# Shiny

Creating interactive dashboards with R

**Jakob Graff**

## Why dashboards?

## Why Shiny?

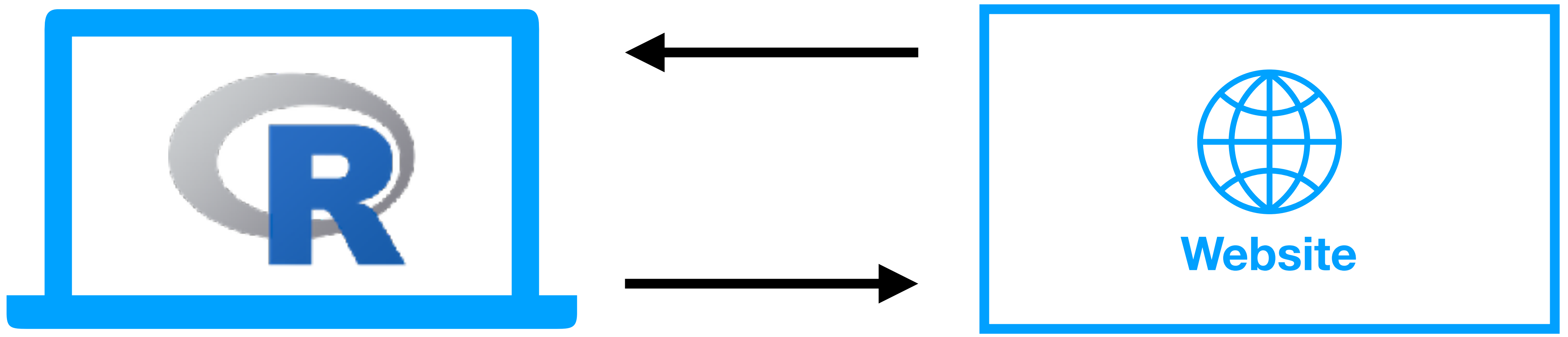
## Why Shiny?

- Free & open source
- Fast & simple
- No knowledge of web development necessary, but integration of HTML, CSS, Javascript possible
- Can easily be deployed on server

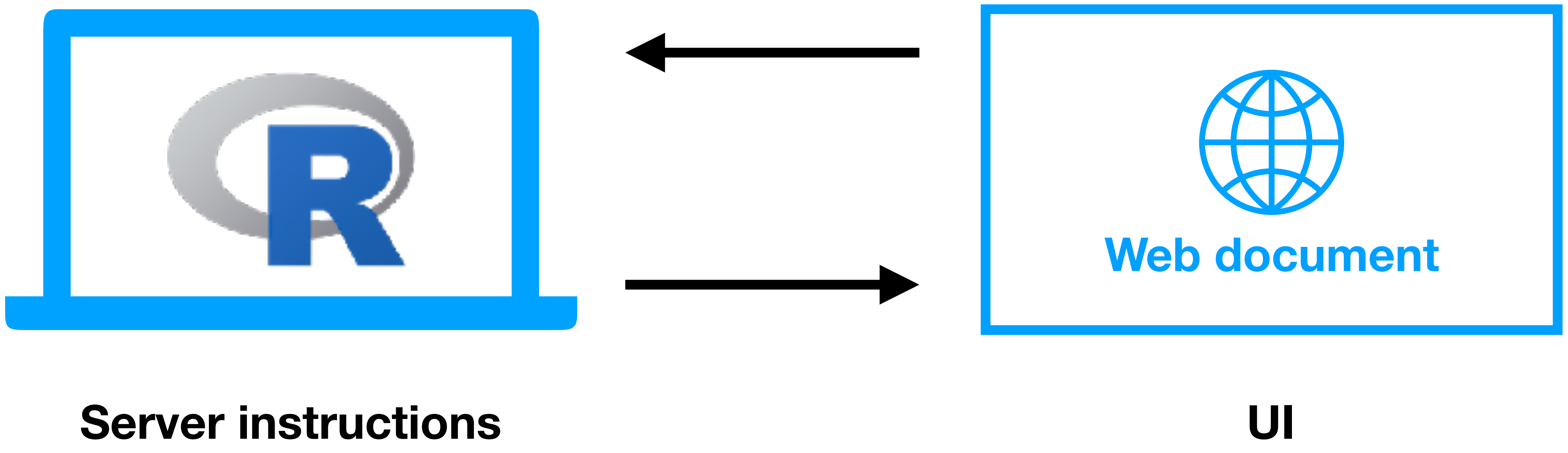
# Agenda

1. Shiny Basics
    a. How Shiny works
    b. UI ingredients
2. More advanced Shiny
    a. Reactive expressions
    b. HTML elements
    c. Styling an app
    d. Deploying apps on a server
3. Summary
4. Tutorial time

