

# Taking a shine to Shiny

## Building R powered web applications

Shlomi Lifshits

Tel Aviv university

May 30, 2014



# Outline

- 1 Introduction
- 2 Hello World
- 3 Understanding reactivity
- 4 Examples

# Cloud services

- Cloud storage
  - ▶ Google BigQuery (package RBigquery).
  - ▶ Infochimps (package infochimps)
- Web services
  - ▶ Yahoo finance and Google finance (package quantmod).
  - ▶ World Bank (package WDI).
  - ▶ Twitter (package twitterR).
  - ▶ more...
- Cloud computation
  - ▶ Amazon EC2
- Cloud visualization and exploration
  - ▶ **Shiny**

<http://cran.r-project.org/web/views/WebTechnologies.html>.

# R web solutions

- Knitr <http://www.rpubs.com/>.
- R2HTML
- rApache : Embed R in the Apache web server <http://rapache.net/>
- Rook :Provides an interface to build web-applications with rApache  
<http://cran.r-project.org/web/packages/Rook/index.html>
- WebDevelopR
- gWidgetsWWW
- FastRWeb <http://rforge.net/FastRWeb/>

# R interactive graphics is using web technology

- Google charts (<https://developers.google.com/chart/interactive/docs/gallery>)
  - ▶ The googleVis package provides an interface between R and the Google Chart Tools API.
- package gridSVG
- package ggvis by Hadley Wickham.

# Source file repository on the cloud GitHub

Many of Shiny examples are located on GitHub as:

```
library(shiny)
```

- Repository

```
runGitHub(repo="kmeans",username="jcheng5")
```

- Folder in a repository

```
runGitHub(repo="shiny-examples",username="rstudio",subdir="001-hello")
```

- Gist

```
runGist("https://gist.github.com/shlomyli/589e4c41f1967192f4e9")
```

# Shiny developer site

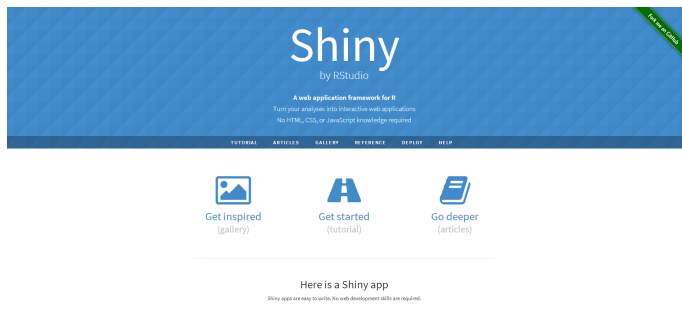


Figure : <http://shiny.rstudio.com/>

# Showcase

- <http://www.rstudio.com/shiny/showcase/>
- <http://shiny.rstudio.com/gallery/>
- <https://github.com/rstudio/shiny/wiki/RStudio-Shiny-Apps>
- <https://github.com/rstudio/shiny-examples>



# Tutorials

- <http://www.rstudio.com/shiny/lessons/Intro/>
- <http://shiny.rstudio.com/tutorial/>
- (deprecated) <http://rstudio.github.io/shiny/tutorial/>

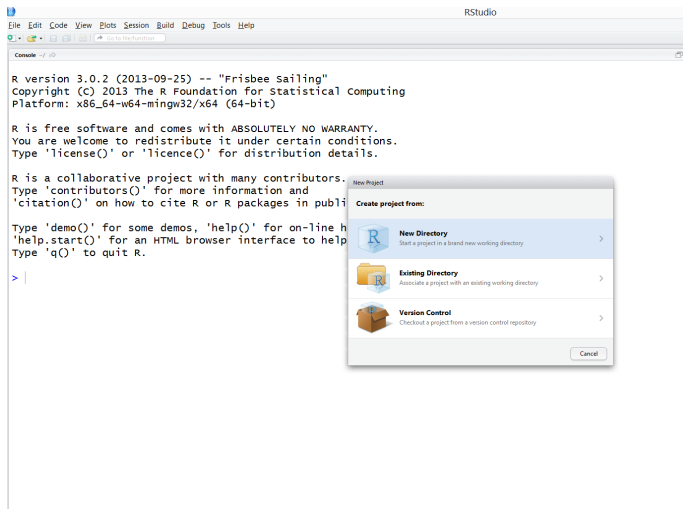
# Required skills

- You need to know R.
- Knowledge in HTML, Java script, CSS is an advantage.

