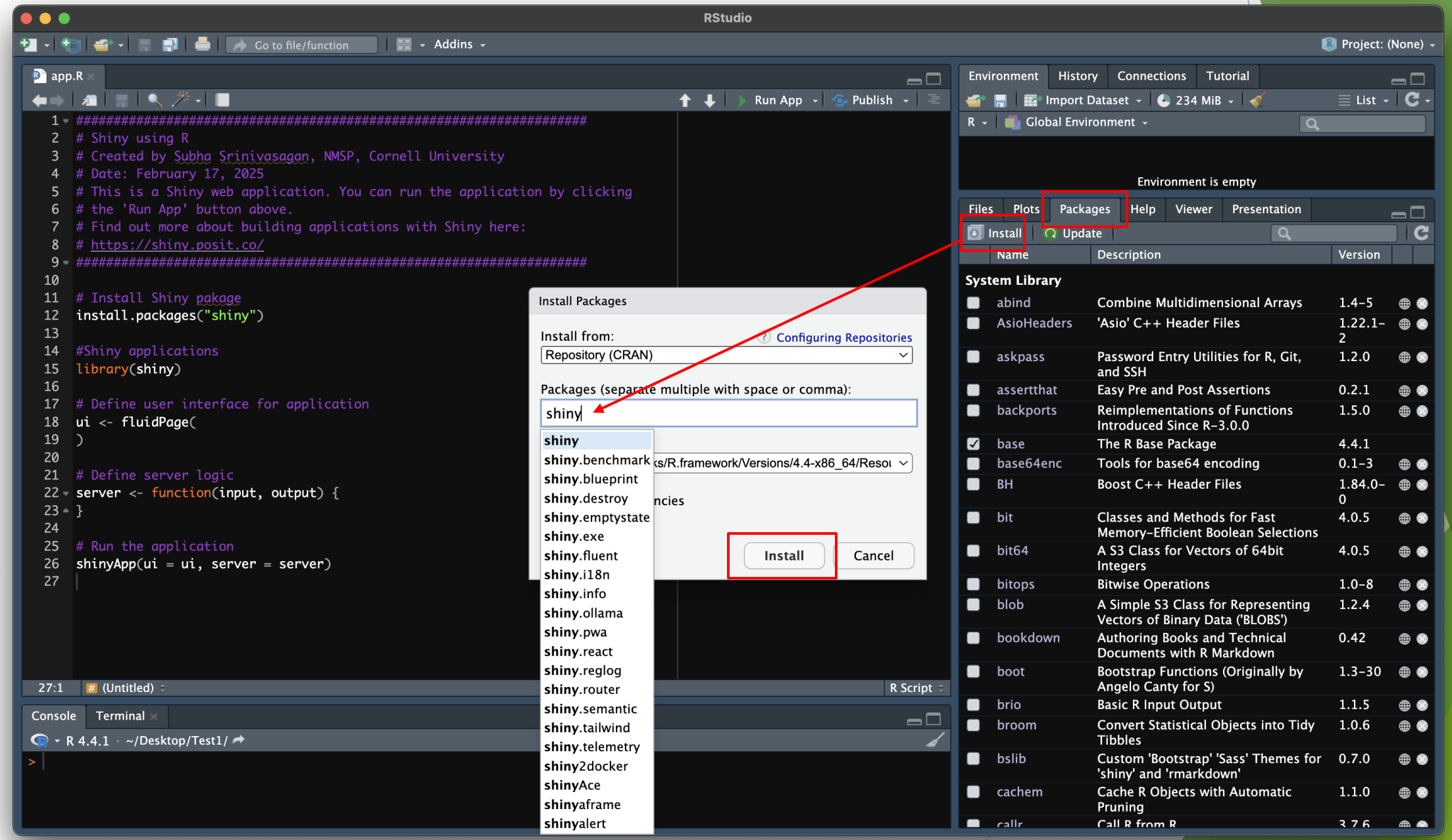


2. User-interface (UI)

```
textInput (“id”, “text”)

numericInput(“id”, “text”, value, min, max)
```

Let’s create our first web application!