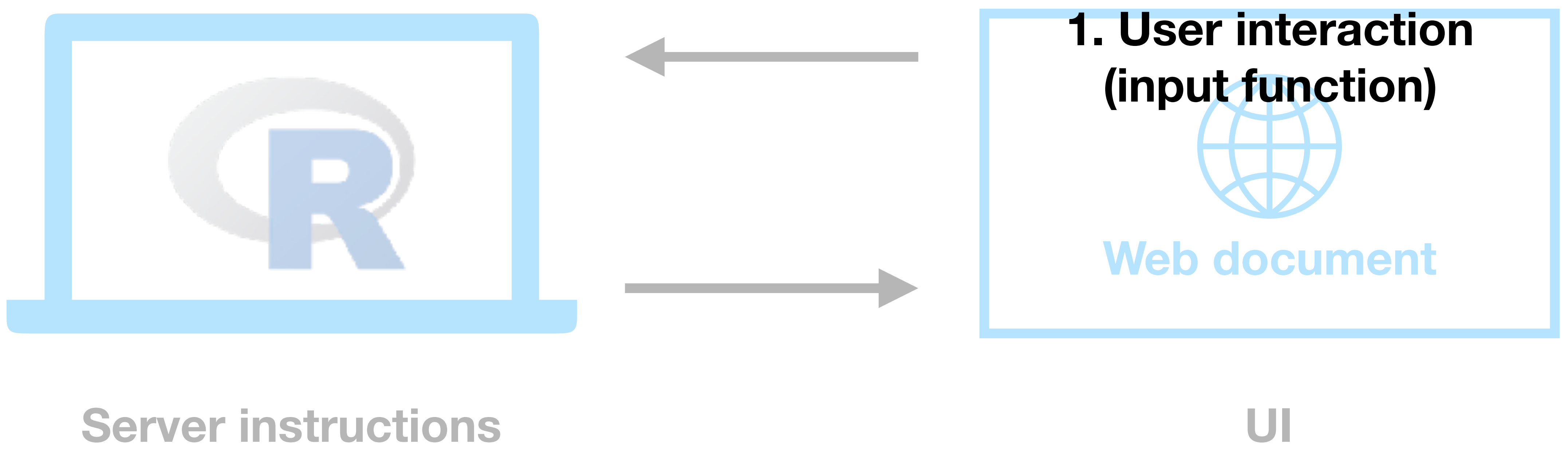
## Shiny Basics: How it Works



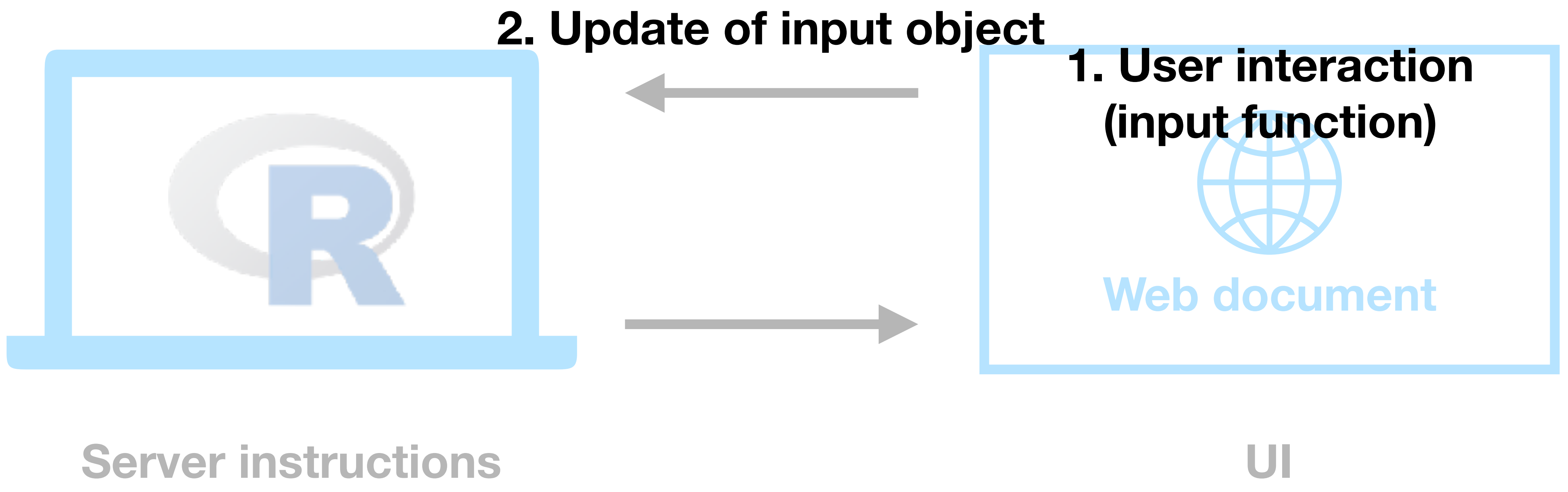
Source: http://shiny.rstudio.com/tutorial/



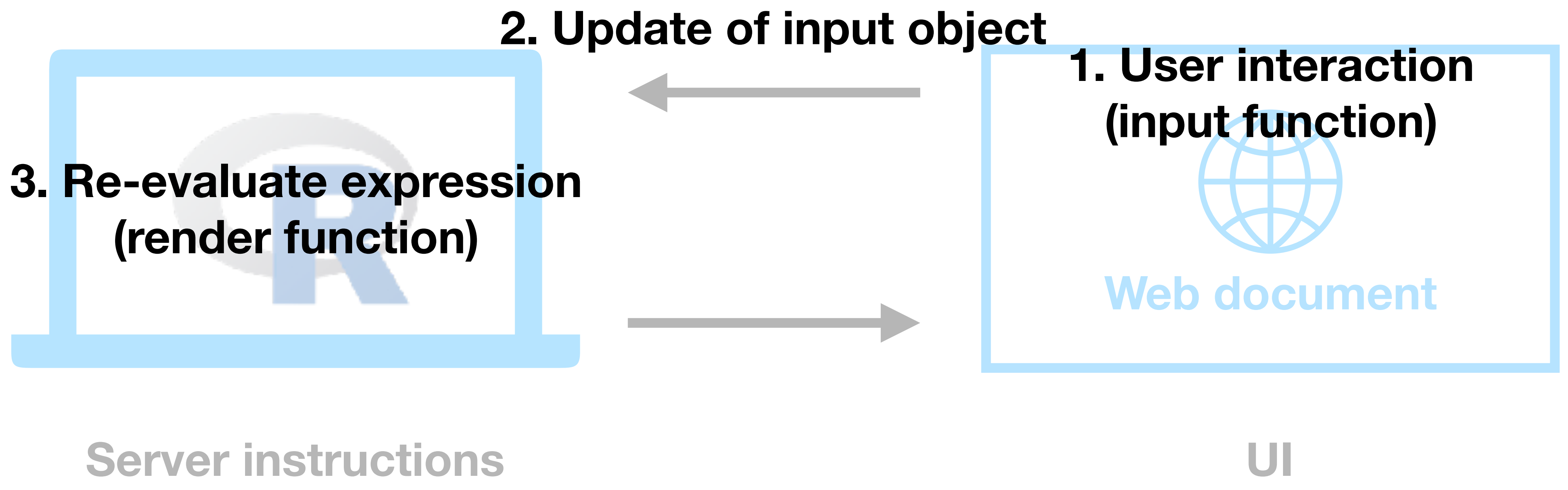
Source: http://shiny.rstudio.com/tutorial/