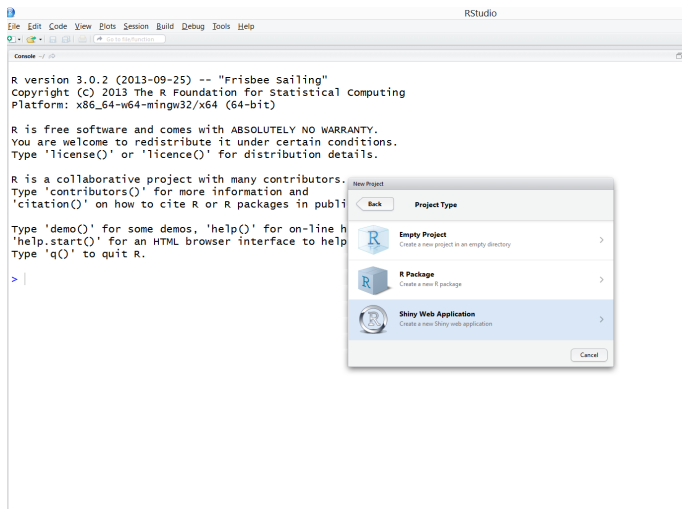
# Development cycle

- ➊ The server and client side is developed locally (using the single client, web server embedded in RStudio).
- ➋ Debugging is possible.
- ➌ Deployment:
  - ▶ RStudio Shiny servers (multiple clients, authorization).
  - ▶ Your Shiny server (free for Linux).
  - ▶ Amazon EC2

