Reactivity 101

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

datacamp

Greeting

Enter Name

Hello

app.R

show with app

```
library(shiny)

ui <- fluidPage(
  titlePanel('Greeting'),
  textInput('name', 'Enter Name'),
  textOutput('greeting')
)

server <- function(input, output, session) {
  output$greeting <- renderText({
    paste('Hello', input$name)
  })
}
```

Reactive source

> User input that comes through a browser interface, typically

```
ui <- fluidPage(
  titlePanel('Greeting'),
  textInput('name', 'Enter Name')
)
server <- function(input, output, session){

}

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

Reactive endpoint

> **Output** that typically appears in the browser window, such as a plot or a table of values

```
ui <- fluidPage(
  titlePanel('Greeting'),
  textInput('name', 'Enter Name'),
  textOutput('greeting')
)
server <- function(input, output, session){
  output$greeting <- renderText({
    paste("Hello", input$name)
  })
}
```

Reactive conductor

> An intermediate that depends on reactive sources, and/or updates reactive endpoints.

```
server <- function(input, output, session){
  output$plot_trendy_names <- plotly::renderPlotly({
    babynames %>%
      filter(name == input$name) %>%
      ggplot(val_bnames, aes(x = year, y = n)) +
      geom_col()
  })
  output$table_trendy_names <- DT::renderDT({
     babynames %>%
      filter(name == input$name)
  })
}
```

Reactive expressions

> Reactive expressions are **lazy** and **cached**.

```
server <- function(input, output, session){
  rval_babynames <- reactive({
    babynames %>%
      filter(name == input$name)
  })
  output$plot_trendy_names <- plotly::renderPlotly({
     rval_babynames() %>%
      ggplot(val_bnames, aes(x = year, y = n)) +
      geom_col()
  })
  output$table_trendy_names <- DT::renderDT({
     rval_babynames()
  })
}
```

Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R

datacamp

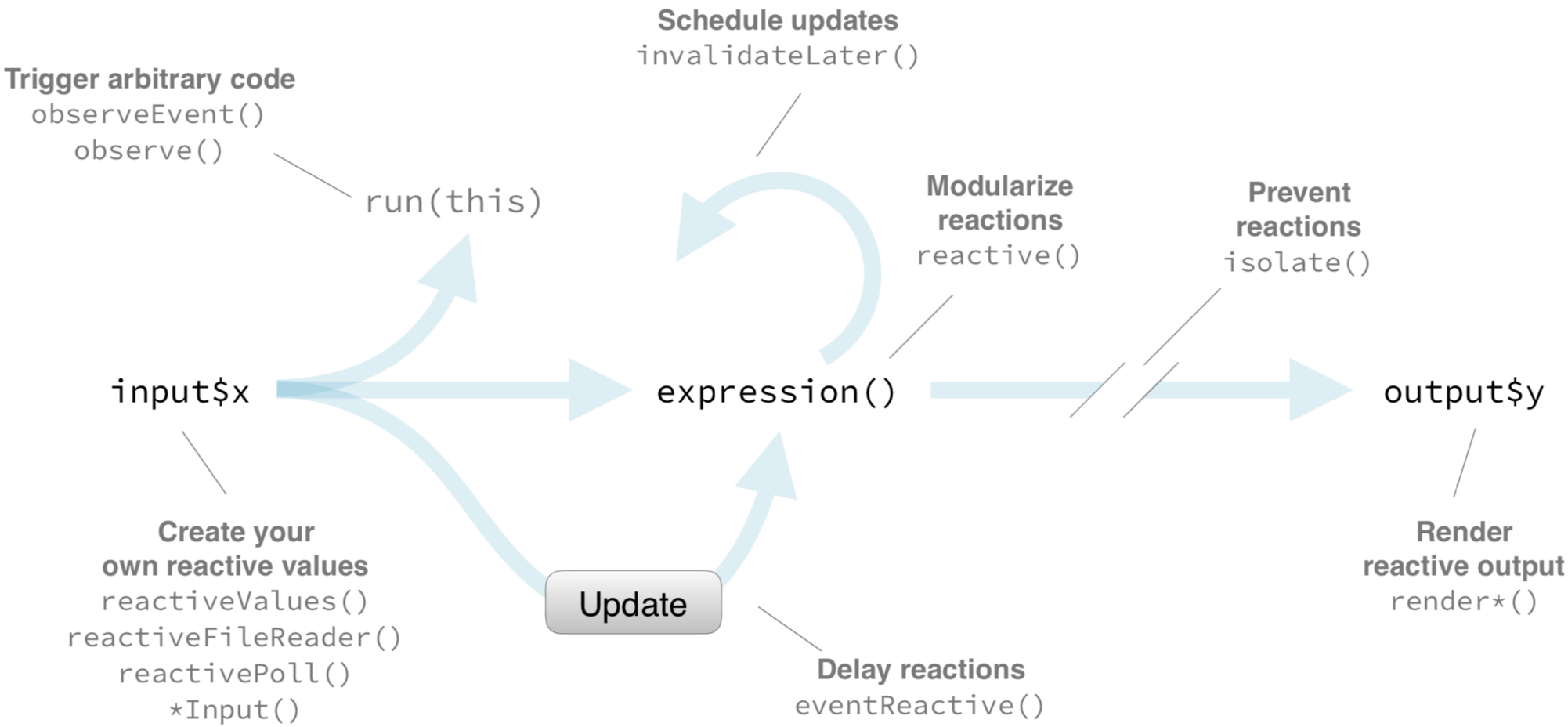
Observers vs. reactives

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

datacamp

Reactive flow



Observers (1/2)

Observers (2/2)