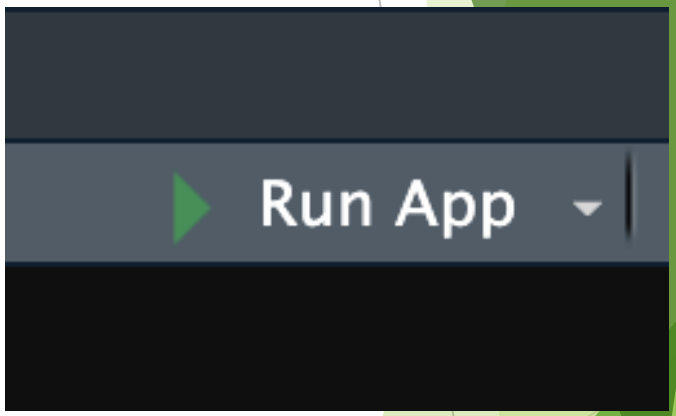
Open file – /Users/home/Desktop/Example1/app.R (GitHub link)



2. User-interface (UI)

```
textInput ("id", "text")
numericInput("id", "text", value, min, max)
```

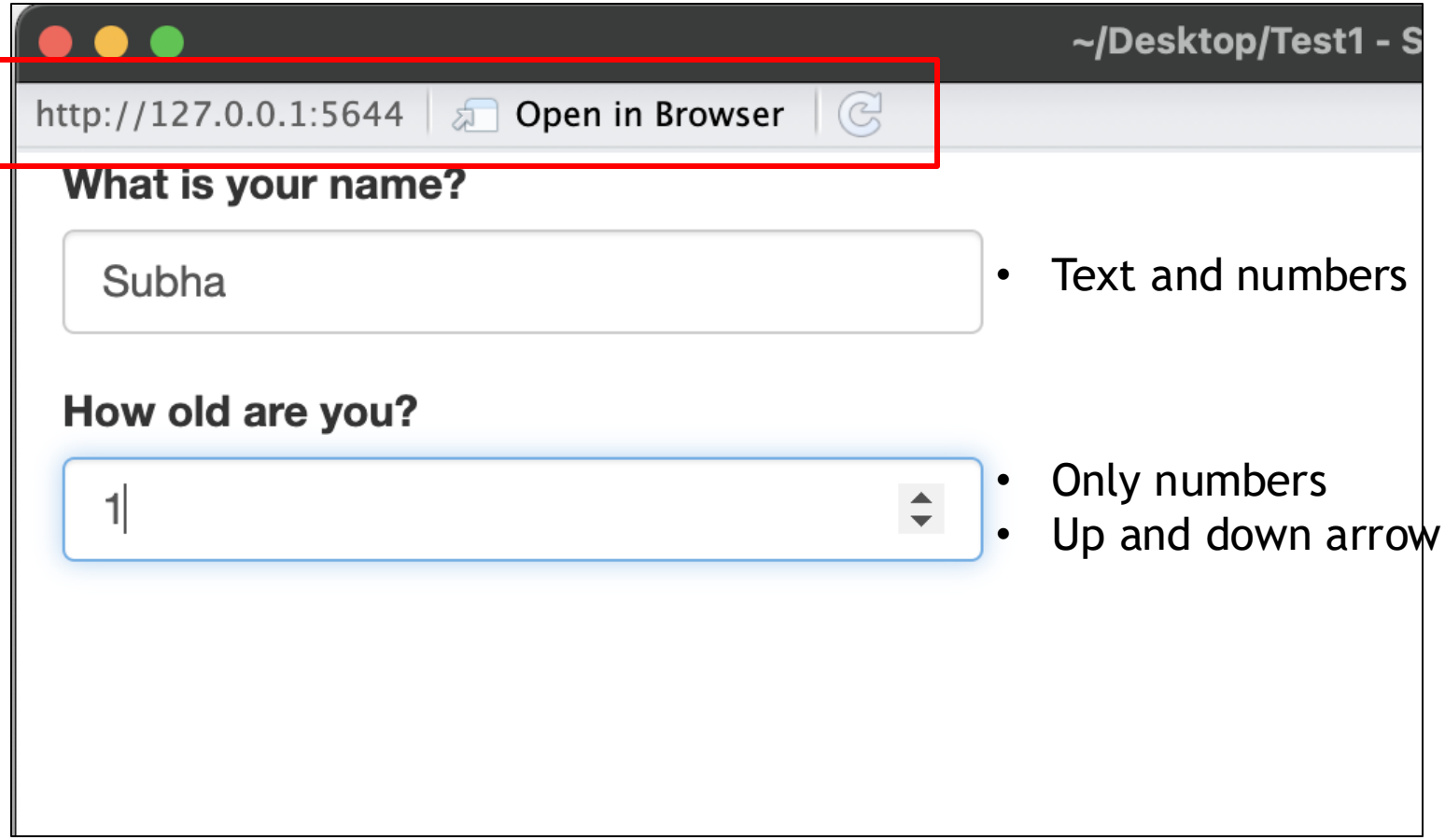
```
# Define user interface for application
ui <- fluidPage(

  # Taking inputs
  textInput("name", "What is your name?"),
  numericInput("age", "How old are you?", value = NULL, min = 0),

)
```