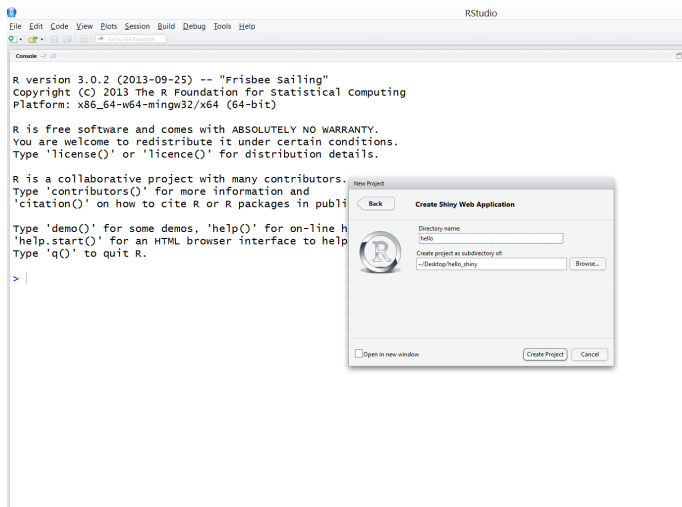
# Hello Shiny



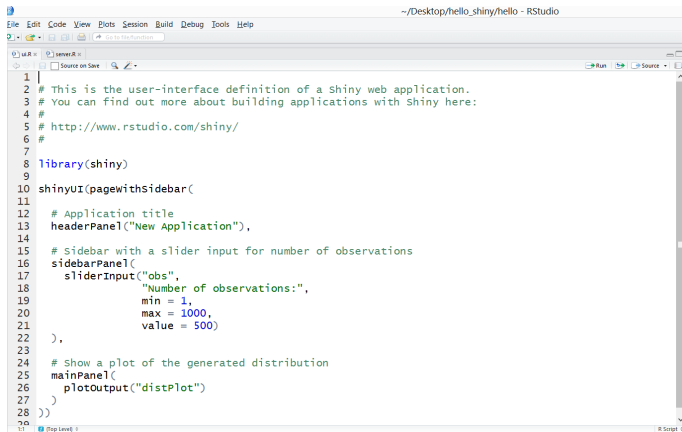
# Hello Shiny



# Hello Shiny



# Hello Shiny

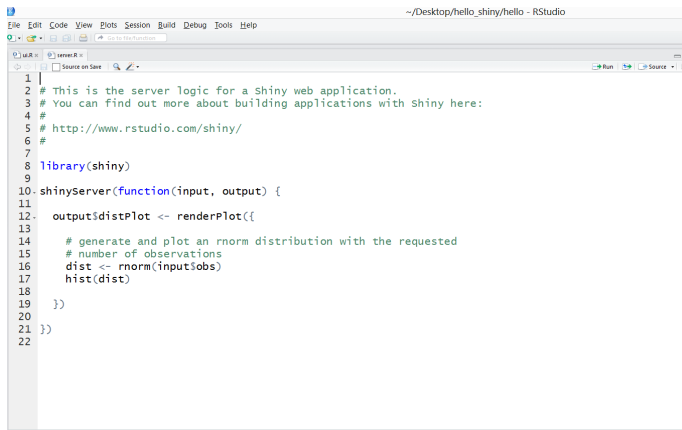


The screenshot shows the RStudio interface with a file named `~/Desktop/hello_shiny/hello - RStudio` open. The editor displays the following R code for a Shiny web application:

```
1 |
2 # This is the user-interface definition of a Shiny web application.
3 # You can find out more about building applications with Shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7
8 library(shiny)
9
10 shinyUI(pageWithSidebar(
11
12   # Application title
13   headerPanel("New Application"),
14
15   # Sidebar with a slider input for number of observations
16   sidebarPanel(
17     sliderInput("obs",
18               "Number of observations:",
19               min = 1,
20               max = 1000,
21               value = 500)
22   ),
23
24   # Show a plot of the generated distribution
25   mainPanel(
26     plotOutput("distPlot")
27   )
28 ))
```

The status bar at the bottom indicates the script is at line 11, column 17, and is at the top level.

# Hello Shiny

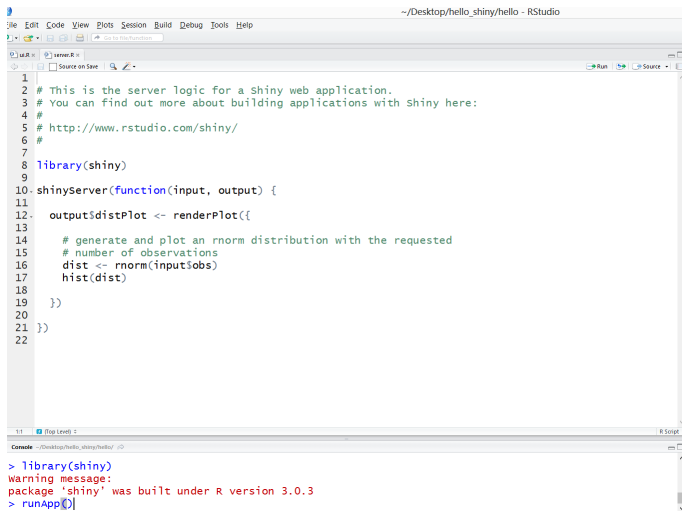


The screenshot shows the RStudio interface with a file named "server.R" open. The code is written in R and defines a shinyServer function. The function takes input and output objects. It uses the shiny library and the renderPlot function to create a histogram. The histogram is generated from a random normal distribution with a mean of 0 and a standard deviation of 1, with the number of observations equal to the input value. The histogram is then rendered as a plot.

```
1 |
2 # This is the server logic for a Shiny web application.
3 # You can find out more about building applications with shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7
8 library(shiny)
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate and plot an rnorm distribution with the requested
15     # number of observations
16     dist <- rnorm(input$obs)
17     hist(dist)
18
19   })
20
21 })
22
```



