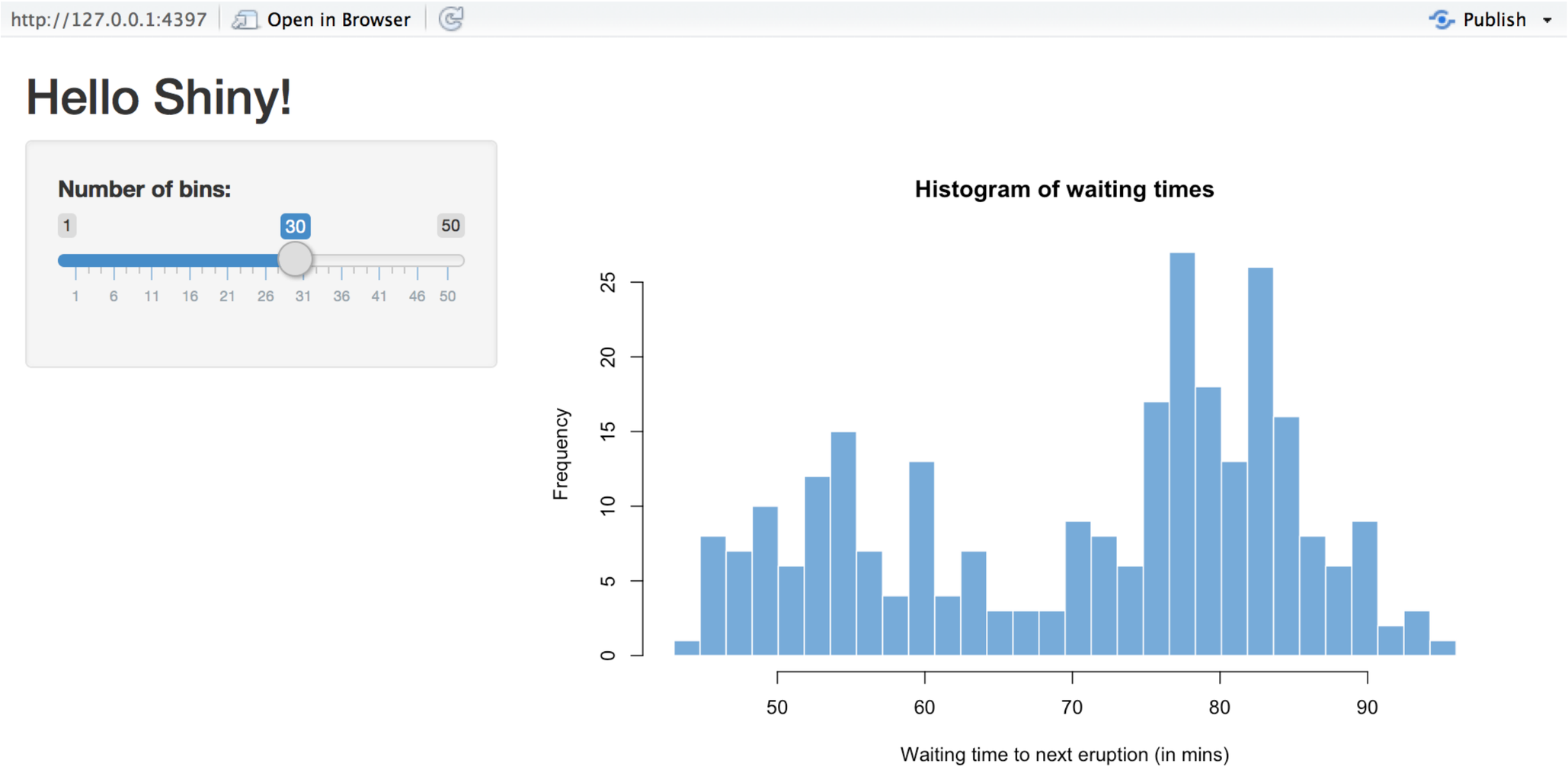
Introduction to Shiny

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

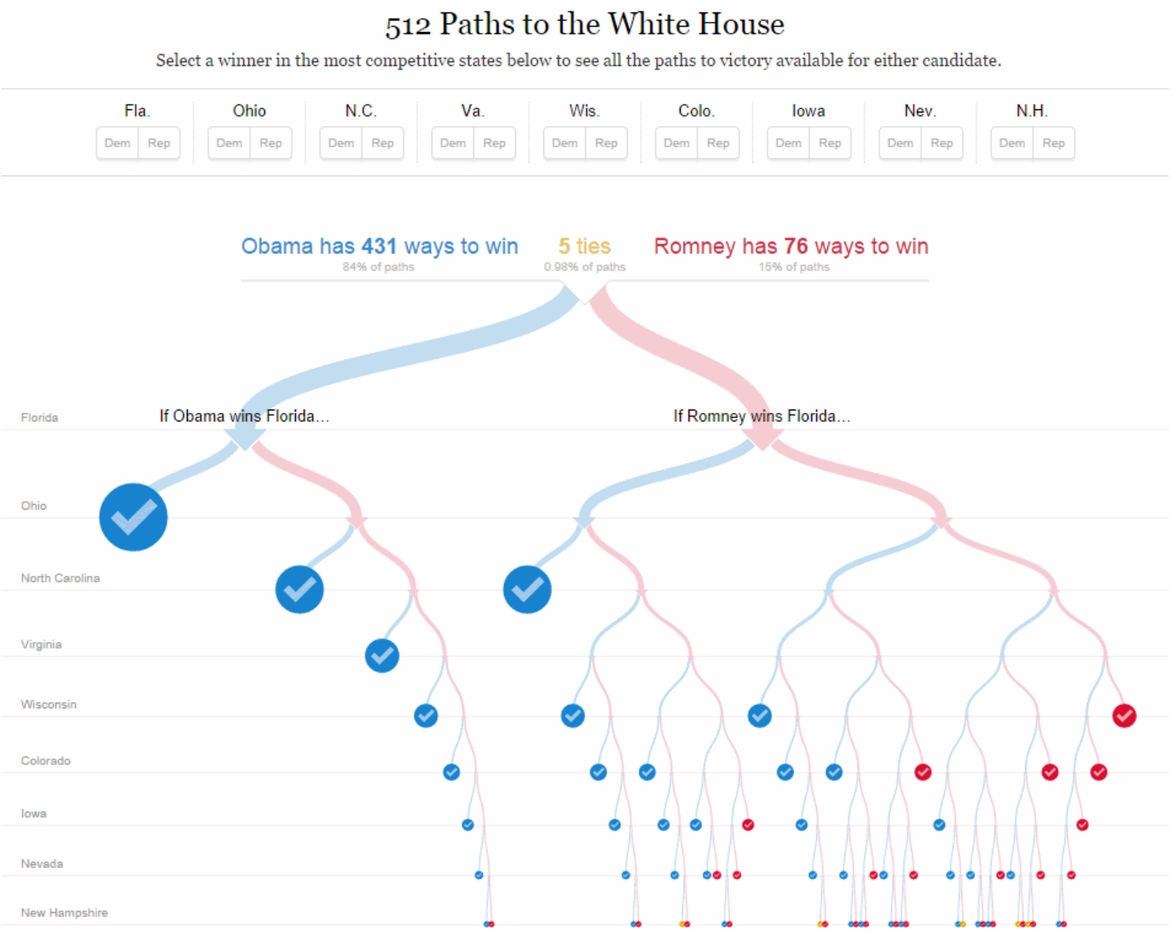
Introduction to Shiny



What is a web app?

