All Training materials are provided "as is" and without warranty and RStudio disclaims any and all express and implied warranties including without limitation the implied warranties of title, fitness for a particular purpose, merchantability and noninfringement.

The Training Materials are licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit [http://creativecommons.org/licenses/by-nc/3.0/us/](http://creativecommons.org/licenses/by-nc/3.0/us/) or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.
How to start with Shiny, Part 1
How to build a Shiny App

Garrett Grolemund
Data Scientist and Master Instructor
May 2015
Email: garrett@rstudio.com
Code and slides at:

bit.ly/shiny-quickstart-1
How to start with Shiny

1. How to build a Shiny app (Today)
2. How to customize reactions (May 27)
3. How to customize appearance (June 3)
Shiny Showcase

www.rstudio.com/products/shiny/shiny-user-showcase/
Learn R
Understand the architecture
Every Shiny app is maintained by a computer running R
Every Shiny app is maintained by a computer running R
Server Instructions

User Interface (UI)
Use the template
library(shiny)

ui <- fluidPage()

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
Close an app
Add elements to your app as arguments to fluidPage()

```r
library(shiny)
ui <- fluidPage("Hello World")

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
```
Build your app around Inputs and Outputs
Build your app around **inputs** and **outputs**
Build your app around **inputs** and **outputs**
Add elements to your app as arguments to `fluidPage()`

```r
ui <- fluidPage(
  # *Input() functions,
  # *Output() functions
)
```
Inputs
Create an input with an *Input() function.

```r
sliderInput(inputId = "num",
            label = "Choose a number",
            value = 25, min = 1, max = 100)
```

```html
<div class="form-group shiny-input-container">
  <label class="control-label" for="num">Choose a number</label>
  <input class="js-range-slider" id="num" data-min="1" data-max="100"
         data-from="25" data-step="1" data-grid="true" data-grid-num="9.9"
         data-grid-snap="false" data-prettify-separator="," data-keyboard="true"
         data-keyboard-step="1.01010101010101"/>
</div>
```
Create an input with an input function.

```r
library(shiny)
ui <- fluidPage(

)

server <- function(input, output) {}

shinyApp(server = server, ui = ui)
```
library(shiny)
ui <- fluidPage(
  sliderInput(inputId = "num",
              label = "Choose a number",
              value = 25, min = 1, max = 100)
)

server <- function(input, output) {}

shinyApp(server = server, ui = ui)
Syntax

```r
sliderInput(inputId = "num", label = "Choose a number", ...)
```

- **input name** (for internal use)
- **Notice:** Id not ID
- **label to display**
- **input specific arguments**

?sliderInput
Outputs
Build your app around inputs and outputs
<table>
<thead>
<tr>
<th>Function</th>
<th>Inserts</th>
</tr>
</thead>
<tbody>
<tr>
<td>dataTableOutput()</td>
<td>an interactive table</td>
</tr>
<tr>
<td>htmlOutput()</td>
<td>raw HTML</td>
</tr>
<tr>
<td>imageOutput()</td>
<td>image</td>
</tr>
<tr>
<td>plotOutput()</td>
<td>plot</td>
</tr>
<tr>
<td>tableOutput()</td>
<td>table</td>
</tr>
<tr>
<td>textOutput()</td>
<td>text</td>
</tr>
<tr>
<td>uiOutput()</td>
<td>a Shiny UI element</td>
</tr>
<tr>
<td>verbatimTextOutput()</td>
<td>text</td>
</tr>
</tbody>
</table>
*Output()

To display output, add it to `fluidPage()` with an *Output() function

`plotOutput("hist")`

- the type of output to display
- name to give to the output object
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
              label = "Choose a number",
              value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {}

shinyApp(ui = ui, server = server)
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
              label = "Choose a number",
              value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {}

shinyApp(ui = ui, server = server)

* Output() adds a space in the ui for an R object.
library(shiny)

ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100),
  plotOutput("hist")
)

server <- function(input, output) {}

shinyApp(ui = ui, server = server)

* Output() adds a space in the ui for an R object.

You must build the object in the server function
Recap

Begin each app with the template

```
library(shiny)
ui <- fluidPage()
server <- function(input, output) {}
shinyApp(ui = ui, server = server)
```

Add elements as arguments to `fluidPage()`

Create reactive inputs with an `*Input()` function

Display reactive results with an `*Output()` function

Assemble outputs from inputs in the server function
Tell the server how to assemble inputs into outputs
Use **3 rules** to write the server function

```r
server <- function(input, output) {
}
```
Save objects to display to output$

```r
server <- function(input, output) {
  output$hist <- # code
}
```
Save objects to display to output

\[
\text{output$hist} \\
\text{plotOutput("hist")}
\]
Build objects to display with `render*()`

```r
server <- function(input, output) {
  output$hist <- renderPlot({
    
  })
}
```
Use the `render*()` function that creates the type of output you wish to make.