# Hello Shiny



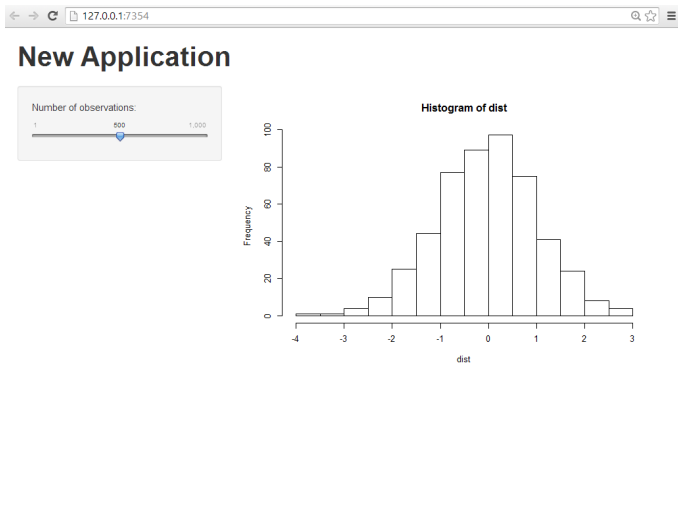
The screenshot shows the RStudio interface with a file named `~/Desktop/hello_shiny/hello - RStudio`. The editor displays a Shiny server script with the following code:

```
1
2 # This is the server logic for a Shiny web application.
3 # You can find out more about building applications with Shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7 library(shiny)
8
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate and plot an rnorm distribution with the requested
15     # number of observations
16     dist <- rnorm(input$obs)
17     hist(dist)
18
19   })
20
21 })
22
```

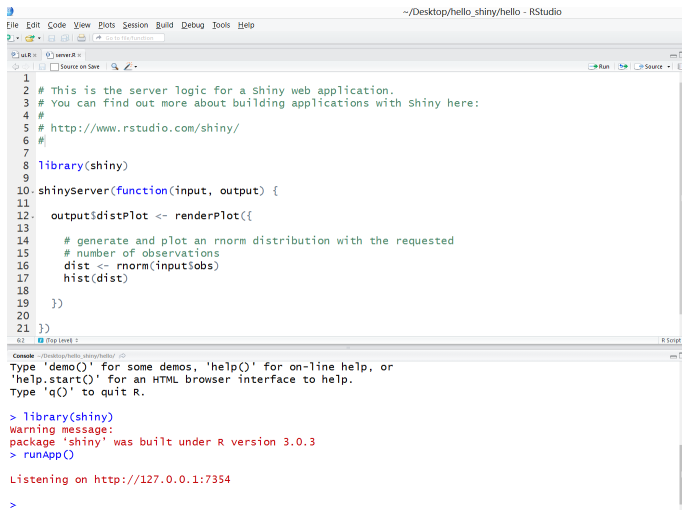
The console at the bottom shows the execution of the script:

```
1:1 [Top Level]
> library(shiny)
warning message:
package 'shiny' was built under R version 3.0.3
> runApp()
```

# Hello Shiny



# Hello Shiny



The screenshot shows the RStudio interface with a file named `~/Desktop/hello_shiny/hello - RStudio`. The script editor contains the following R code:

```
1
2 # This is the server logic for a Shiny web application.
3 # You can find out more about building applications with Shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7
8 library(shiny)
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate and plot an rnorm distribution with the requested
15     # number of observations
16     dist <- rnorm(input$obs)
17     hist(dist)
18
19   })
20
21 })
```