Don't forget!

2. User-interface (UI)

Local server



- Text and numbers
- Only numbers
- Up and down arrow

3. Server (Reactive Programming)

- Application’s back end logic
- Creates and renders output based on inputs from UI
- Directly render or react to actions
- Render functions should match the input functions

3. Server (Reactive Programming)

```
# Define user interface for application
ui <- fluidPage(

  # Taking inputs
  textInput("name", "What is your name?"),
  numericInput("age", "How old are you?", value = NULL, min = 0),

  # Placeholder for output
  textOutput("text"),

)
```

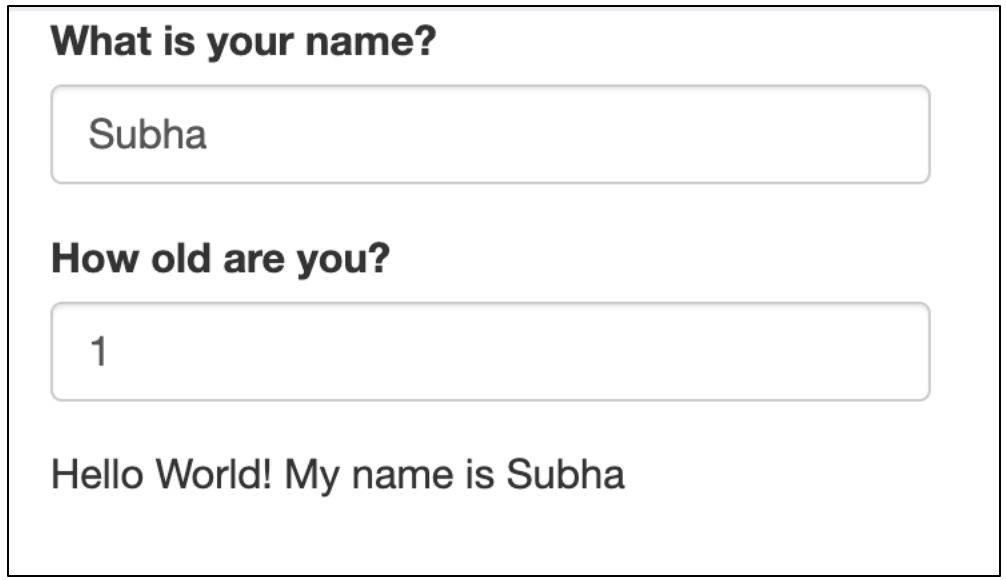
id

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

  output$text <- renderText("Hello World! My name is")

}
```

3. Server (Reactive Programming)

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

  output$text <- renderText(paste("Hello World! My name is",input$name))

}
```