- Updates based on user input/interaction
- Made up of UI & server

What is a web app?



- Displays paths to the White House for different presidential candidates.

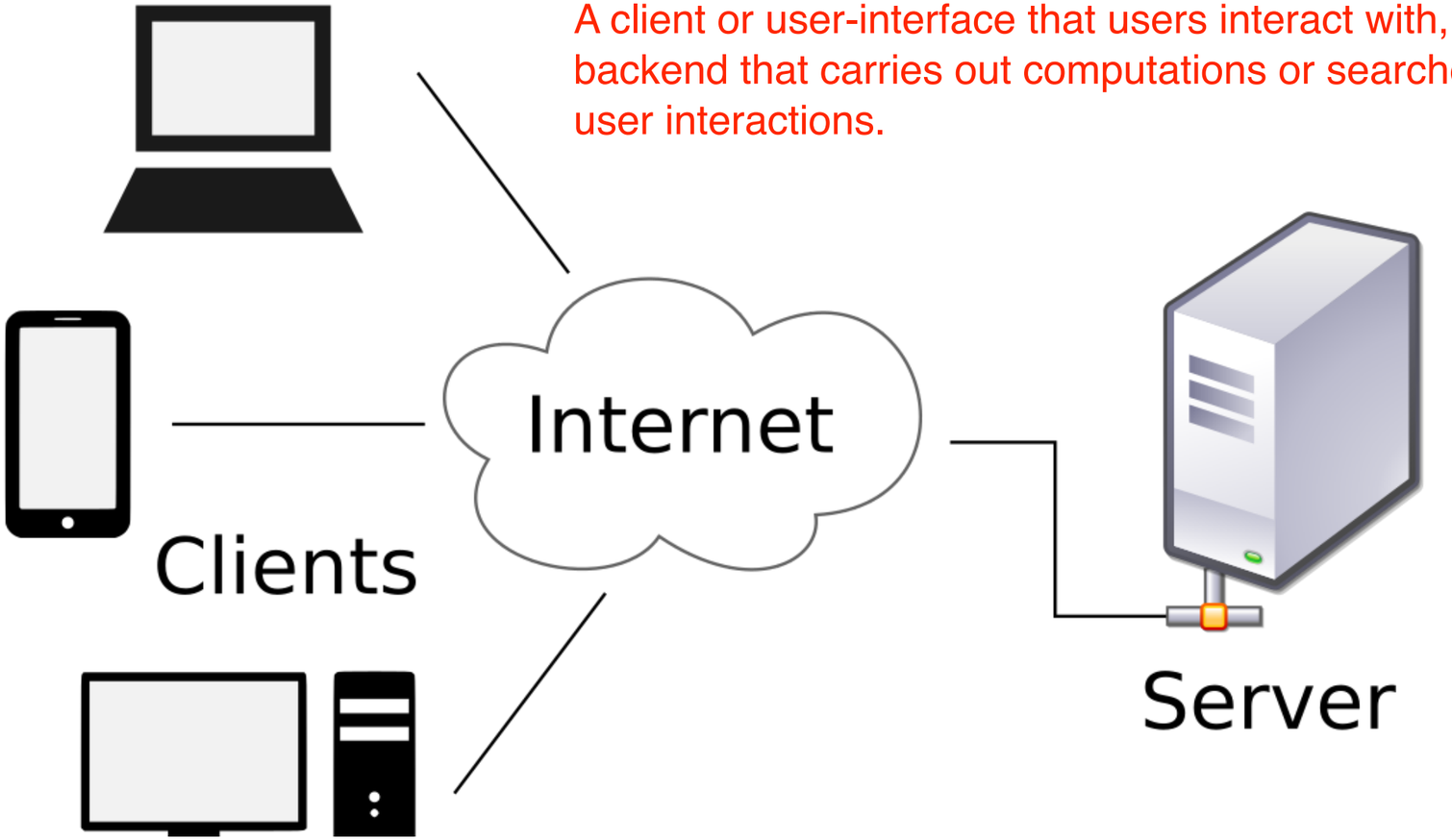
What is a web app?