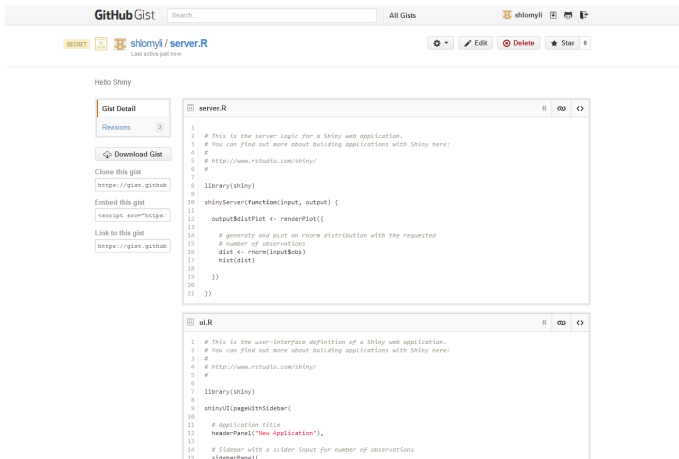
The console window shows the following output:

```
> library(shiny)
warning message:
package 'shiny' was built under R version 3.0.3
> runApp()

Listening on http://127.0.0.1:7354

>
```

# Hello Shiny



GitHub Gist

Search...

All Gists

shlomyli

SECRET

shlomyli / server.R

Last active just now

Settings Edit Delete Star 0

Hello Shiny

Gist Detail

Revisions 2

Download Gist

Clone this gist

<https://gist.github.com>

Embed this gist

<script async src="https://gist.github.com"></script>

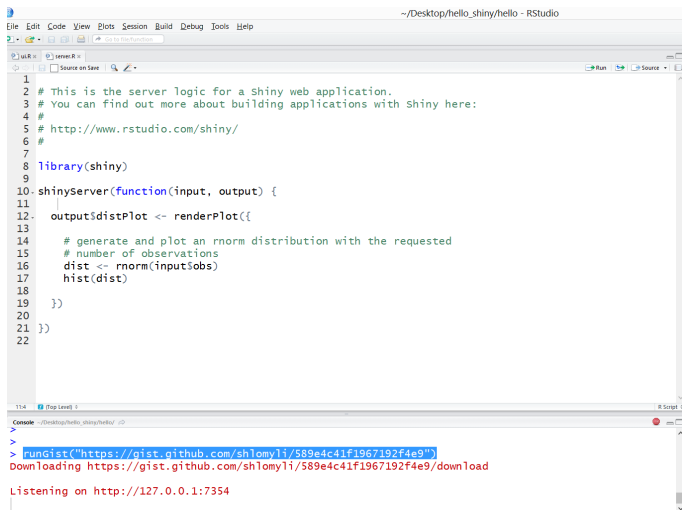
Link to this gist

<https://gist.github.com>

```
server.R
1
2 # This is the server logic for a Shiny web application.
3 # You can find out more about building applications with Shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7 library(shiny)
8
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate and plot an rnorm distribution with the requested
15     # number of observations
16     dist <- rnorm(input$numObs)
17     hist(dist)
18   })
19
20
21 })
```

```
ui.R
1
2 # This is the user-interface definition of a Shiny web application.
3 # You can find out more about building applications with Shiny here:
4 #
5 # http://www.rstudio.com/shiny/
6 #
7 library(shiny)
8
9 shinyUI(pageWithSidebar(
10
11   # Application title
12   headerPanel("New Application"),
13
14   # Sidebar with a slider input for number of observations
15   sidebarPanel()
```

# Hello Shiny



The screenshot shows the RStudio interface with a file named `~/Desktop/hello_shiny/hello - RStudio`. The script editor contains the following R code:

```
1 # This is the server logic for a shiny web application.
2 # You can find out more about building applications with shiny here:
3 #
4 # http://www.rstudio.com/shiny/
5 #
6
7 library(shiny)
8
9
10 shinyServer(function(input, output) {
11
12   output$distPlot <- renderPlot({
13
14     # generate and plot an rnorm distribution with the requested
15     # number of observations
16     dist <- rnorm(input$obs)
17     hist(dist)
18
19   })
20
21 })
22
```