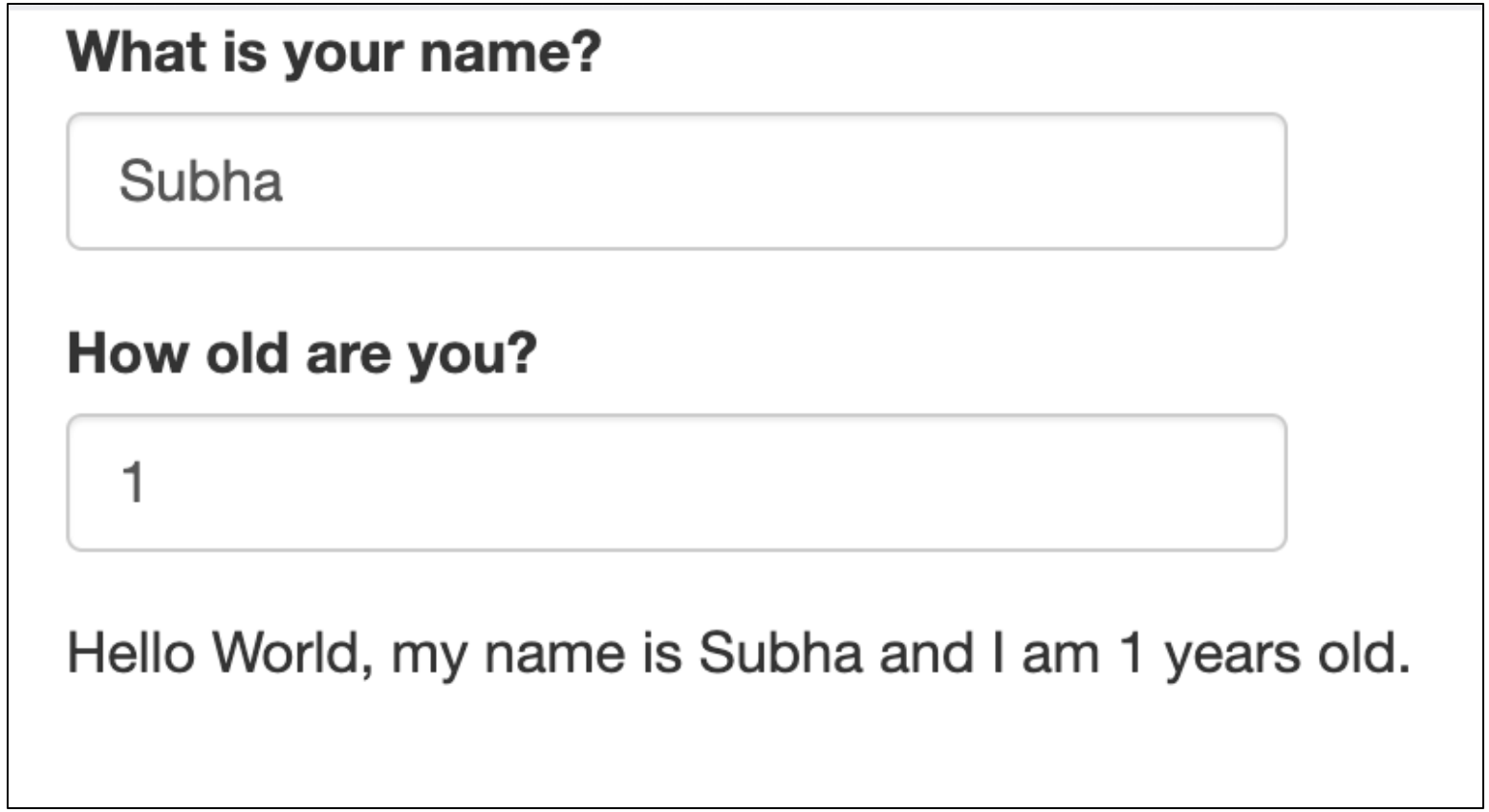
paste("string",variable)




```
# Taking inputs
textInput("name", "What is your name?"),
```