<table>
<thead>
<tr>
<th>function</th>
<th>creates</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>renderDataTable()</code></td>
<td>An interactive table (from a data frame, matrix, or other table-like structure)</td>
</tr>
<tr>
<td><code>renderImage()</code></td>
<td>An image (saved as a link to a source file)</td>
</tr>
<tr>
<td><code>renderPlot()</code></td>
<td>A plot</td>
</tr>
<tr>
<td><code>renderPrint()</code></td>
<td>A code block of printed output</td>
</tr>
<tr>
<td><code>renderTable()</code></td>
<td>A table (from a data frame, matrix, or other table-like structure)</td>
</tr>
<tr>
<td><code>renderText()</code></td>
<td>A character string</td>
</tr>
<tr>
<td><code>renderUI()</code></td>
<td>a Shiny UI element</td>
</tr>
</tbody>
</table>
render*() 
Builds reactive output to display in UI

```r
renderPlot({ hist(rnorm(100)) })
```

- type of object to build
- code block that builds the object
Build objects to display with `render*()`

```r
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(100))
  })
}
```
Build objects to display with `render*()`

```r
server <- function(input, output) {
  output$hist <- renderPlot({
    title <- "100 random normal values"
    hist(rnorm(100), main = title)
  })
}
```
Access **input** values with `input$`

```r
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}
```
Access `input` values with `input$`.

`sliderInput(inputId = "num", ...)`

`input$num`
Input values

The input value changes whenever a user changes the input.

```
input$num = 25
```

```
input$num = 50
```

```
input$num = 75
```
Input values

The input value changes whenever a user changes the input.

Input values

Output will automatically update if you follow the 3 rules.
Reactivity 101

Reactivity automatically occurs whenever you use an input value to render an output object

```r
function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}
```
renderPlot({
  hist(rnorm(input$num))
})
renderPlot({
    hist(rnorm(input$num))
})
Recap: Server

Use the server function to assemble inputs into outputs. Follow 3 rules:

1. Save the output that you build to `output$`

```r
output$hist <-
```

2. Build the output with a `render*()` function

```r
renderPlot({
  hist(rnorm(input$num))
})
```

3. Access input values with `input$`

```r
input$num
```

Create reactivity by using *Inputs* to build rendered *Outputs*
Share your app
Every Shiny app is maintained by a computer running R
Every Shiny app is maintained by a computer running R
How to save your app

One directory with every file the app needs:

- `app.R` (*your script which ends with a call to `shinyApp()`*)
- datasets, images, css, helper scripts, etc.

You must use this exact name (`app.R`)
Two file apps

```r
library(shiny)

ui <- fluidPage(
    sliderInput(inputId = "num",
                label = "Choose a number",
                value = 25, min = 1, max = 100),
    plotOutput("hist")
)

server <- function(input, output) {
    output$hist <- renderPlot({
        hist(rnorm(input$num))
    })
}

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

# server.R

```r
library(shiny)

function(input, output) {
    output$hist <- renderPlot({
        hist(rnorm(input$num))
    })
}
```

© 2015 RStudio, Inc.

Slides at: bit.ly/shiny-quickstart-1
Two file apps

One directory with two files:

• server.R
• ui.R

You must use these exact names
Launch an app
Display options
Use shinyapps.io by RStudio
Shinyapps.io

A server maintained by RStudio

- free
- easy to use
- secure
- scalable
Getting started guide

shiny.rstudio.com/articles/shinyapps.html
<table>
<thead>
<tr>
<th>Plan</th>
<th>Price</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREE</td>
<td>$0/mo</td>
<td>5 Applications &lt;br&gt;25 Active Hours &lt;br&gt;Community Support &lt;br&gt;RStudio Branding</td>
</tr>
</tbody>
</table>
Build your own server!
A back end program that builds a linux web server specifically designed to host Shiny apps.
Shiny Server Pro

www.rstudio.com/products/shiny/shiny-server/

✅ Secure access - LDAP, GoogleAuth, SSL, and more
✅ Performance - fine tune at app and server level
✅ Management - monitor and control resource use
✅ Support - direct priority support

45 day evaluation free trial
Recap: Sharing

Save your app in its own directory as `app.R`, or `ui.R` and `server.R`

Host apps at `shinyapps.io` by:

1. Sign up for a free `shinyapps.io` account

2. Install the `shinyapps` package

Build your own server with **Shiny Server** or **Shiny Server Pro**
Learn more
You now how to

Build an app

Create interactions

Share your apps
How to start with Shiny

1. How to build a Shiny app (Today)

2. How to customize reactions (May 27)

3. How to customize appearance (June 3)
The Shiny Development Center

shiny.rstudio.com