Introduction to Shiny

Biostatistics 140.776

Shiny

- Shiny is a platform for creating interactive R programs embedded into a web page.
- Suppose that you create a prediction algorithm, with shiny you can very easily create web input form that calls R and thus your prediction algorithm and displays the results.
- Using Shiny, the time to create simple, yet powerful, web-based interactive data products in R is minimized.
- However, it lacks the flexibility of full featured (and more complex) solutions.
- Shiny is made by the fine folks at RStudio.

Shiny

- Shiny doesn't really require it, but as with all web programming, a little awareness of HTML, CSS and JavaScript is very helpful
- HTML gives a web page structure and sectioning as well as markup instructions
- CSS describes how content is presented
- JavaScript is for interactivity
- There are too many tutorials online to count for getting basic proficiency in these topics to count.
- Shiny uses bootstrap (no relation to the statistics bootstrap) style, which (to me) seems to look nice and renders well on mobile platforms

A Shiny Project

- A shiny project is a directory containing at least two parts
- **ui.R** (for user interface) controls how it looks
- **server.R** controls what it does
- Creating a “New Shiny Project” in RStudio will create these files for you and fill them with example code

A Shiny Project

- The components/functions in the ui.R file communicate with the functions in the server.R file
- Communication is done by passing R objects back and forth in the background
- Communication depends on mutually agreed upon variable/object names

ui.R

```
library(shiny)

shinyUI(fluidPage(
        titlePanel(”Biostatistics Rulz!"),
        sidebarPanel(
                h3('Sidebar text')
        ),
        mainPanel(
                h3('Main Panel text')
        )
))
```

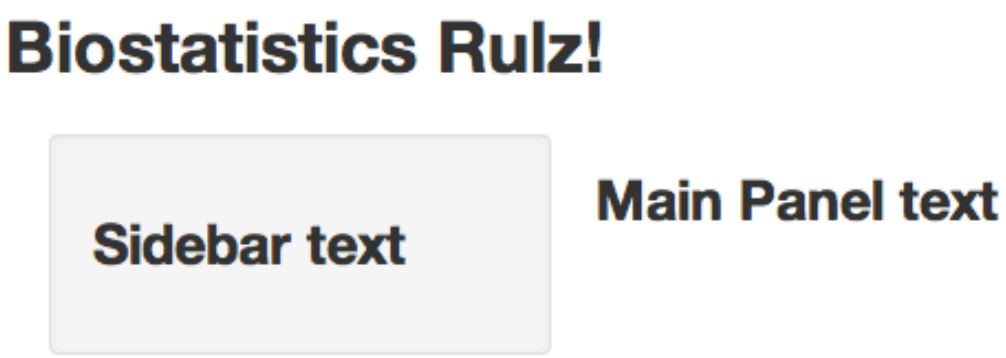
server.R

```
library(shiny)

shinyServer(function(input, output) {
   ## Nothing for now
})
```

Run It!

- In R, change to the directories with these files and type `runApp( )`
- Or put the path to the directory as an argument
- It should open an browser window with the app running
- In RStudio just click the “Run App” button