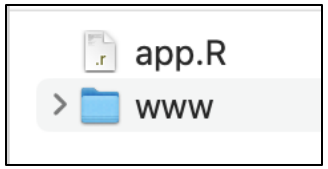
id

3. Server (Reactive Programming)

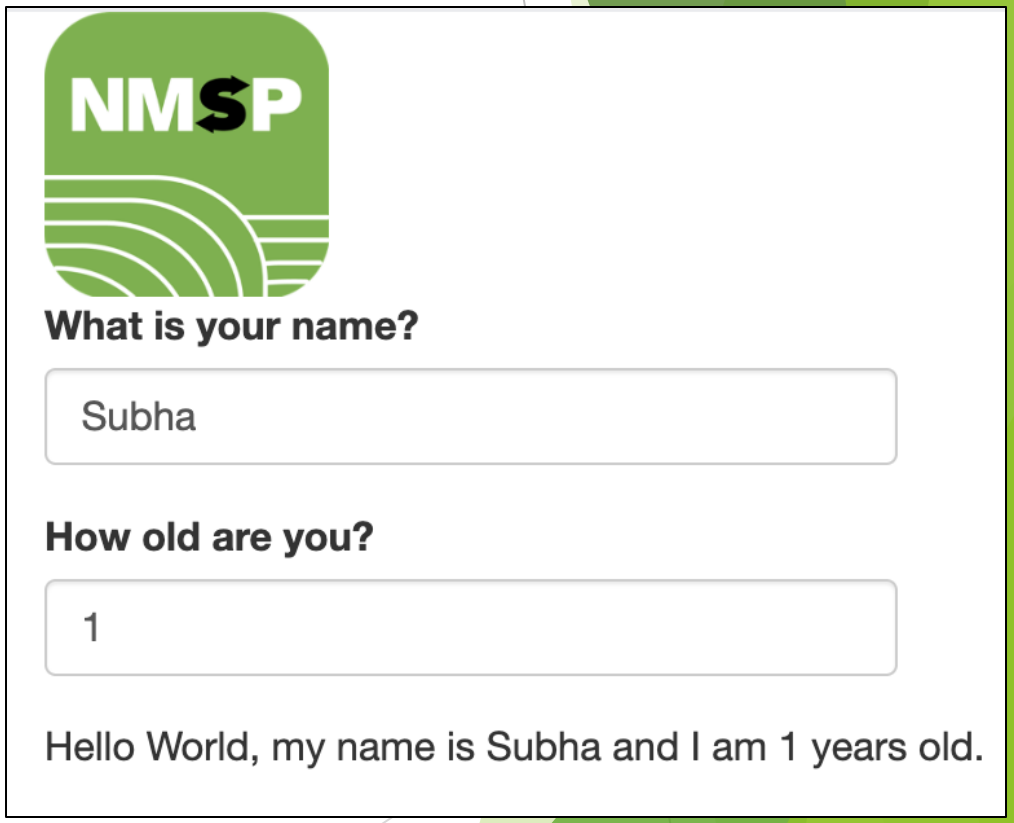


4. Adding Image

Images are required to be stored in a folder names www




```
# Define user interface for application
ui <- fluidPage(

  # Displaying image
  img(src = "NMSP.png", height = "100px", width = "100px"),

  # Taking inputs
  textInput("name", "What is your name?"),
  numericInput("age", "How old are you?", value = NULL, min = 0),

  # Placeholder for output
  textOutput("text")

)
```