- DataCamp mobile app

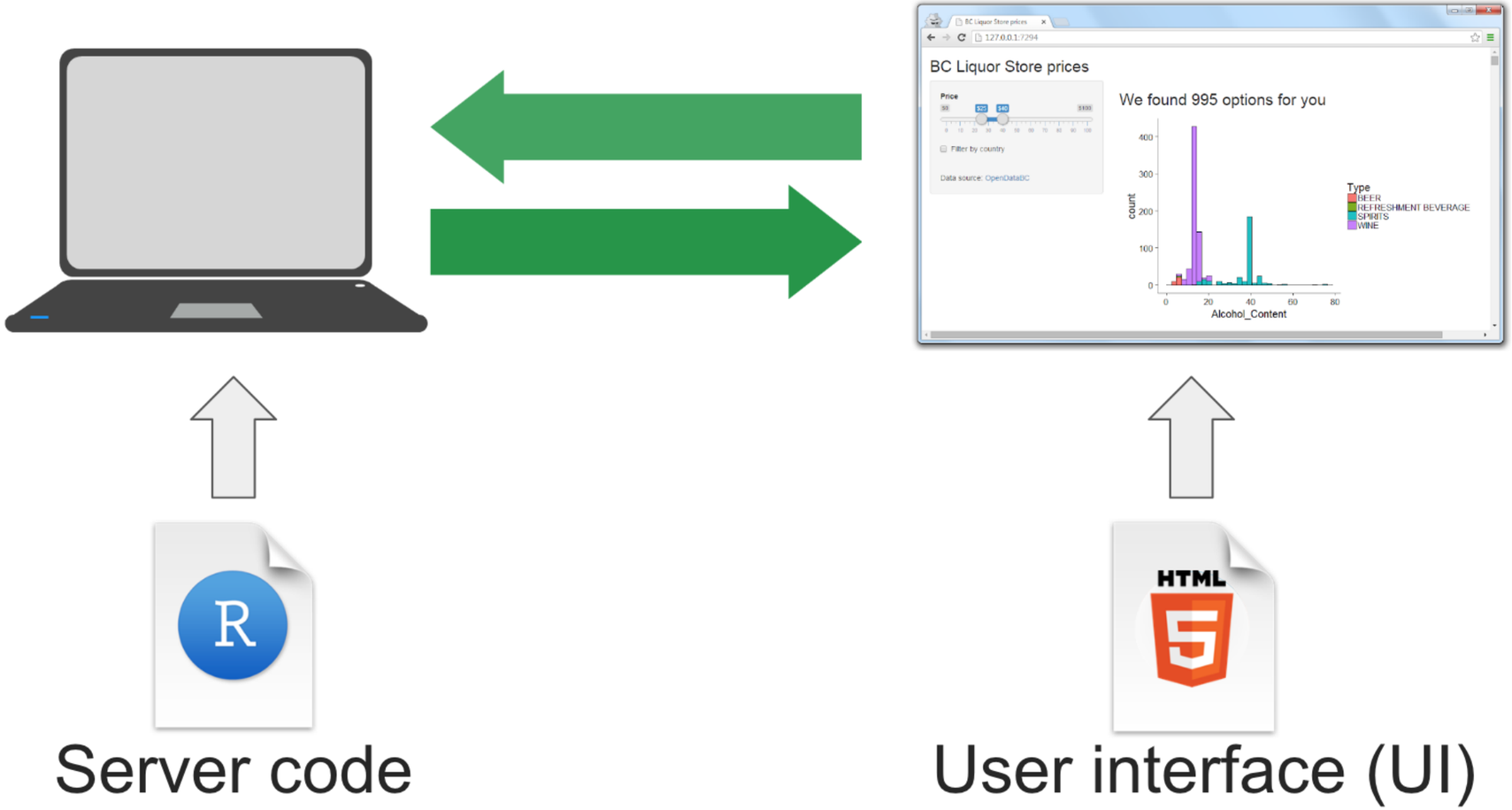
How does a web app work?