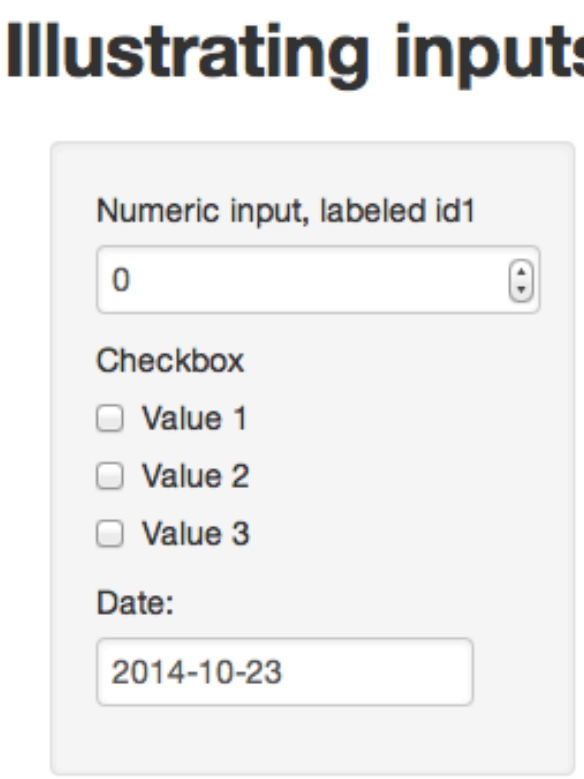
Result



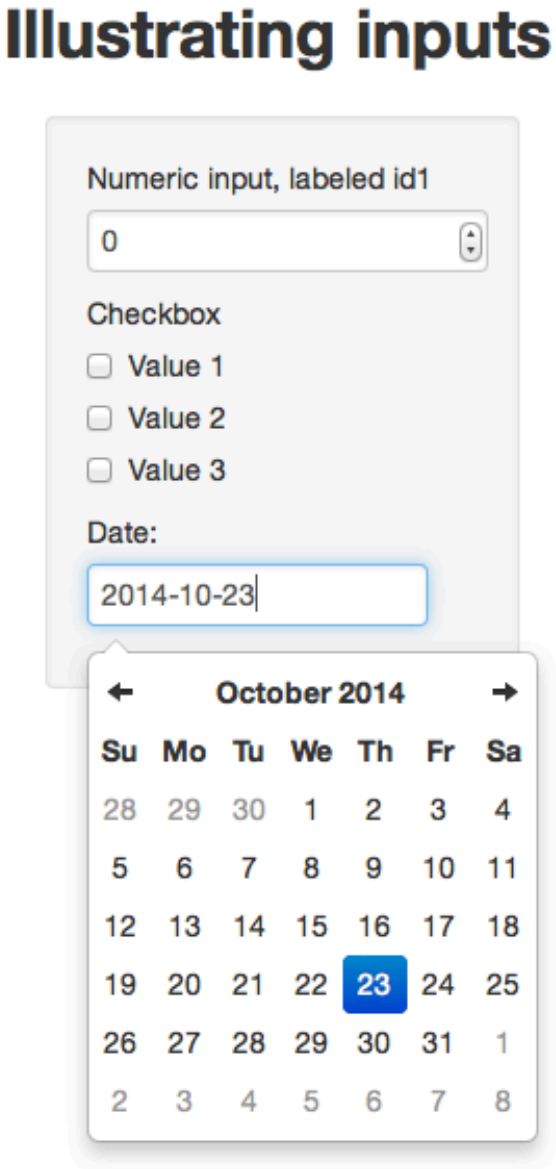
Inputs

```
shinyUI(fluidPage(
        titlePanel("Illustrating inputs"),
        sidebarPanel(
                numericInput('id1', 'Numeric input, labeled id1', 0,
                             min = 0, max = 10, step = 1),
                checkboxGroupInput("id2", "Checkbox",
                                   c("Value 1" = "1",
                                     "Value 2" = "2",
                                     "Value 3" = "3")),
                dateInput("date", "Date:")
        ),
        mainPanel(

        )
))
```

Result


Illustrating inputs
Numeric input, labeled id1
0
Checkbox
Value 1
Value 2
Value 3
Date:
2014-10-23