5. Adding Button

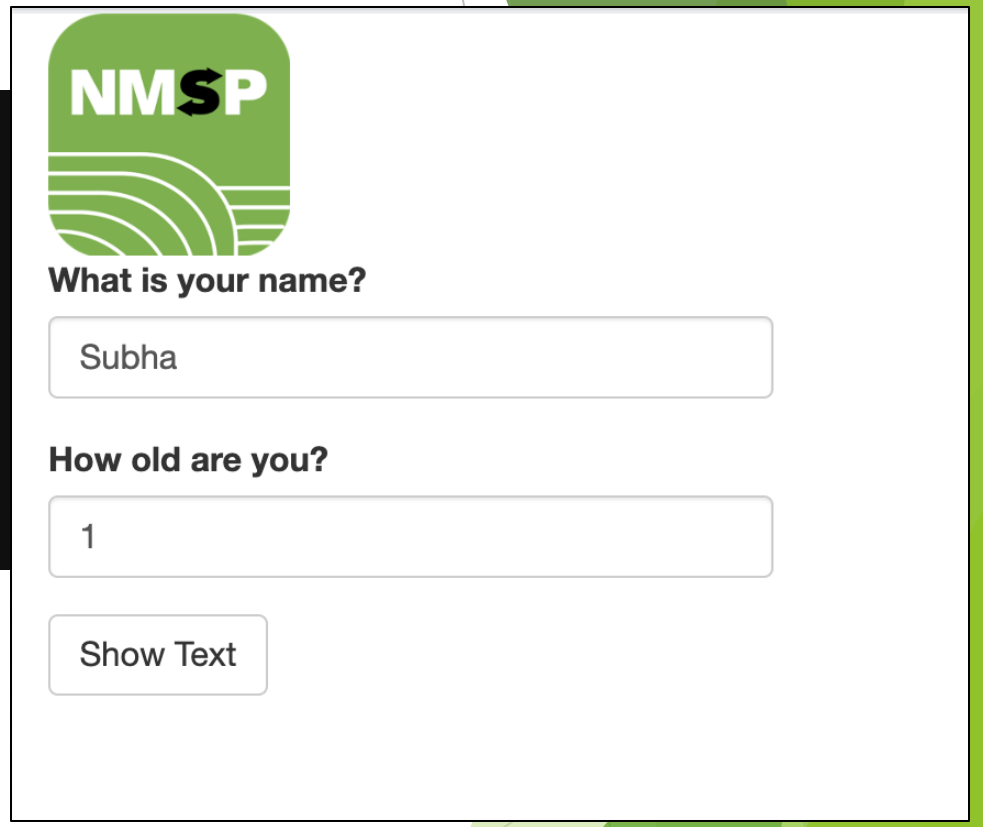
Invokes the reactivity action (not just render) from the server


```r
# Define user interface for application
ui <- fluidPage(

  # Displaying image
  img(src = "NMSP.png", height = "100px", width = "100px"),

  # Taking inputs
  textInput("name", "What is your name?"),
  numericInput("age", "How old are you?", value = NULL, min = 0),

  # Action button to trigger text display
  actionButton("show_text", "Show Text"),

  # Placeholder for output
  textOutput("text")

)
```