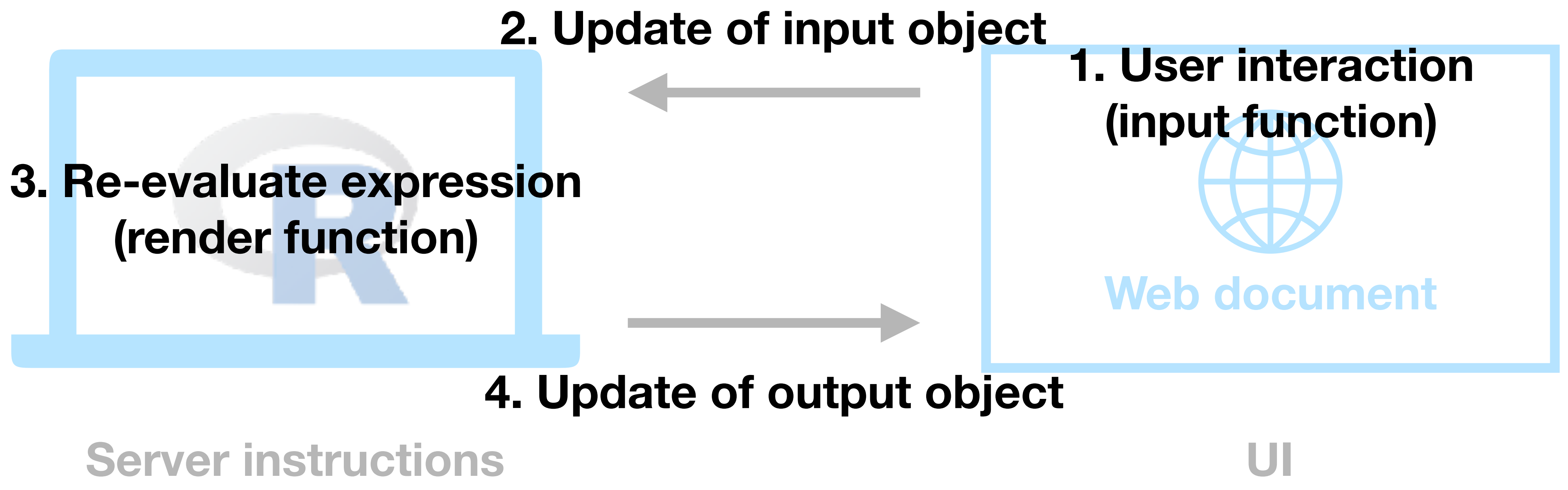
Source: http://shiny.rstudio.com/tutorial/



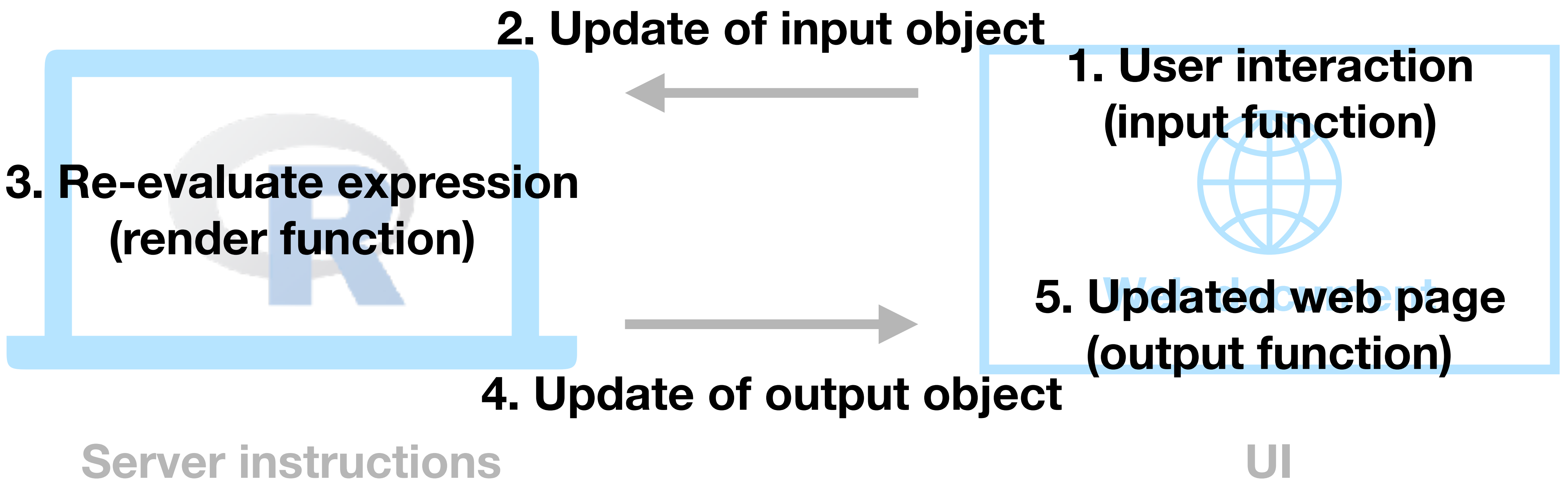
Source: http://shiny.rstudio.com/tutorial/