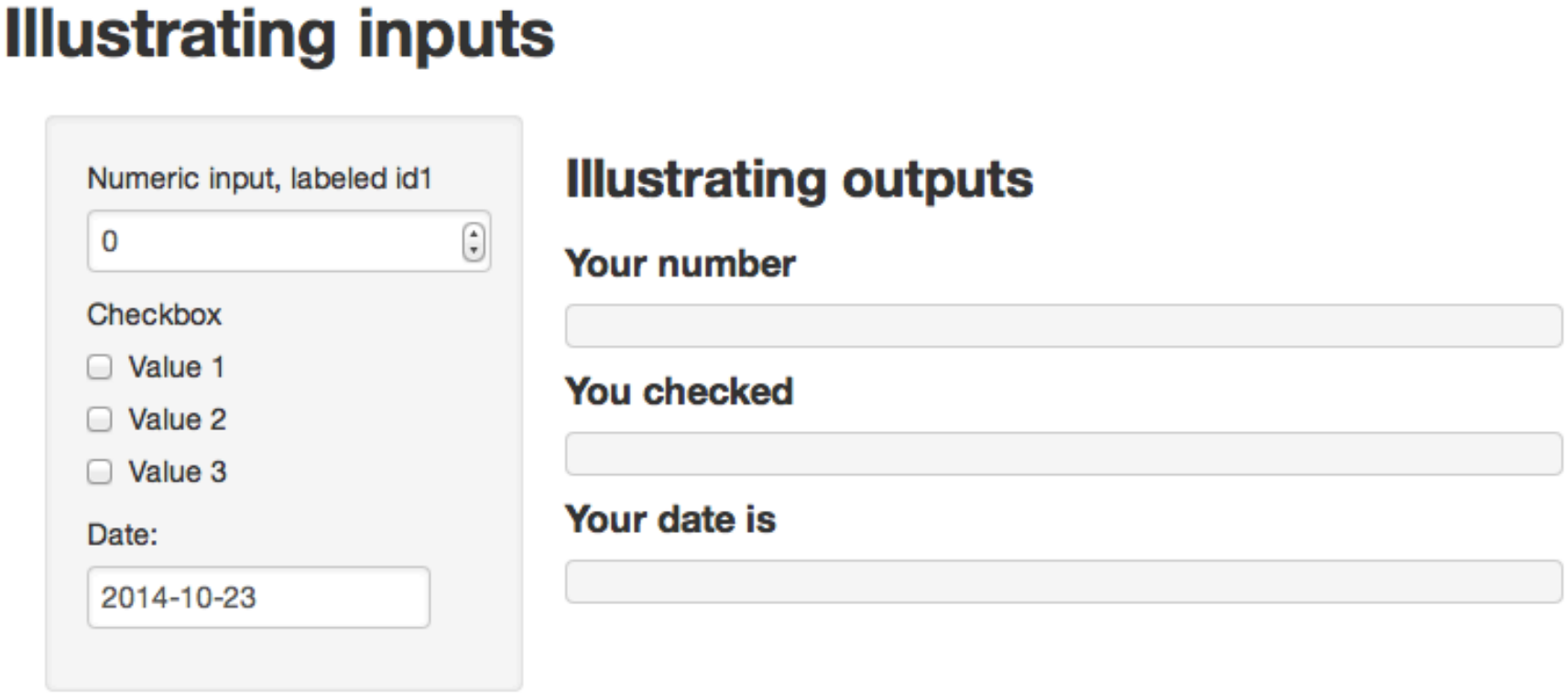
Result


Illustrating inputs
Numeric input, labeled id1
0
Checkbox
Value 1
Value 2
Value 3
Date:
2014-10-23
October 2014
Su Mo Tu We Th Fr Sa
28 29 30 1 2 3 4
5 6 7 8 9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31 1
2 3 4 5 6 7 8