A web app is a thing that updates based on user input/interaction

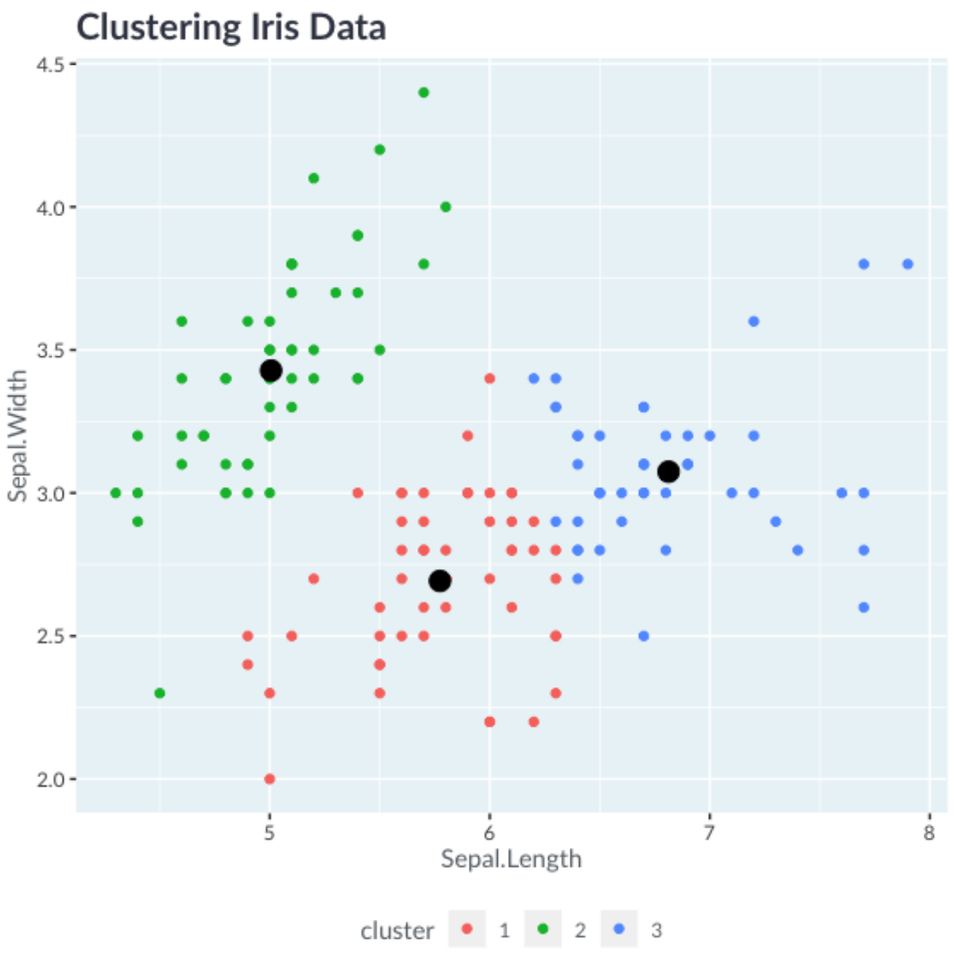
Most web apps have two parts:
A client or user-interface that users interact with, and a server or backend that carries out computations or searches based on the user interactions.



What is Shiny?



Why should data scientists build web apps?



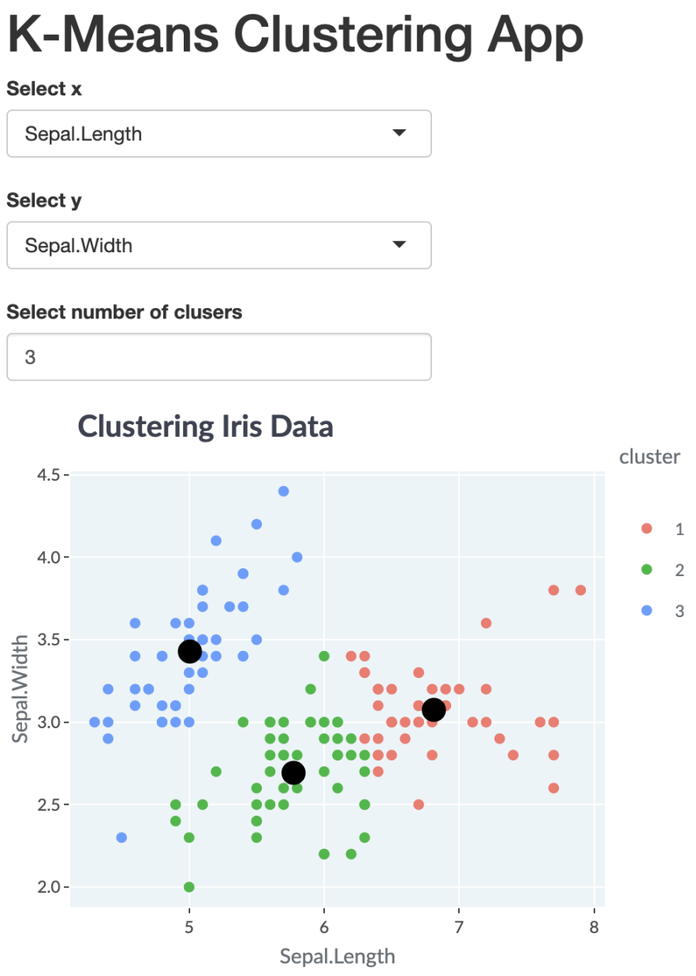
Why should data scientists build web apps?

```
plot_kmeans(
  data = iris,
  x = 'Sepal.Length',
  y = 'Sepal.Width',
  nb_clusters = 3
)
```