Source: http://shiny.rstudio.com/tutorial/



Source: http://shiny.rstudio.com/tutorial/



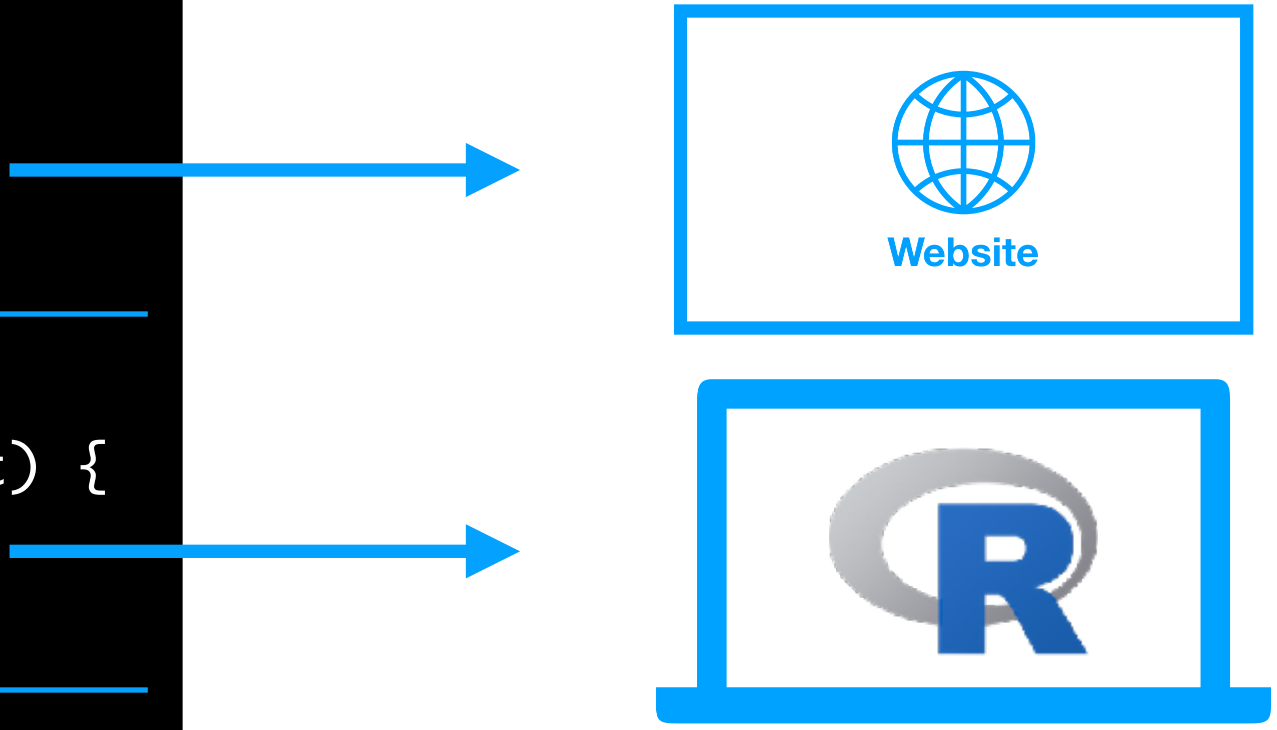
Source: http://shiny.rstudio.com/tutorial/

# Shiny in R script

```
library(shiny)

# UI
ui <- fluidPage(
   # UI elements: inputs, outputs
)

# Server
server <- function(input, output) {
   # (reactive) R expressions
}

# Declare Shiny object
shinyApp(ui = ui, server = server)
```

# Shiny in R script

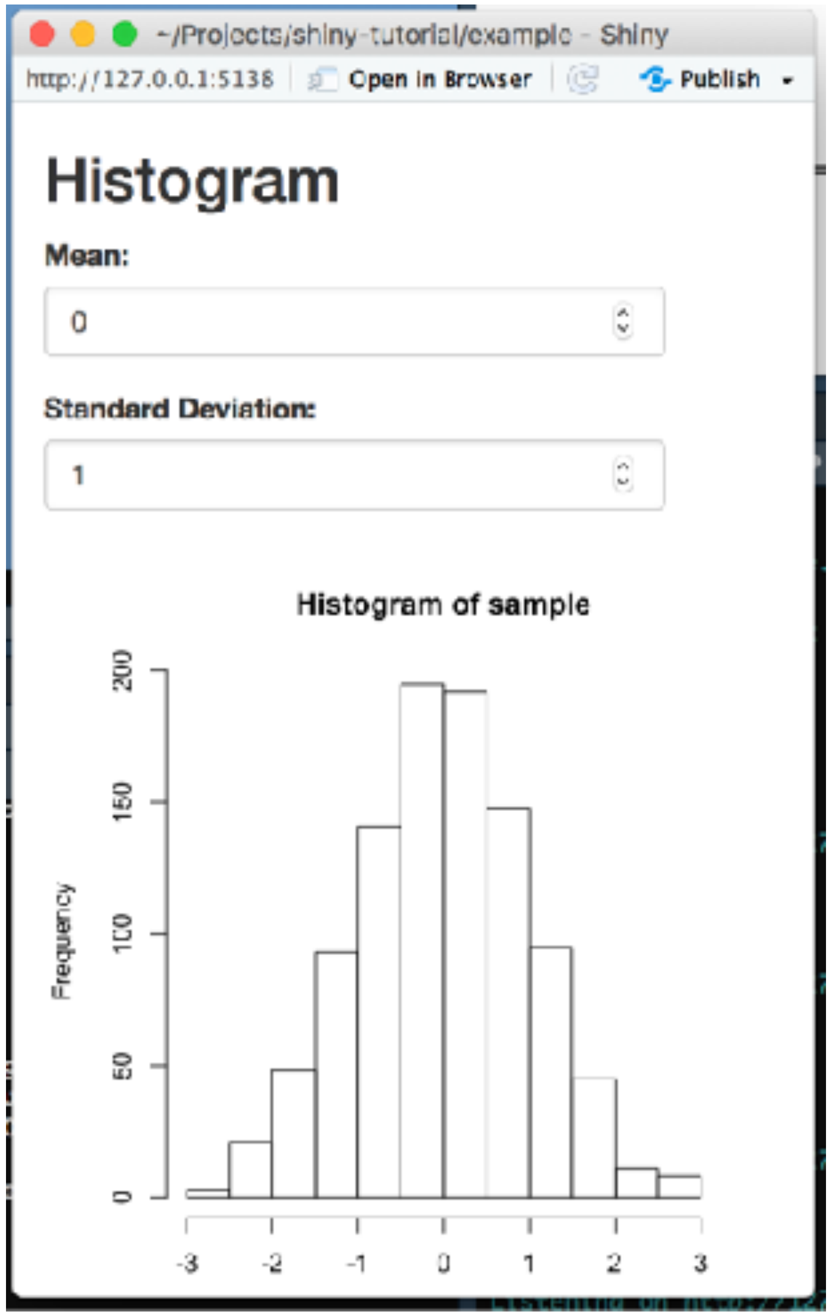
```r
library(shiny)

# UI
ui <- fluidPage(
   # UI elements inputs, outputs
)

# Server
server <- function(input, output) {
   # (reactive) R expressions
}

# Declare Shiny object
shinyApp(ui = ui, server = server)
```



