Shiny 学习

- 说明：

1. 此中文版 Shiny 教程以 RStudio 官网教程为蓝本，原版教程的链接是：http://shiny.rstudio.com/tutorial/
2. 此翻译目的仅出于个人兴趣，并非官方的标准翻译；
3. 水平与时间皆有限，翻译教程肯定存在很多不准确之处，欢迎将指导意见与建议发送至我的邮箱（邮箱请见下方）；
4. 个人认为：Shiny 是一款很强大的 R 包，热爱 Shiny，才会进行这次翻译，希望能够有 R 语言爱好者共同努力、共同参与，将更多优秀的 R 语言文档进行翻译与推介；
5. 转载或使用此份中文版 Shiny 教程请注明出处。

- 译者：

罗晨（LUO Chen 新浪微博：@罗晨_Scorpion）
中国传媒大学 2011 级传播学（媒体市场调查与分析方向）
邮箱：luochen19941112@foxmail.com

- 校对者：

冷雪（LEN Xue 新浪微博：@冷雪_Sharon）
中国传媒大学 2011 级媒体创意

目 录

Shiny 是一个 R 语言中的网络应用程序框架，可以将你的数据分析变成互动性的网络应用（web apps），上手不需要具备 HTML、CSS 或 JavaScript 的相关知识。

谁应该学习 Shiny 教程？

如果你已掌握 R 语言编程，但还不了解 Shiny，那么你会在阅读本教程后获益匪浅。

如果你刚刚接触 R 语言，那么在学习本教程之前推荐给你的基础学习资源有 www.rstudio.com/training。如果你不确定是否准备好学习 Shiny，那么请点击链接进行测验（http://shiny.rstudio.com/tutorial/quiz/ ）。

如果你在日常工作和学习中已经使用到 Shiny，那么你可能会想要跳过本教程，访问开发中心（Development Center）的文章（Article）部分（http://shiny.rstudio.com/articles/ ）。在文章部分，有涵盖各种高水平的 Shiny 应用专题。

开始 Shiny

这 7 个课程将把你从 R 编程者带入到 Shiny 开发者的领域。每个课时大约 20 分钟，讲授一项新的 Shiny 技能。到课程结束之时，你将会知晓如何建立和部署一个属于你自己的 Shiny 应用程序。

课程清单

课程 1-欢迎来到 Shiny
课程 2-用户界面（user interface）布局
课程 3-添加控制组件
课程 4-响应式（reactive）输出
课程 5-使用 R 脚本和数据
课程 6-使用响应式（reactive）表达式
课程 7-分享你的应用（apps）

课程 1-欢迎来到 Shiny

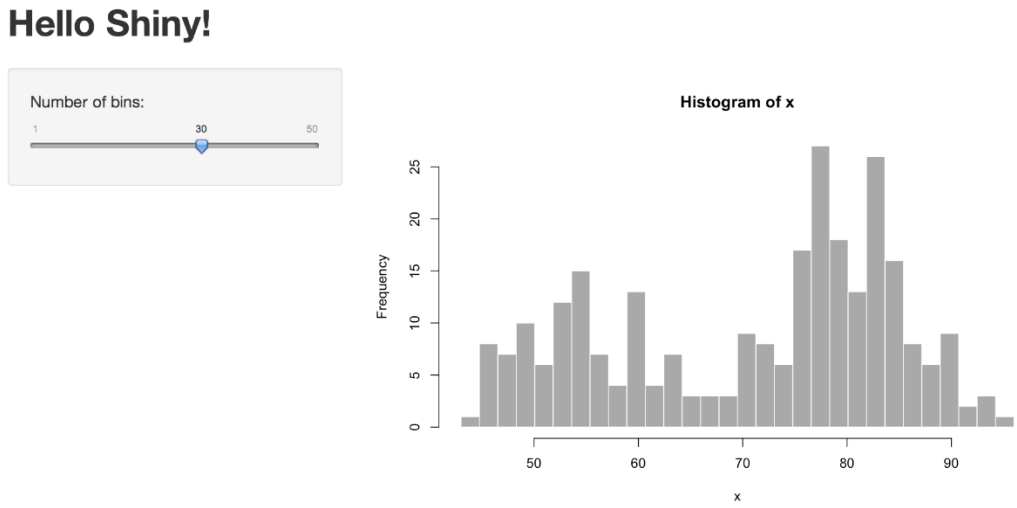
Shiny 是一个 R 包，它让通过 R 语言建立互动性网络应用（apps）变得更加简洁，本课程将立刻让你着手进行 Shiny 应用的构建。