Why should data scientists build web apps?

```r
library(shiny)
ui <- fluidPage(
  h1('K-Means Clustering App'),
  selectInput('x', 'Select x', names(iris), 'Sepal.Length'),
  selectInput('y', 'Select y', names(iris), 'Sepal.Width'),
  numericInput('nb_clusters', 'Select number of clusers', 3),
  plotly::plotlyOutput('kmeans_plot')
)

server <- function(input, output, session){
  output$kmeans_plot <- plotly::renderPlotly({
    plot_kmeans(iris, input$x, input$y, input$nb_clusters)
  })
}

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

Why should data scientists build web apps?



Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R

Build a "Hello, world" Shiny app

BUILDING WEB APPLICATIONS WITH SHINY IN R

Kaelen Medeiros
Data Scientist

Parts of a Shiny app

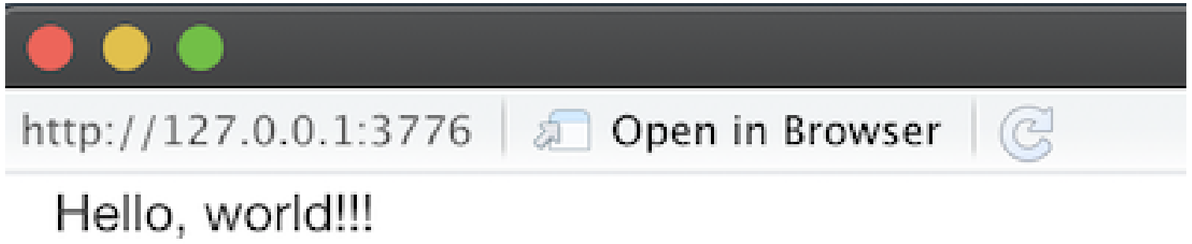
```
library(shiny)

ui <- fluidPage()

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


}

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

- Load `shiny`
- Create the UI with a HTML function
- Define a custom function to create the server
- Run the app

Creating the UI doesn't require a specialized Shiny function. Most people use fluidPage, which allows for implementation of a blank HTML fluid page layout, organized by rows and columns, for your app.
The server is created in R by defining a custom function.
session argument: to specify the specific session

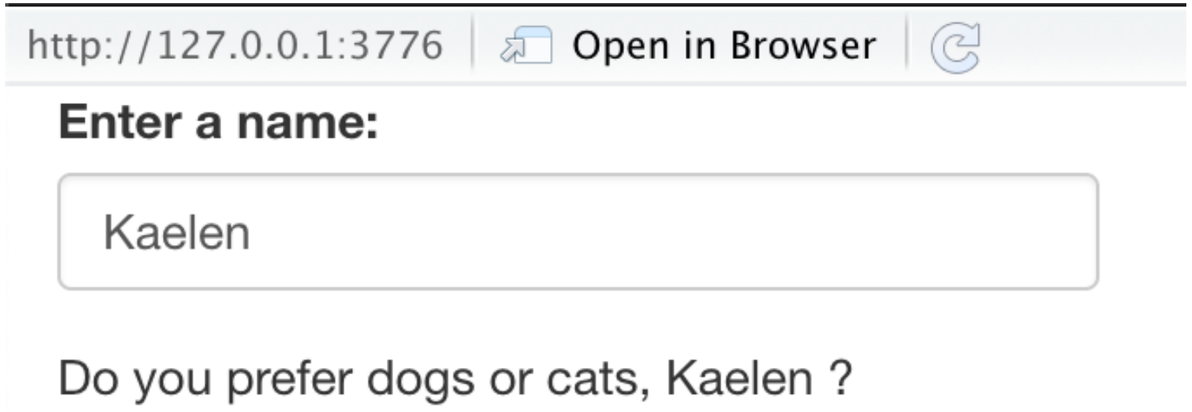
Hello, world!!!

```r
library(shiny)

ui <- fluidPage(
    "Hello, world!!!"
)

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

}

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



Ask a question (with an input!)