## Let’s check out an example

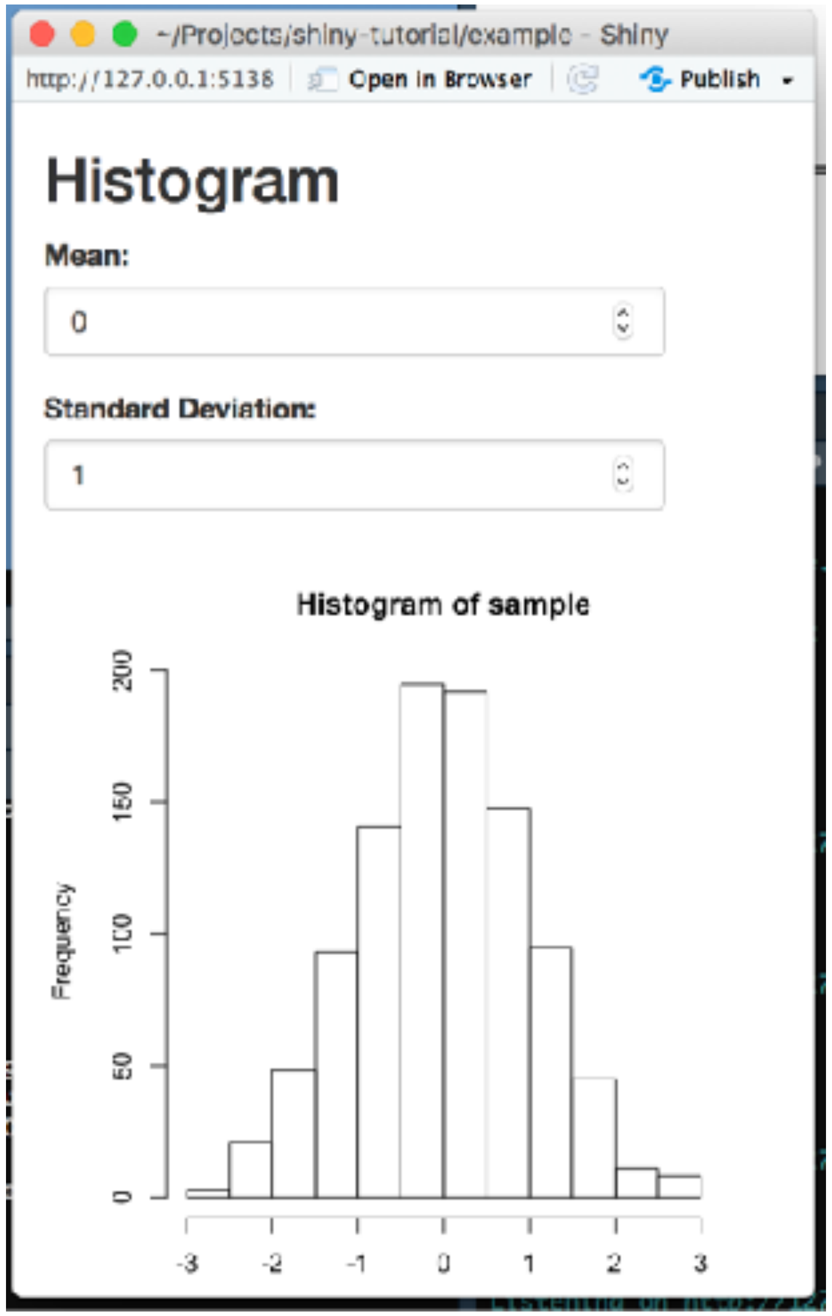
# Input and Output in UI and Server

```
# UI
ui <- fluidPage(
   titlePanel("Histogram"),
   #Input
   numericInput(inputId = "mean",
                label = "Mean:",
                value = 0),
   numericInput(inputId = "sd",
                label = "Standard Deviation:",
                value = 1, min = 0),
   #Output
   plotOutput(outputId = "hist")
)

# Server
server <- function(input, output) {
   output$hist <- renderPlot({
       sample <- rnorm(1000,
                       mean = input$mean,
                       sd = input$sd)
       hist(sample)
   })
}
```



# Input and Output in UI and Server

```r
# UI
ui <- fluidPage(
   titlePanel("Histogram"),
   #Input
   numericInput(inputId = "mean",
                label = "Mean:",
                value = 0),
   numericInput(inputId = "sd",
                label = "Standard Deviation:",
                value = 1, min = 0),
   #Output
   plotOutput(outputId = "hist")
)

# Server
server <- function(input, output) {
   output$hist <- renderPlot({
       sample <- rnorm(1000,
                   mean = input$mean,
                   sd = input$sd)
       hist(sample)
   })
}
```


~/Projects/shiny-tutorial/example - Shiny

http://127.0.0.1:5138 | Open in Browser | Publish

Histogram

Mean:

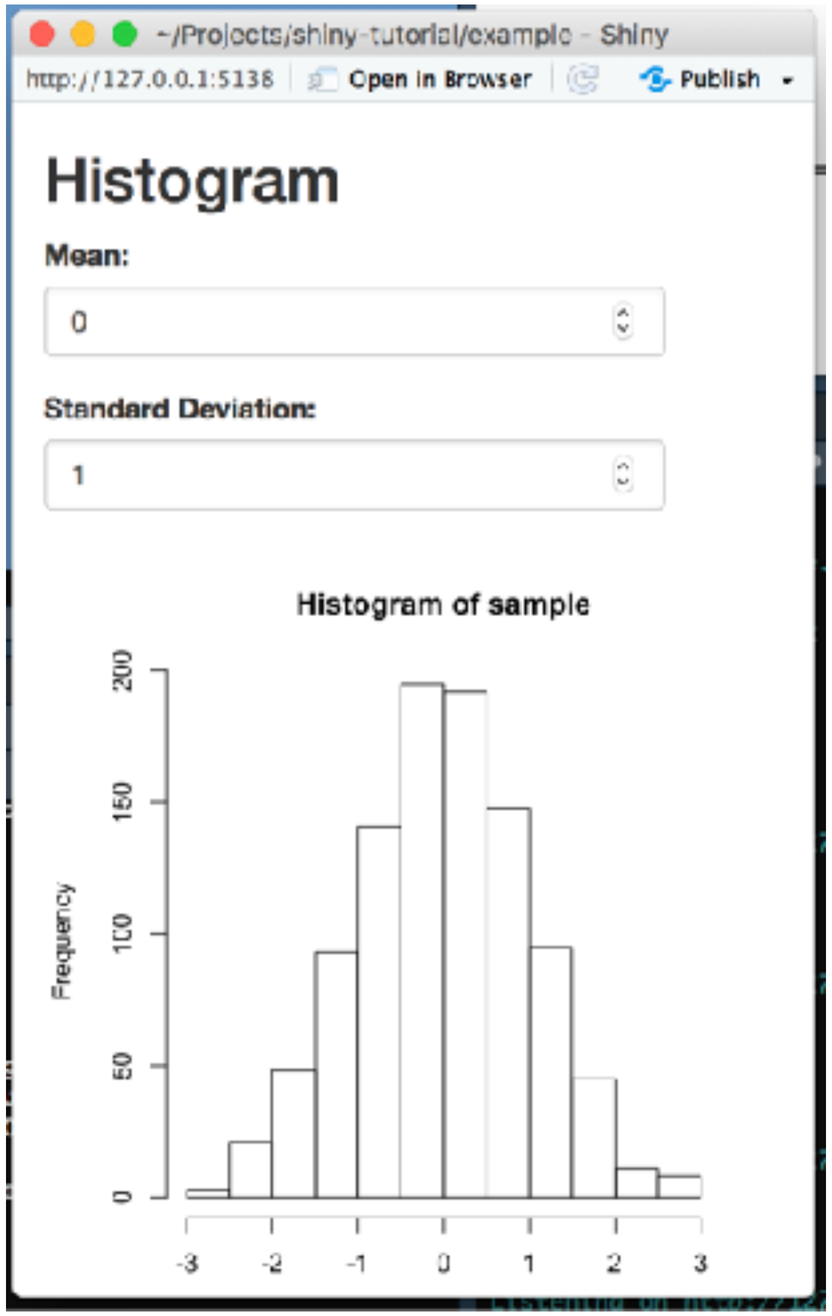
0

Standard Deviation:

1

# Input and Output in UI and Server

```r
# UI
ui <- fluidPage(
   titlePanel("Histogram"),
   #Input
   numericInput(inputId = "mean",
                label = "Mean:",
                value = 0),
   numericInput(inputId = "sd",
                label = "Standard Deviation:",
                value = 1, min = 0),
   #Output
   plotOutput(outputId = "hist")
)

# Server
server <- function(input, output) {
   output$hist <- renderPlot({
       sample <- rnorm(1000,
                      mean = input$mean,
                      sd = input$sd)
       hist(sample)
   })
}
```



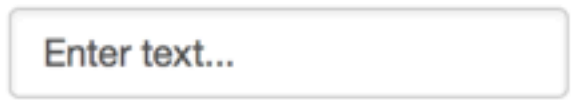
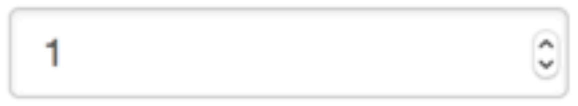
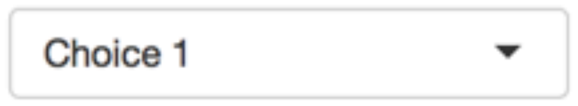
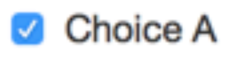
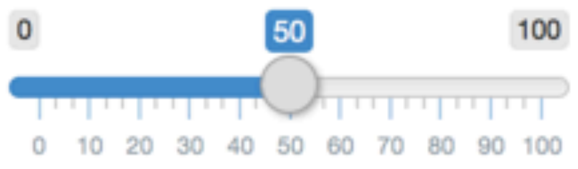
Histogram

Mean: 0

Standard Deviation: 1

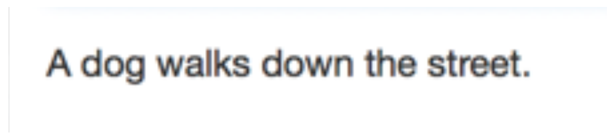
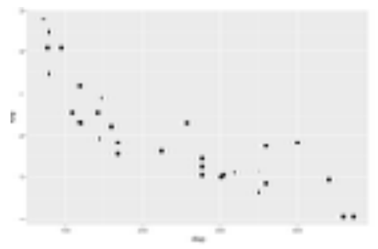
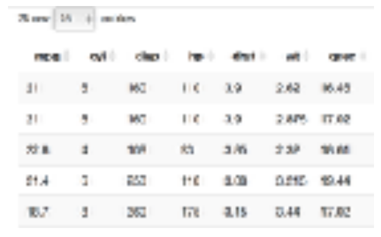
## Shiny Basics:
## Dashboard Ingredients

# Examples of Widgets

| | | |
|---|---|---|
| Text | `textInput()` |  |
| Numeric Value | `numericInput()` |  |
| Drop down | `selectInput()` |  |
| Check box | `checkboxInput()` |  |
| Slider | `sliderInput()` |  |