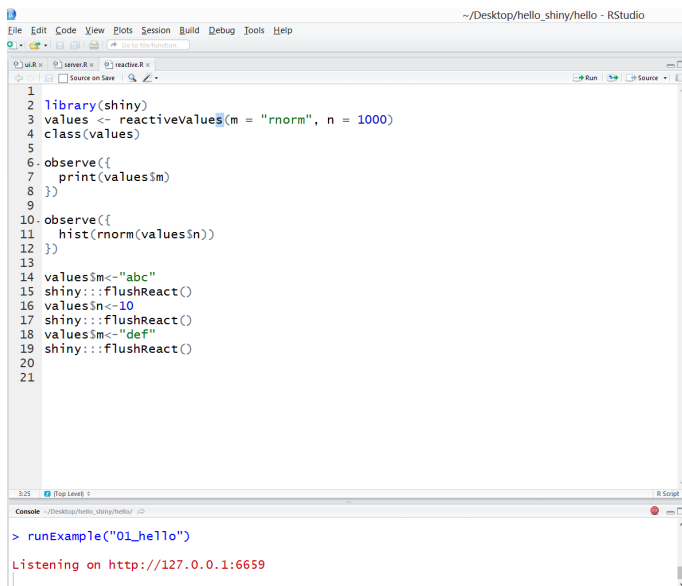
The console at the bottom shows the following output:

```
11:4 [Top Level]
>
> runGist("https://gist.github.com/shlomyli/589e4c41f1967192f4e9")
Downloading https://gist.github.com/shlomyli/589e4c41f1967192f4e9/download
Listening on http://127.0.0.1:7354
```

# Hello Shiny

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
5 <script src="shared/jquery.js" type="text/javascript"></script>
6 <script src="shared/shiny.js" type="text/javascript"></script>
7 <link rel="stylesheet" type="text/css" href="shared/shiny.css"/>
8 <script type="application/shiny-singletons">d9824d41b9a6ae7e83ba073d83925e0d8434247</script>
9 <link rel="stylesheet" type="text/css" href="shared/bootstrap/css/bootstrap.min.css"/>
10 <script src="shared/bootstrap/js/bootstrap.min.js"></script>
11 <meta name="viewport" content="width=device-width, initial-scale=1.0"/>
12 <link rel="stylesheet" type="text/css" href="shared/bootstrap/css/bootstrap-responsive.min.css"/>
13 <title>New Application</title>
14 <link rel="stylesheet" type="text/css" href="shared/slider/css/jquery.slider.min.css"/>
15 <script src="shared/slider/js/jquery.slider.min.js"></script>
16 </head>
17 <body>
18 <div class="container-fluid">
19 <div class="row-fluid">
20 <div class="span12" style="padding: 10px 0px;">
21 <h1>New Application</h1>
22 </div>
23 </div>
24 <div class="row-fluid">
25 <div class="span4">
26 <form class="well">
27 <div>
28 <label class="control-label" for="obs">Number of observations:</label>
29 <input id="obs" type="slider" name="obs" value="500" class="jquery.slider" data-from="1" data-to="1000" data-step="1"
30 data-skin="plastic" data-round="FALSE" data-locale="us" data-format="#,#0.####" data-smooth="FALSE"/>
31 </div>
32 </form>
33 </div>
34 <div class="span8">
35 <div id="distPlot" class="shiny-plot-output" style="width: 100%; height: 400px"></div>
36 </div>
37 </div>
38 </body>
39 </html>
```

# Hello Shiny



The screenshot shows the RStudio interface with a script editor and a console. The script editor contains the following R code:

```
1  
2 library(shiny)  
3 values <- reactiveValues(m = "rnorm", n = 1000)  
4 class(values)  
5  
6 observe({  
7   print(values$m)  
8 })  
9  
10 observe({  
11   hist(rnorm(values$n))  
12 })  
13  
14 values$m<-"abc"  
15 shiny::flushReact()  
16 values$n<-10  
17 shiny::flushReact()  
18 values$m<-"def"  
19 shiny::flushReact()  
20  
21
```

The console at the bottom shows the following output:

```
> runExample("01_hello")  
Listening on http://127.0.0.1:6659
```

# Shiny app

A Shiny application is comprised of two files:

- `ui.R` : describes the graphical design of the web page, the widgets and their organization.
- `server.R`: describes the logic behind (data processing, statistical computations).



# HTML form

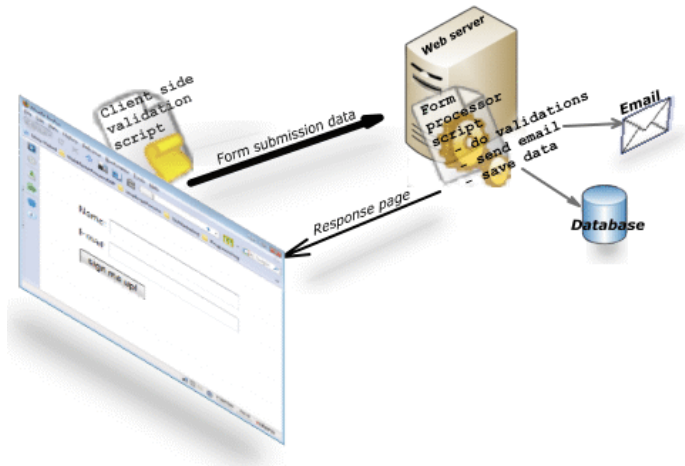


Figure : From: <http://www.javascript-coder.com/html-form/html-form-tutorial-p1.phtml>.

# Shiny is a reactive application

- The graphical user interface *reacts* to user selections.
- An object saves a list of expressions that depend on it.
- When the value of an object is changed it will change the status of these expression that depend on it to *invalid* to (need update).
- Shiny server will cycle through all expression, selectively updating expressions.

## Understanding reactivity

```
library(shiny)
shinyServer(function(input, output) {
  output$distPlot <- renderPlot({
    dist <- rnorm(input$obs)
    hist(dist)
  })
})
```

The expression is wrapped in a call to `renderPlot` to indicate that:

- It is "reactive" and therefore should re-execute automatically when inputs change.
- Its output is a plot.

# Shiny Widgets

The screenshot displays the Shiny Widgets gallery, a collection of interactive UI components. The gallery is organized into a grid of 12 panels, each demonstrating a different widget. The top of the gallery features a blue header with the Shiny logo and navigation links. Each panel includes a title, a brief description, a visual representation of the widget, and a 'View Code' button.

- Action button:** A button labeled 'Action'.
- Single checkbox:** A checkbox labeled 'Choice 1'.
- Checkbox group:** A group of three checkboxes labeled 'Choice 1', 'Choice 2', and 'Choice 3'.
- Date input:** A date input field showing '2014-03-01'.
- Date range:** A date range input field showing '2014-03-01' to '2014-03-02'.
- File input:** A file input field with a 'Choose File' button.
- Numeric input:** A numeric input field showing '1'.
- Radio buttons:** A group of three radio buttons labeled 'Choice 1', 'Choice 2', and 'Choice 3'.
- Select box:** A select box showing 'Choice 1'.
- Slider:** A slider input field showing a value of 10.
- Slider range:** A slider range input field showing a range from 10 to 20.
- Text input:** A text input field showing 'Enter text...'.

# kmeans

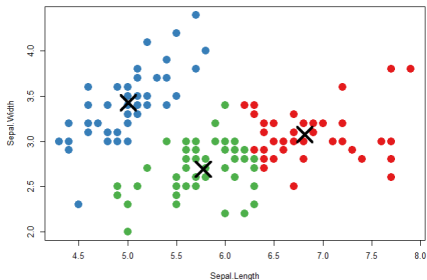
Written by Joe Cheng (RStudio).

## Iris k-means clustering

X Variable  
Sepal.Length

Y Variable  
Sepal.Width

Cluster count  
3



```
runGitHub(runGitHub("kmeans", "jcheng5"))
```

# server.R

```
palette(c("#E41A1C", "#377EB8", "#4DAF4A", "#984EA3",
          "#FF7F00", "#FFFF33", "#A65628", "#F781BF", "#999999"))

shinyServer(function(input, output, session) {

  # Combine the selected variables into a new data frame
  selectedData <- reactive({
    iris[, c(input$xcol, input$ycol)]
  })

  clusters <- reactive({
    kmeans(selectedData(), input$clusters)
  })

  output$plot1 <- renderPlot({
    par(mar = c(5.1, 4.1, 0, 1))
    plot(selectedData(),
          col = clusters()$cluster,
          pch = 20, cex = 3)
    points(clusters()$centers, pch = 4, cex = 4, lwd = 4)
  })
})
```