```
ui <- fluidPage(
  textInput('name', 'Enter your name')
)

server <- function(input, output, session){
  observe({
    showNotification(
      paste("You entered the name", input$name)
    )
  })
}
```

Observers vs. reactives

Role

- `reactive()` is for calculating values, without side effects.
- `observe()` is for performing actions, with side effects.

Differences

- **Return Values**: Reactive expressions return values, but observers don't.
- **Evaluation**: Observers eagerly respond to changes in their dependencies, while reactive expressions are lazy.
- **Side Effects**: Observers are primarily useful for their side effects, wherease, reactive expressions must NOT have side effects

Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R

datacamp

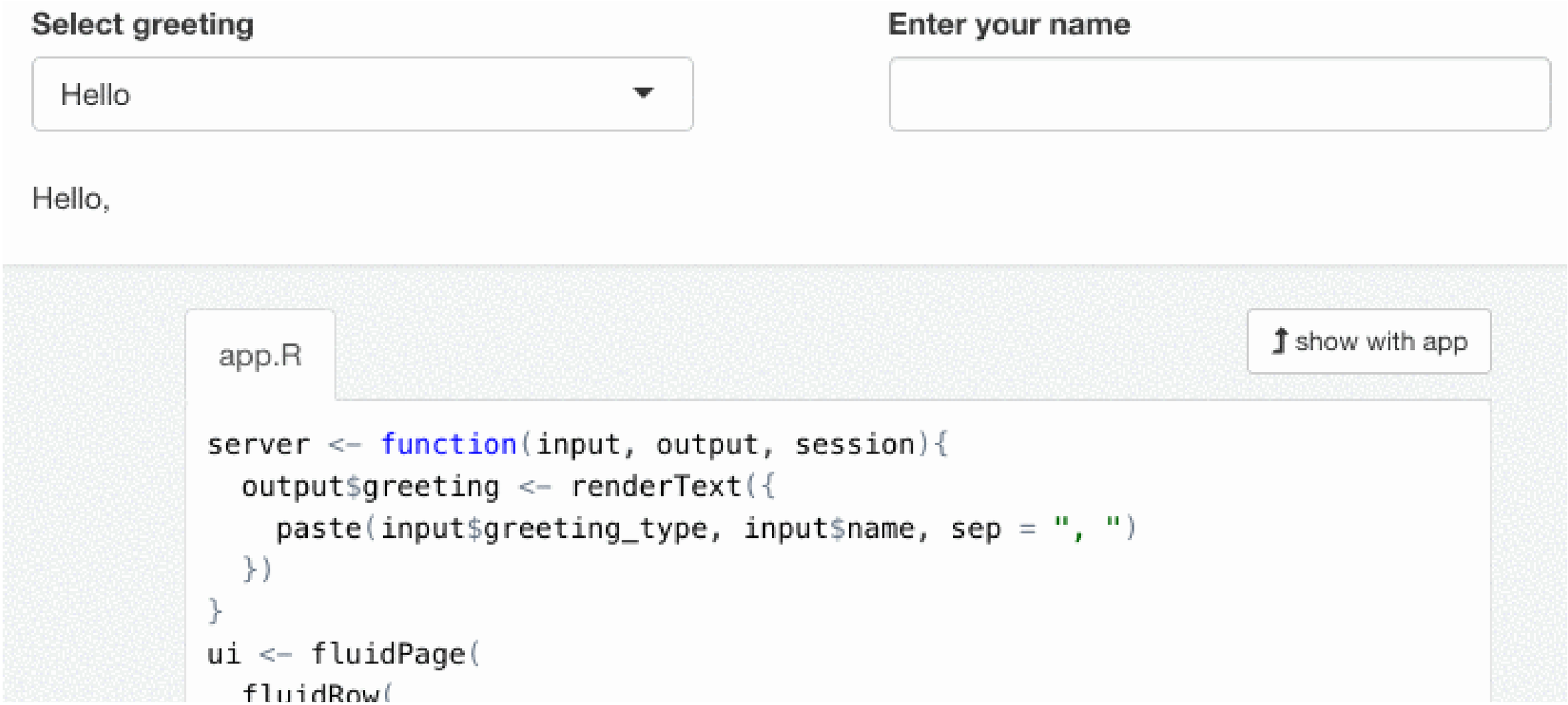
Stop - delay - trigger

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

datacamp

Isolating actions (1/3)



Isolating actions (2/3)

app.R

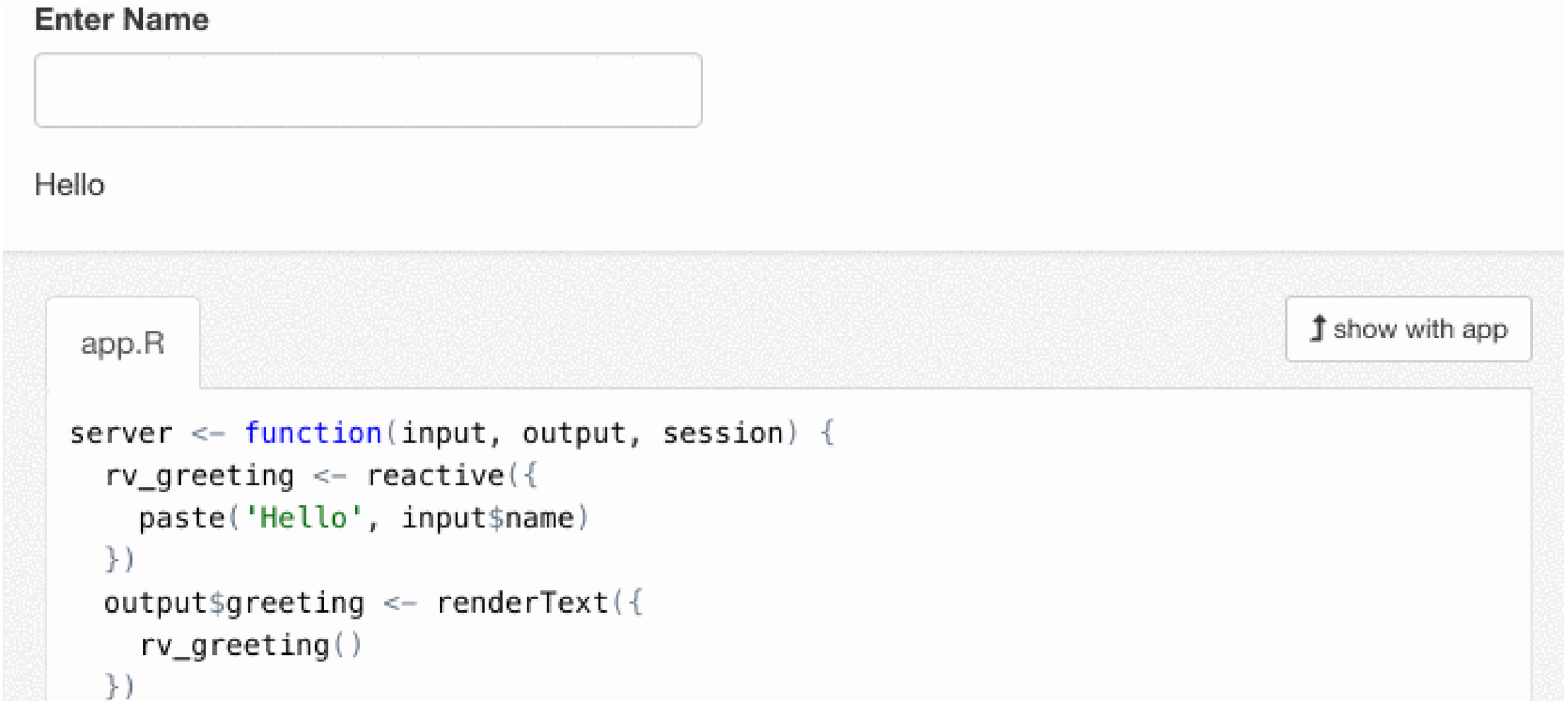
show with app