```
ui <- fluidPage(
    textInput("name", "Enter a name:"),
    textOutput("q")
)
server <- function(input, output) {
    output$q <- renderText({
        paste("Do you prefer dogs
                    or cats,",
              input$name, "?")
      })
}
```



You could take in a name from the user, and use that name to wish that person “hello”, or to ask them a question.
textInput allows users to enter text, and takes three arguments: a unique id that will be used to refer to this input, a label that is displayed to the user, and an optional default value.
The full output is built in the server using the renderText function and is assigned to an output object, output$q.
Back up in the UI, use the textOutput function to display the output q.
If you get an error message resembling Parsing error in script.R:4:3: unexpected symbol, it is very likely that you have forgotten to use a comma to separate the arguments to one of the functions.

Let's practice!

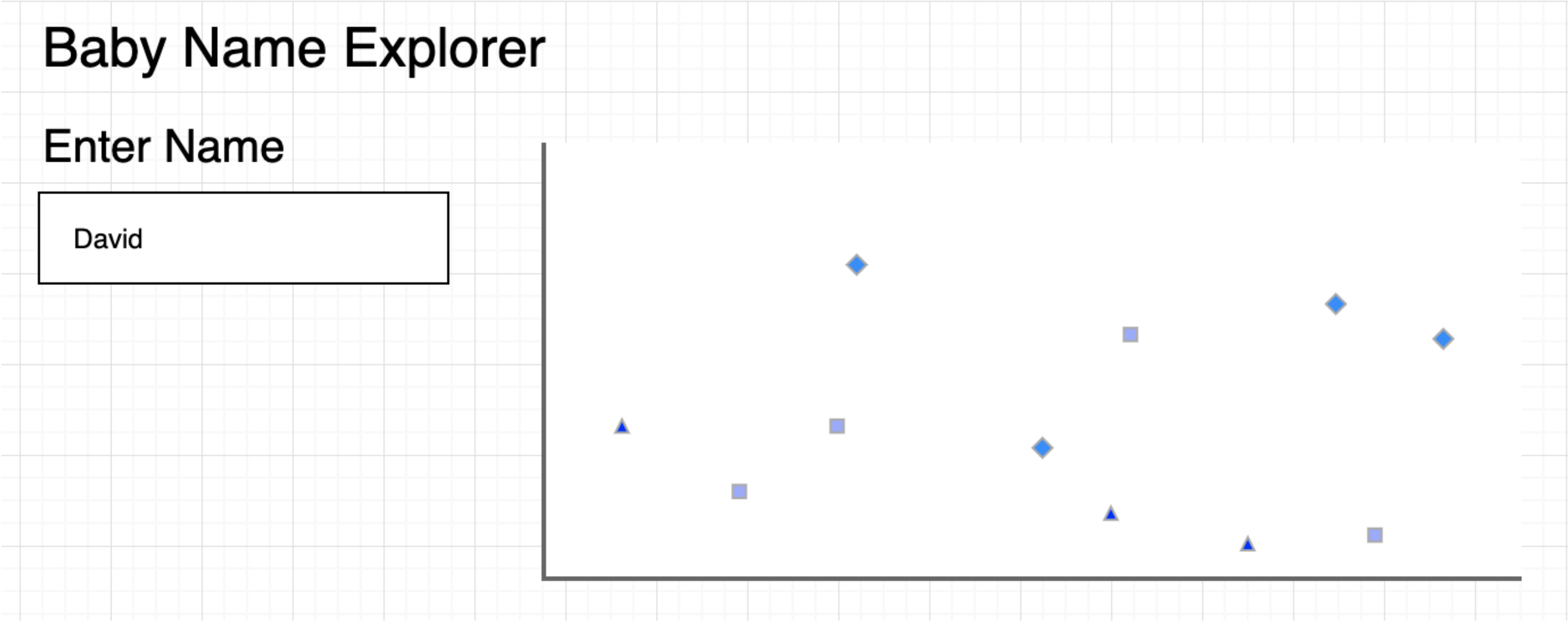
BUILDING WEB APPLICATIONS WITH SHINY IN R

Build a babynames explorer Shiny app

BUILDING WEB APPLICATIONS WITH SHINY IN R

Ramnath Vaidyanathan
VP of Product Research

Sketch your app

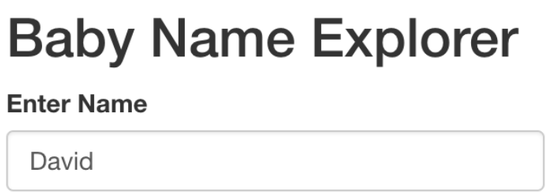


Add inputs (UI)

```
ui <- fluidPage(
  titlePanel("Baby Name Explorer"),
  textInput('name', 'Enter Name', 'David')
)
```

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

}
```

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



The first step is to add a text input to the UI that will allow a user to enter their (or any other) name. David is the default value.

Add outputs (UI/server)

```
ui <- fluidPage(
  titlePanel("Baby Name Explorer"),
  textInput('name', 'Enter Name', 'David'),
  plotOutput('trend')
)
```