Source: http://shiny.rstudio.com/tutorial/written-tutorial/lesson3/

# Examples of Outputs

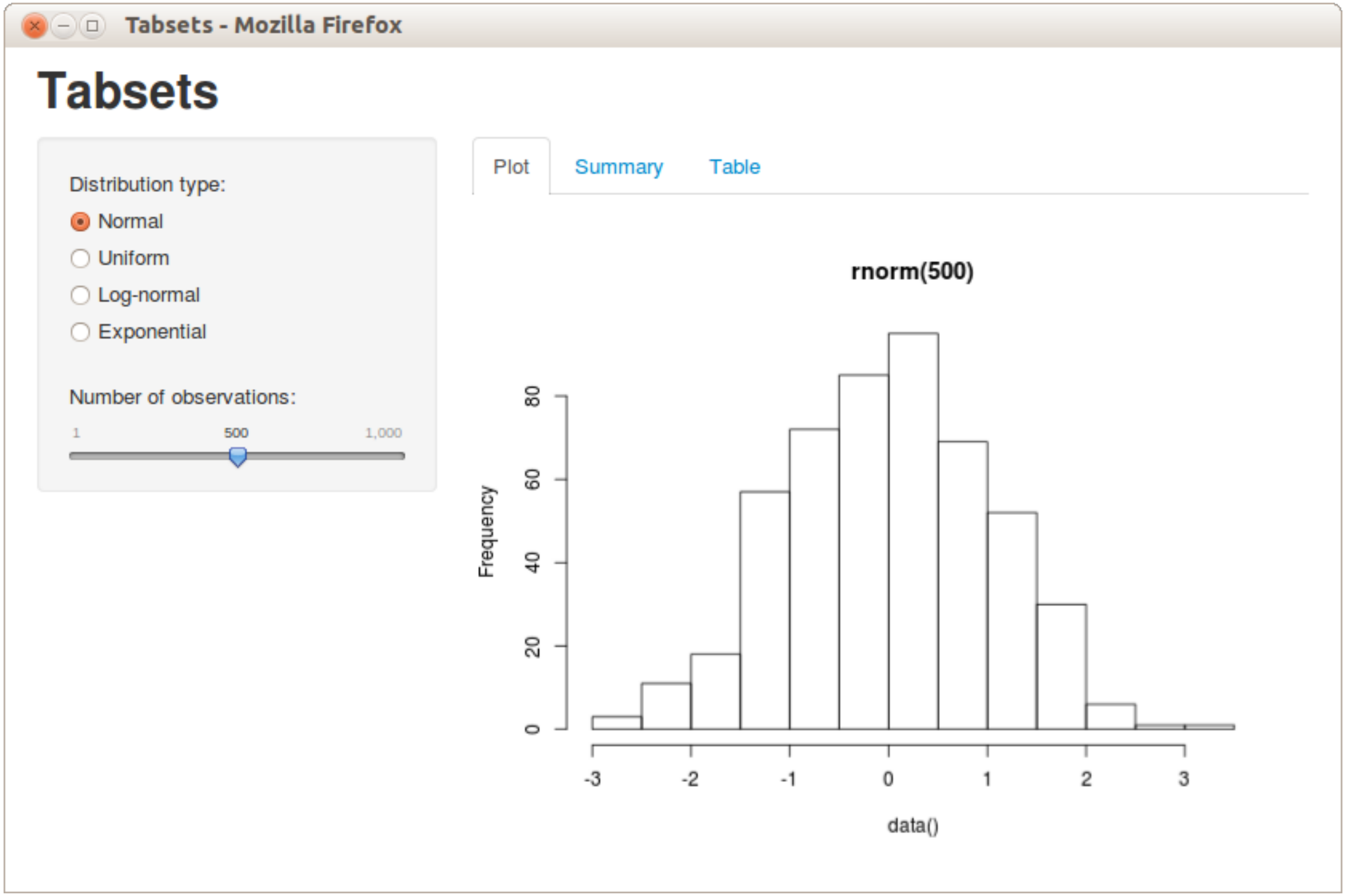
| | | |
|---|---|---|
| Text | `textOutput()` |  A dog walks down the street. |
| Plot | `plotOutput()` |  |
| Data table | `dataTableOutput()` |  |
| Image | `imageOutput()` | |

Show 10 entries

| mpg | cyl | disp | hp | drat | wt | qsec |
|---|---|---|---|---|---|---|
| 21 | 6 | 160 | 110 | 3.9 | 2.62 | 16.46 |
| 21 | 6 | 160 | 110 | 3.9 | 2.875 | 17.02 |
| 22.8 | 4 | 108 | 93 | 3.85 | 2.32 | 18.61 |
| 21.4 | 6 | 258 | 110 | 3.08 | 3.215 | 19.44 |
| 18.7 | 8 | 360 | 175 | 3.15 | 3.44 | 17.02 |

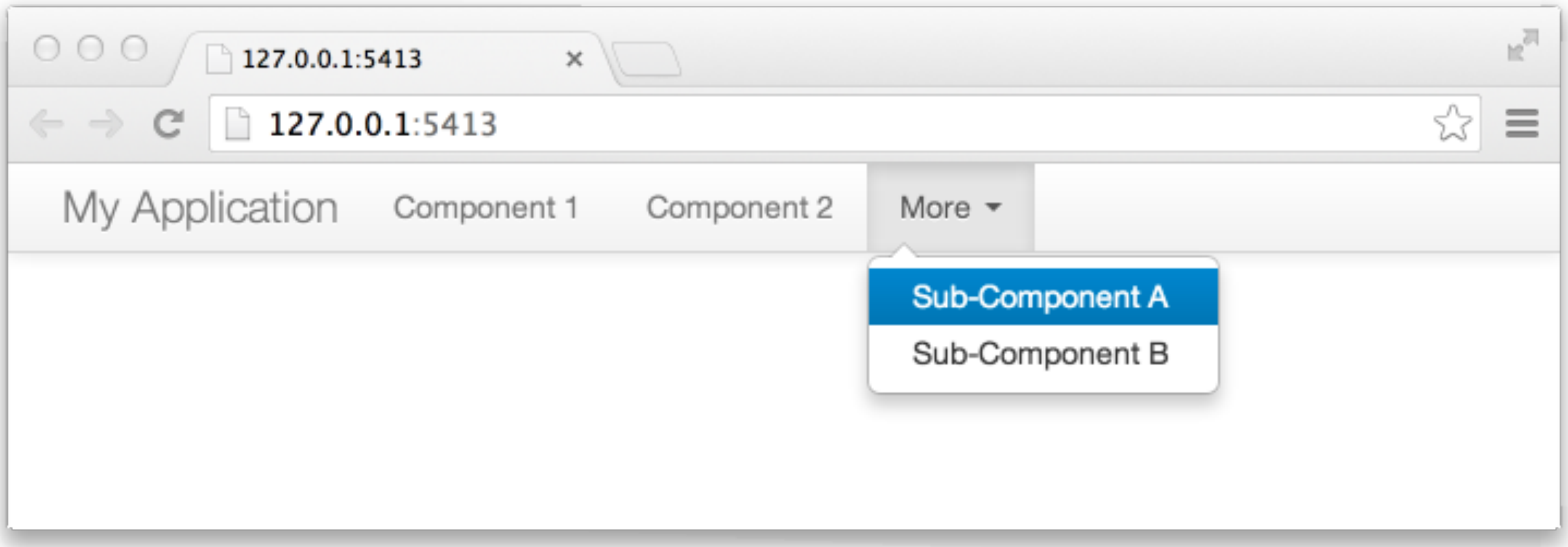
Source: http://shiny.rstudio.com/tutorial/written-tutorial/lesson3/