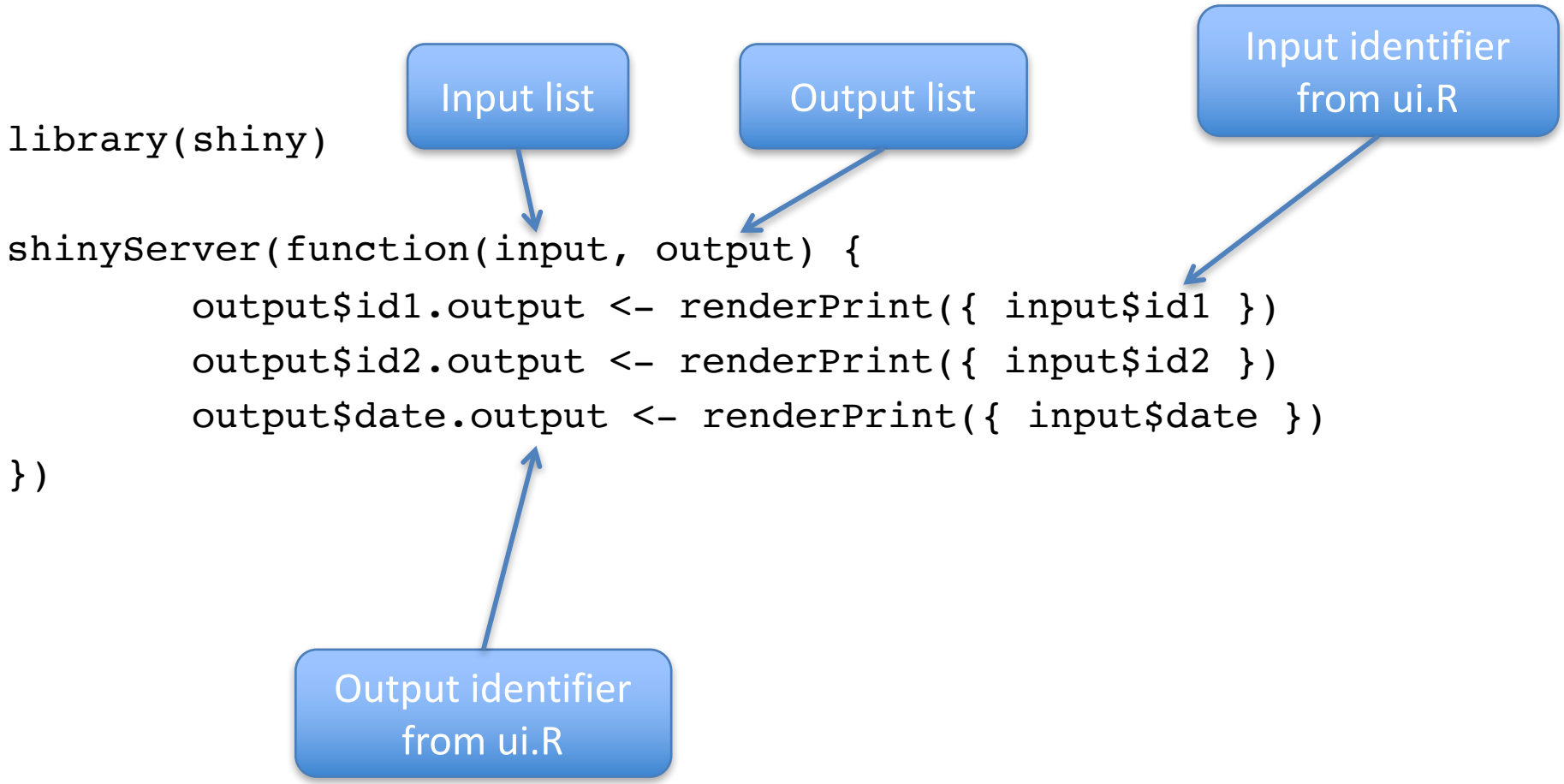
Inputs + Outputs

```
shinyUI(fluidPage(
        titlePanel("Illustrating inputs"),
        sidebarPanel(
                numericInput('id1', 'Numeric input, labeled id1', 0, min = 0,
                             max = 10, step = 1),
                checkboxGroupInput("id2", "Checkbox",
                                   c("Value 1" = "1",
                                     "Value 2" = "2",
                                     "Value 3" = "3")),
                dateInput("date", "Date:")
        ),
        mainPanel(
                h3('Illustrating outputs'),
                h4('Your number'),
                verbatimTextOutput("id1.output"),
                h4('You checked'),
                verbatimTextOutput("id2.output"),
                h4('Your date is'),
                verbatimTextOutput("date.output")
        )
))
```