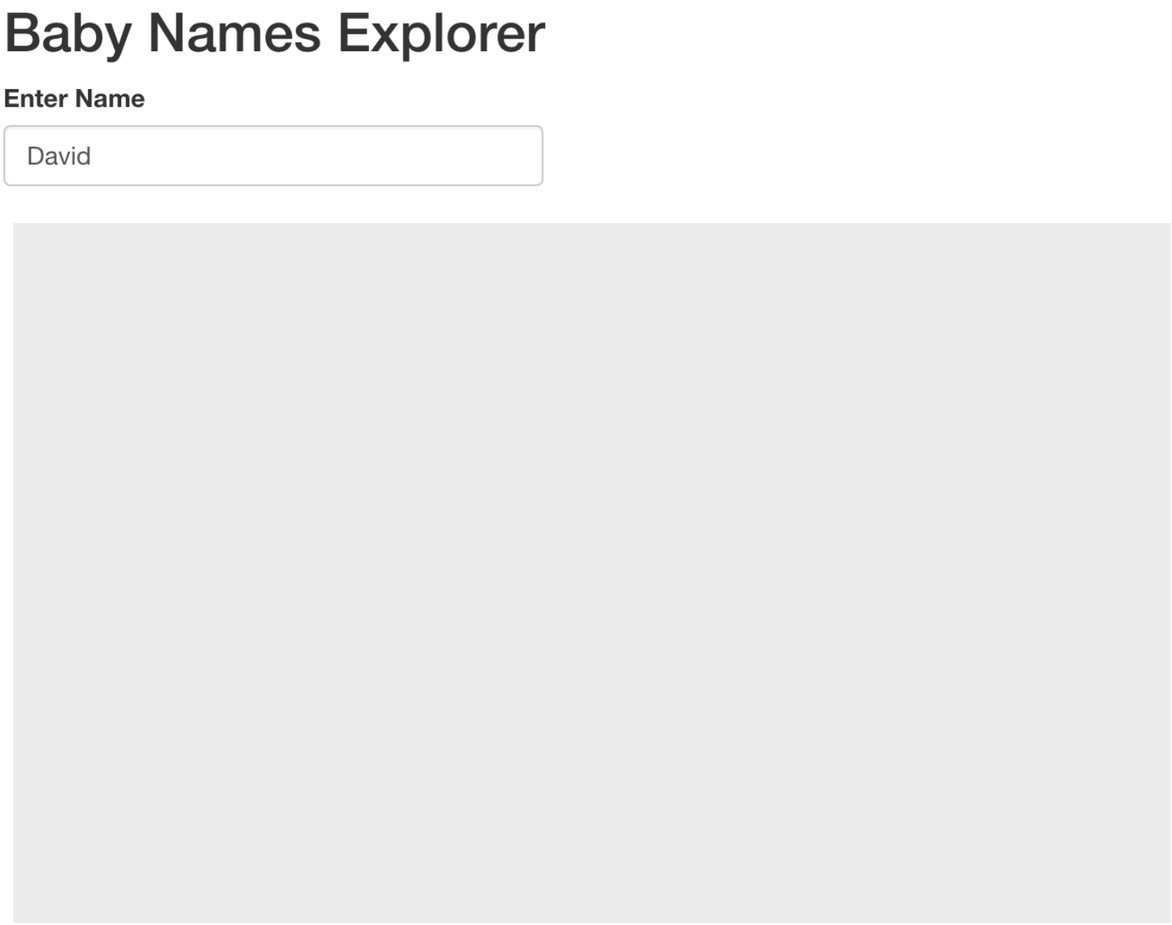
```
server <- function(input, output, session){
  output$trend <- renderPlot({
    ggplot()
  })
}
```

The next step in building your app is to add an empty plot as a placeholder. In order to add a plot p assigned to an object named x to a Shiny app, you need to:

1. Render a plot object using renderPlot({ggplot()}).
2. Assign the rendered plot to output$trend.
3. Display the plot in the UI using plotOutput("trend").

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

Add outputs (UI/server)