# Layout Examples



Source: http://shiny.rstudio.com/articles/display-modes.html

# Layout Examples



Source: http://shiny.rstudio.com/articles/display-modes.html

## Adding a layout to our example

# Shiny in 2 R Scripts

**app.R**

```
library(shiny)

# UI
ui <- fluidPage(
   # UI elements
)

# Server
server <- function(input, output) {
   # R expressions
}

# Shiny object
shinyApp(ui = ui, server = server)
```

**ui.R**

```
fluidPage(
   # UI elements
)
```

**server.R**

```
library(shiny)


function(input, output) {
   # R expressions
}
```

## (More) Advanced Shiny

# Manage Reactive Expressions

- Expressions inside render functions react to changes in input values
- We can further manage reactions, e.g.
  - Modularise reactions with `reactive()`
  - Trigger reactions, e.g. with a button
  - Prevent reactions

## Let’s go back to our example

# HTML Elements

- An app’s UI creates an HTML document.
- You can add static HTML elements to your app’s UI

# HTML Elements

- An app's UI creates an HTML document.
- You can add static HTML elements to your app's UI
- `tags` contains list of functions, e.g.
  - `tags$header()`
  - `tags$link()`

## HTML Elements

- An app’s UI creates an HTML document.
- You can add static HTML elements to your app’s UI
- `tags` contains list of functions, e.g.
  - `tags$header()`
  - `tags$link()`
- Common tags have wrapper functions
  - `tags$p() = p()`
  - `tags$strong() = strong()`

# Themes

1. The easy way: `shinythemes` library

```
fluidPage(
   theme = shinytheme(theme = “united”)
   # UI elements
)
```

## Themes

1. The easy way: `shinythemes` library

```
fluidPage(
    theme = shinytheme(theme = “united”)
    # UI elements
)
```

2. The hard way: your custom CSS file

   Include a CSS file in the www subdirectory of the app

```
fluidPage(
    theme =“yourstylesheet.css”
    # UI elements
)
```

# Deploy the App on a Server

Directory structure:

```
<app-name>
 -app.R             (or ui.R and server.R)
 -global.R          (optional)
 -DESCRIPTION       (optional)
 -README            (optional)
 -www               (optional directory)
 -[any other files] (optional, e.g. data)
```

Simply add this directory to a running Shiny server to deploy it.

# Summary (1/2)

- Shiny apps consist of a website and server instructions.

# Summary (1/2)

- Shiny apps consist of a website and server instructions.
- Inputs are send to server, processed in render functions. The results are displayed as outputs on the website.

# Summary (1/2)

- Shiny apps consist of a website and server instructions.
- Inputs are send to server, processed in render functions. The results are displayed as outputs on the website.
- Shiny provides a wide selection of input and output functions.

# Summary (1/2)

- Shiny apps consist of a website and server instructions.
- Inputs are send to server, processed in render functions. The results are displayed as outputs on the website.
- Shiny provides a wide selection of input and output functions.
- You can use different layouts to structure your UI.

## Summary (2/2)

- You can customise the interactive behaviour of your app with functions that manage reactivity.

# Summary (2/2)

- You can customise the interactive behaviour of your app with functions that manage reactivity.
- You can use static HTML elements in your UI.

## Summary (2/2)

- You can customise the interactive behaviour of your app with functions that manage reactivity.
- You can use static HTML elements in your UI.
- You can style your app using the shinythemes library or CSS files.

# Summary (2/2)

- You can customise the interactive behaviour of your app with functions that manage reactivity.
- You can use static HTML elements in your UI.
- You can style your app using the shinythemes library or CSS files.
- Deploying and app on a server is easy. Just stick to the naming conventions.

## Your new best friend:

http://shiny.rstudio.com/articles/cheatsheet.html

# Questions?

**jgraff@babbel.com**

## Let’s get busy!

Clone or download

http://bit.ly/2zfX6AU

# Instructions

The app in the tutorial directory explores the build-in R dataset "mtcars". In this tutorial you will expand this app.

Run the app by either using the `runApp(“tutorial")` command or by hitting "Run App" in the upper right corner of the script editor. Explore the interactive behaviour of the app and how it is implemented codewise.

**1st task**
Improve the layout of the tab "Scatter Plots" by moving the input elements into a side bar and the plot into the main panel. You can do this using the sidebarLayout function.

**2nd task**
Add a check box to the sidebar panel that allows the user to add a regression line to the plot.

*Hint*: You can add `geom_smooth(method = "lm")` to the ggplot object to add a regression line to the plot.

**3rd task**
The "Data" tab is blank. Implement the following:
The tab should display the mtcars data in a table. The sidebar panel should allow user to filter the data for specific levels of the variables cyl and gear (e.g. drop down menus or radio buttons).