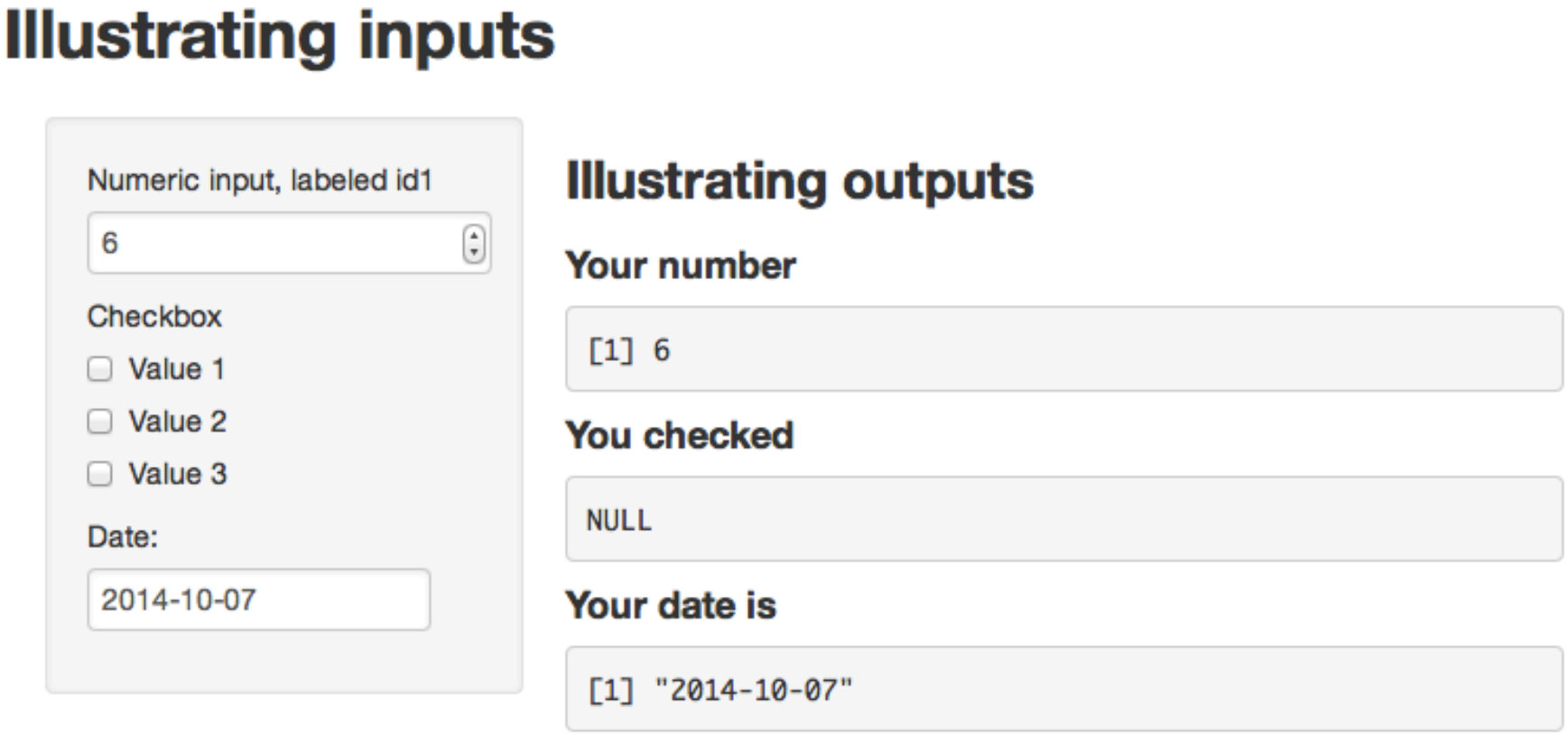
Inputs + Outputs



server.R


Input list
Output list
Input identifier from ui.R
library(shiny)
shinyServer(function(input, output) {
output$id1.output <- renderPrint({ input$id1 })
output$id2.output <- renderPrint({ input$id2 })
output$date.output <- renderPrint({ input$date })
})
Output identifier from ui.R

Inputs + Outputs



Autcomplete App

- Build a simple web app that takes text and predicts the currently typed word
- Need a text input field on the left
- Show output on the right
- Run the `autocomplete()` function behind the scenes and dynamically update as user types

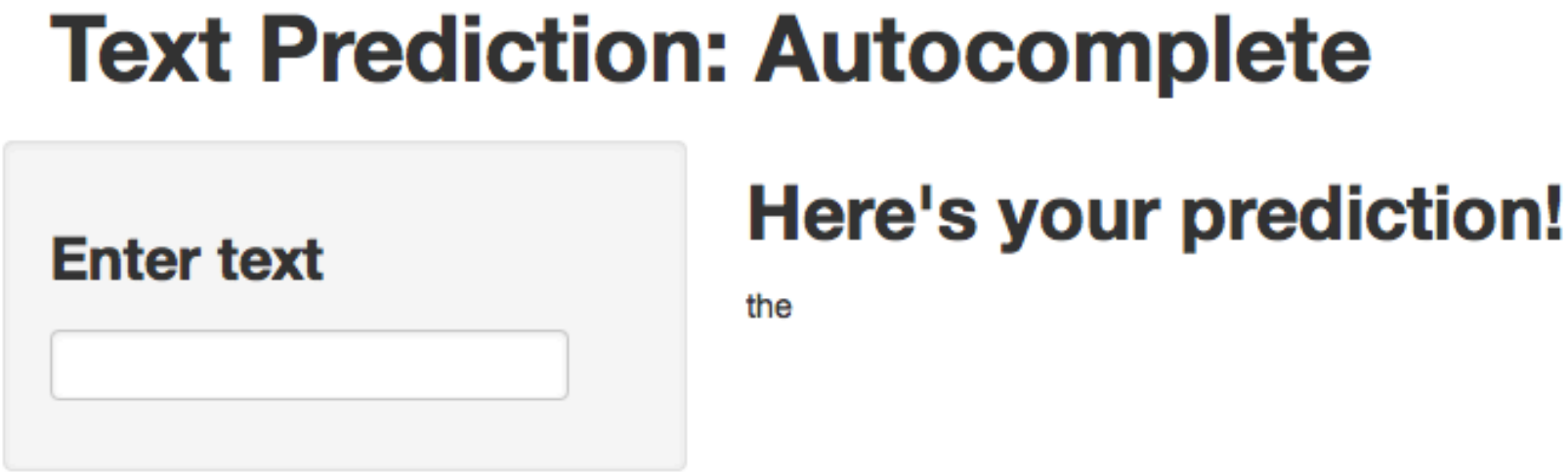
ui.R

```
library(shiny)

shinyUI(fluidPage(
  headerPanel("Text Prediction: Autocomplete"),
  sidebarLayout(
    sidebarPanel(
        textInput(inputId = "letters", label = h3("Enter text"))
    ),
    mainPanel(
        h2("Here's your prediction!"),
        textOutput("completion")
    )
  )
))
```