```
shinyUI(pageWithSidebar(  
  headerPanel('Iris k-means clustering'),  
  sidebarPanel(  
    selectInput('xcol', 'X Variable', names(iris)),  
    selectInput('ycol', 'Y Variable', names(iris),  
               selected=names(iris)[[2]]),  
    numericInput('clusters', 'Cluster count', 3,  
                 min = 1, max = 9)  
  ),  
  mainPanel(  
    plotOutput('plot1')  
  )  
))
```

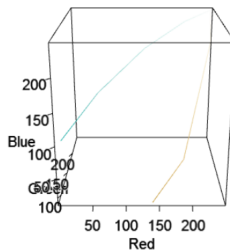
Example by Jeff Allen (Trestle Technology)

## WebGL Demo

Color Palette  
 BrBG

☐ Display principal components.

All source available on [Github](#)

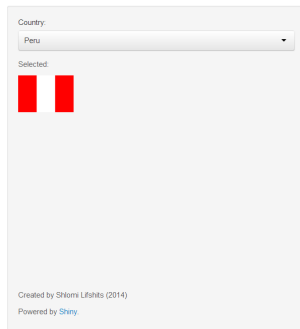


```
runGitHub("shiny-sandbox", "trestletech", subdir="rgl")
```



## Example by Shlomi Lifshits

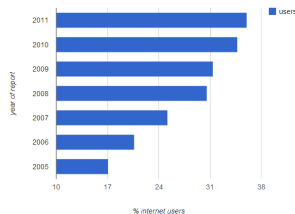
### Rate of change in number of internet users across the world



[1] "Last report: 36 %"



<http://data.worldbank.org/indicator>



Formula  $\frac{x}{y} \cdot 100$ , where x is the total number of users and y is the total population size.

```
runGitHub(repo="shiny",username="shlomyli",subdir="WDI")
```

- Apply the data query through WDI.
- Getting the flag of the selected country (HTML).
- Rendering formulae using MathJax.
- Using interactive plots from Google Charts.

## Example by Shlomi Lifshits

## Pairwise scatter plot

Import data from Excel:

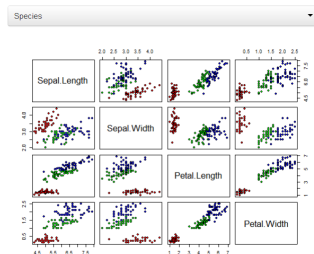
```

181 6.9 3.1 5.4 2.1 virginica
182 6.7 3.1 5.6 2.4 virginica
183 6.9 3.1 5.1 2.3 virginica
184 5.8 2.7 5.1 1.9 virginica
185 6.8 3.2 5.9 2.3 virginica
186 6.7 3.3 5.7 2.5 virginica
187 6.7 3 5.2 2.3 virginica
188 6.3 2.5 5 1.9 virginica
189 6.5 3 5.2 2 virginica
190 6.2 3.4 5.4 2.3 virginica
191 5.9 3 5.1 1.8 virginica
192

```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3	1.4	0.1	setosa
4.3	3	1.1	0.1	setosa
5.8	4	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa

Choose class name



```
runGitHub(repo="shiny",username="shlomyli",subdir="Excel")
```

# Excel

- Import an Excel spreadsheet by Copy and paste.
- Dynamic UI
- Using interactive tables from Google Charts.

# Thank you for your attention

- Shlomi Lifshits
- e-mail: [shlomi.lifshits@yahoo.com](mailto:shlomi.lifshits@yahoo.com)
- Homepage: <https://sites.google.com/site/shlomilifshits/>
- Linked-in:  
<http://www.linkedin.com/pub/shlomi-lifshits/6/a33/408>