5. Adding Button

Invokes the reactivity action (not just render) from the server

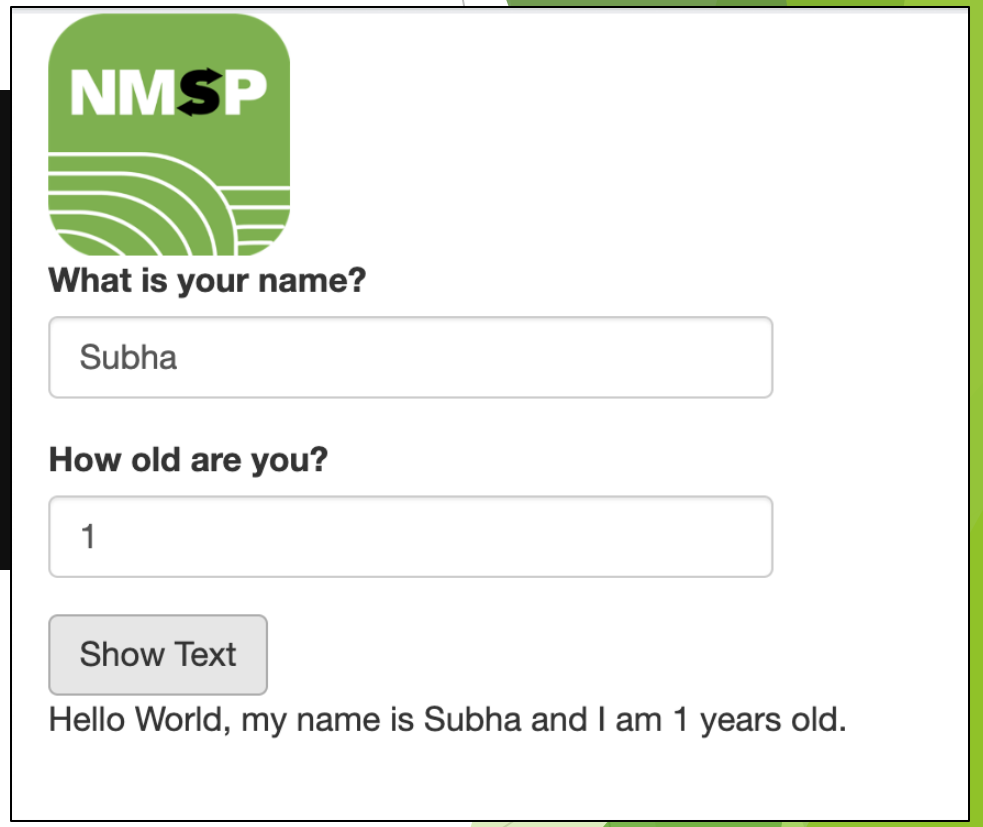
```
# Define server logic
server <- function(input, output) {

  #Reactive value to store button click event
  text_reactive <- eventReactive(input$show_text, {
    paste("Hello World, my name is", input$name, "and I am", input$age, "years old.")
  })

  output$text <- renderText(text_reactive())

}
```



5. Adding Button

Invokes the reactivity action (not just render) from the server

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

  #Reactive value to store button click event
  text_reactive <- eventReactive(input$show_text, {
    paste("Hello World, my name is", input$name, "and I am", input$age, "years old.")
  })

  output$text <- renderText(text_reactive())

}
```

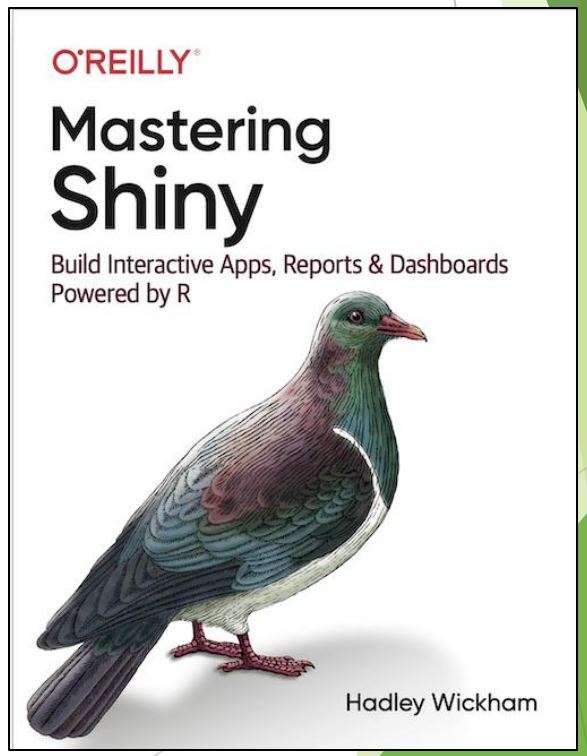


6. Deploying Shiny Apps

1. Run locally – local host server
   - Run App button
   - `shiny::runApp('app', port = 3838)`
2. Web-based hosting
   - Shinyapps.io
   - Rstudio Connect
   - Shiny server
   - Heroku

7. Resources

1. Rstudio tutorial – link
2. Mastering Shiny by Hadley Wickham – link
   - Free online book



8. Calculator Webtool Using Shiny in R

Simple Calculator

Created by Subha Srinivasagan, NMSP, Cornell University

Enter first number:

Enter second number:

Addition (+) | Subtraction (-) | Multiplication (×) | Division (÷)

Thank you!

 sn558@cornell.edu

 https://github.com/Subhashree9