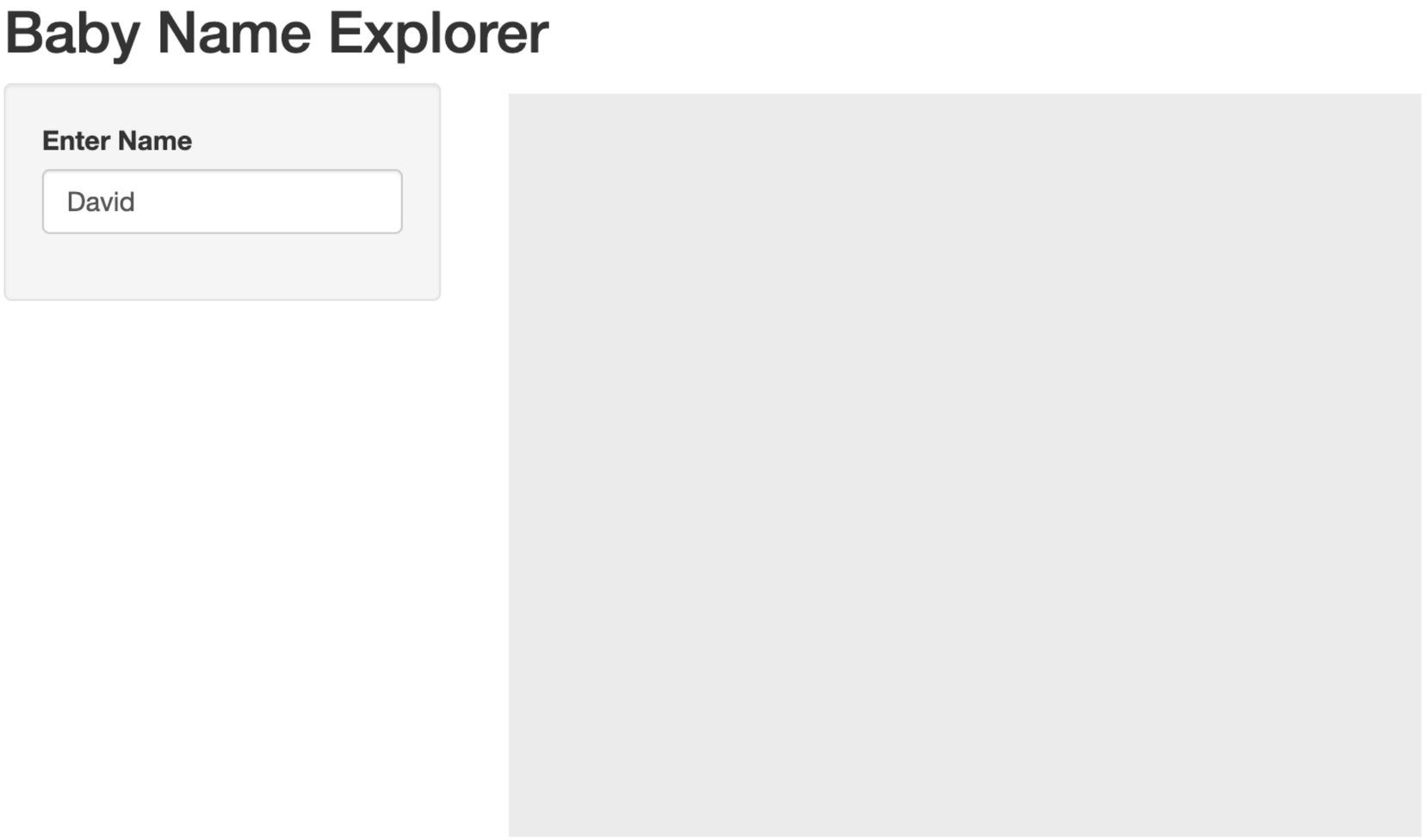


Update layout (UI)

```
ui <- fluidPage(
  titlePanel("Baby Name Explorer"),
  sidebarLayout(
    sidebarPanel(
      textInput('name', 'Enter Name', 'David')
    ),
    mainPanel(
      plotOutput('trend')
    )
  )
)
```

```
server <- function(input, output, session){
  output$trend <- renderPlot({ggplot()})
}
```

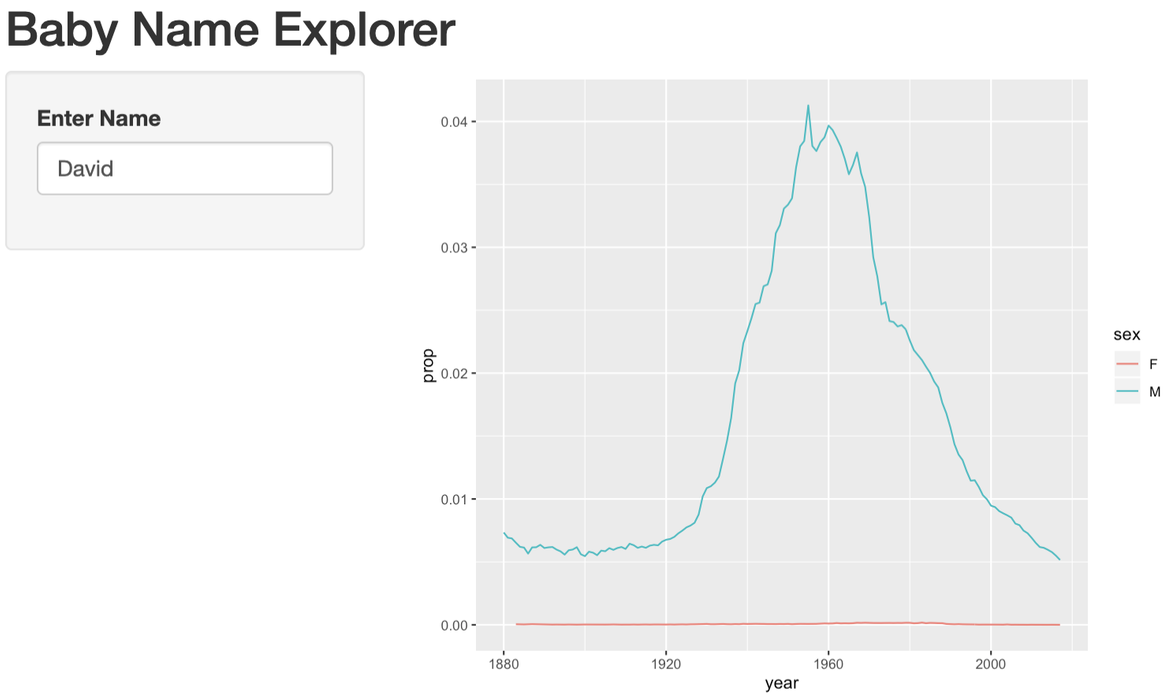
Update layout (UI)



Update output (server)

```
ui <- fluidPage(
  ...
)
```

```
server <- function(input, output, session){
  output$trend <- renderPlot({
    data_name <- subset(
      babynames, name == input$name
    )
    ggplot(data_name) +
      geom_line(
        aes(x = year, y = prop, color = sex)
      )
  })
}
```



Let's practice!

BUILDING WEB APPLICATIONS WITH SHINY IN R