```
server <- function(input, output, session){
  output$greeting <- renderText({
    paste(isolate({input$greeting_type}), input$name, sep = ", ")
  })
}
ui <- fluidPage(
  fluidRow(
```

Isolating actions (3/3)

```
server <- function(input, output, session){
  output$greeting <- renderText({
    paste(
      isolate(
        input$greeting_type
      )
      , input$name, sep = " ,")
  })
}
```

Delaying actions (1/3)



Enter Name
Show Greeting
app.R
show with app
server <- function(input, output, session) {
rv_greeting <- eventReactive(input$show_greeting, {
paste('Hello', input$name)
})
output$greeting <- renderText({
rv_greeting()
})
}

Delaying actions (3/3)

```
server <- function(input, output, session){
  rv_greeting <- eventReactive(input$show_greeting, {
    paste("Hello", input$name)
  })
  output$greeting <- renderText({
    rv_greeting()
  })
}
```

Triggering actions (1/2)

Triggering actions (2/2)

```
server <- function(input, output, session){
  observeEvent(input$show_greeting, {
    showModal(modalDialog(paste("Hello", input$name)))
  })
}
```

Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R

datacamp

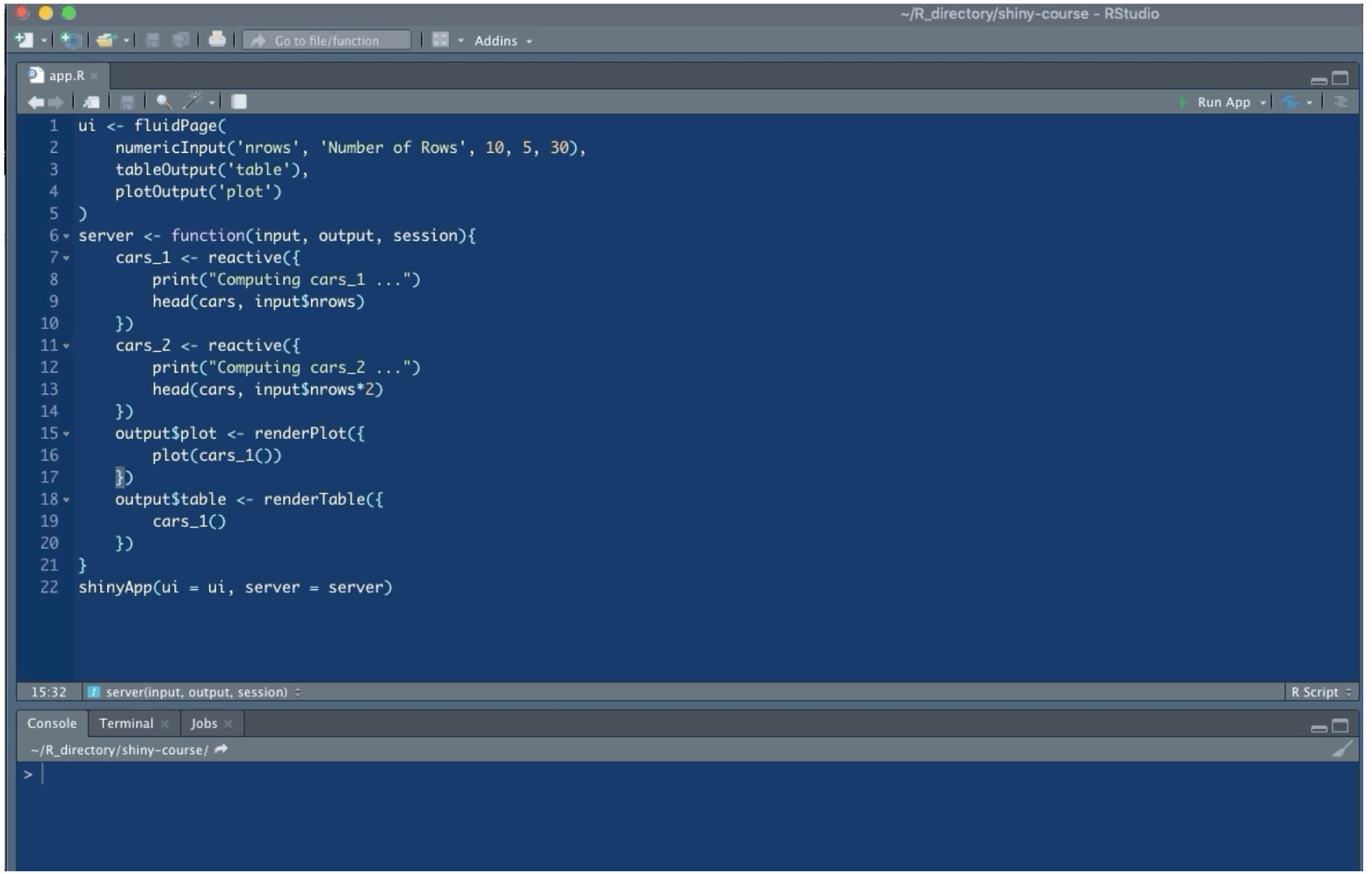
Applying reactivity concepts

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

datacamp

Reactivity 101



Reactives and observers

- Reactive sources are accessible through any `input$x` .
- Reactive conductors are good for slow or expensive calculations, and are placed between sources and endpoints.
- Reactive endpoints are accessible through any `output$y` , and are observers, primarily used for their side effects, and not directly to calculate things.

Stop, Delay, Trigger

- Stop with `isolate()`
- Delay with `eventReactive()`
- Trigger with `observeEvent()`

Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R

datacamp