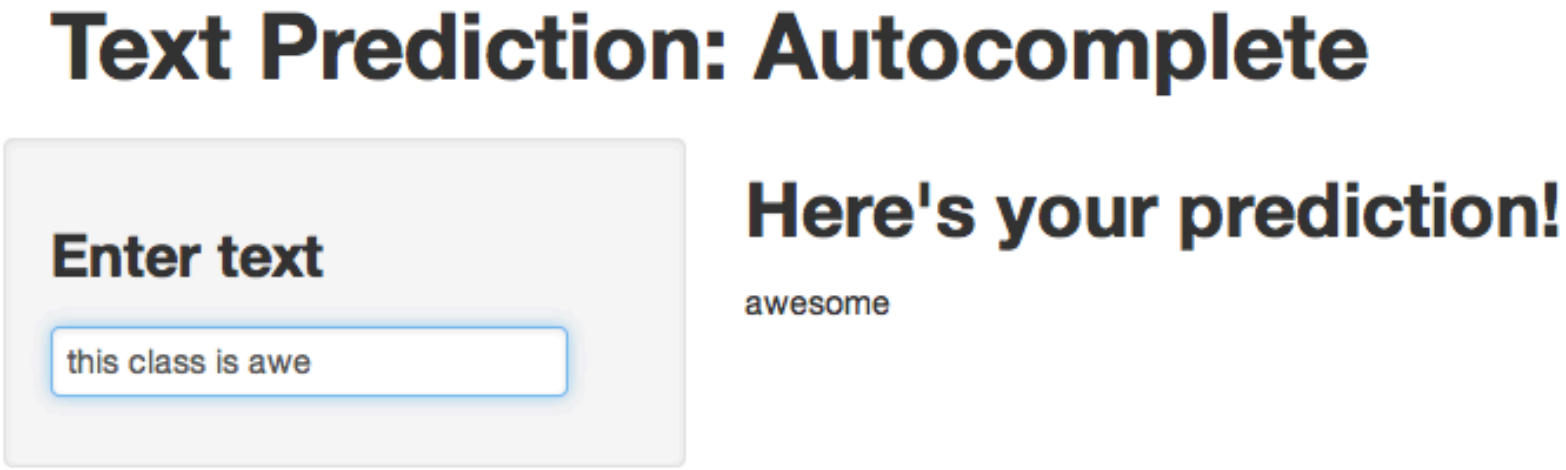
server.R

```
library(shiny)
source("autocomplete.R")

shinyServer(function(input, output) {
        output$completion <- renderText({
                autocomplete(input$letters)
        })
})
```

Autocomplete



Autocomplete



Deploying Your App

- Shiny apps can be deployed on RStudio's shinyapps.io server
- Need to install the shinyapps package from RStudio
- Setup account at shinyapps.io (you can use Google or GitHub accounts)
- Send server your credentials
- After building/testing your app locally on your computer run `deployApp()`

Summary

- Manipulate gives you a quick and dirty way to create interactive plots within RStudio
- Shiny lets you create web apps without having to focus on nuts and bolts of web programming
- It’s possible to get into the nitty gritty web stuff if you want
- Apps can be deployed on the web quickly