如果你尚未安装 Shiny 包，请新建一个 R 任务，并连接网络，在控制台执行以下代码：

```
>install.packages("shiny")
```

本课程以 RStudio 集成开发环境为基础，请点击 http://www.rstudio.com/ide/download/preview 下载 RStudio。

示例



Shiny 包中内置了 11 个示例，详细地展现 Shiny 的基本特性。每一个示例都是一个独立的应用程序（self-contained Shiny app）。

Hello Shiny 示例根据 R 的 faithful 数据集绘制一个直方图，并且由可变数量的 bins 进行操控。用户可以通过滑动条更改 bins 的数量，应用会立刻对更改动作做出响应。你可以使用 Hello Shiny 来探索一个 Shiny 应用的结构并且创造属于你的第一个应用程序。

执行 Hello Shiny，键入：

```
>library(Shiny)
>runExample("01_hello")
```

Shiny 应用的结构

Shiny 应用包含两个基本的组成部分：

- 一个用户界面脚本（a user-interface script）
- 一个服务器脚本（a server script）

用户界面(ui)脚本控制应用的布局与外表。它定义在一个称作 ui.R 的源脚本中。在 Hello Shiny 的示例中，ui.R 的脚本如下：

```
library(shiny)
# Define UI for application that draws a histogram
```

```
shinyUI(fluidPage(
    #Application title
    titlePanel("Hello Shiny!"),

    #sidebar with a slide input for the number of bins
    slidebarLayout(
        slidebarPanel(
            slideInput("bins",
                       "Number of bins:",
                        min = 1,
                        max = 50,
                        value = 30)
        ),

        #Show a plot of the generated distribution
        mainPanel(
            plotOutput("distPlot")
        )
    )
))
```

Server.R 脚本包含建构应用所需要的一些重要指示，Hello Shiny 示例的 server.R 的脚本如下：

```
library(shiny)

# Define server logic required to draw a histogram
shinyServer(function(input, output) {

  # Expression that generates a histogram. The expression is
  # wrapped in a call to renderPlot to indicate that:
  #
  #  1) It is "reactive" and therefore should re-execute automatically
  #     when inputs change
  #  2) Its output type is a plot

  output$distPlot <- renderPlot({
    x    <- faithful[, 2]  # Old Faithful Geyser data
    bins <- seq(min(x), max(x), length.out = input$bins + 1)

    # draw the histogram with the specified number of bins
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

在某种程度上，Hello Shiny 中的 server.R 脚本非常简单。脚本进行了一些基础运算然后根据 bins 数量绘制直方图。

然而，你也许会注意到脚本中很大部分包裹在 renderPlot 中。以上脚本中的注释解释了一些 renderPlot 的功能，但如果你还是存有疑惑，不必担心。我们会在后续课程中详细介绍这一概念。

多操作 Hello Shiny 应用，然后反复浏览源代码。努力培养对应用程序工作流程的思维感觉。

当 Hello Shiny 应用在运行的时候，你的 R 处于繁忙状态，因此你不能执行别的 R 命令。若是希望恢复自己的 R，点击退出或者单击停止符号（停止符号在 Rstudio 控制面板的右上角）。

运行一个应用

每一个 Shiny 应用都拥有相同的结构：存放在一个目录中的两个 R 脚本。最低要求是：一个 Shiny 应用有一个 ui.R 文件和一个 server.R 文件。

你可以在一个目录中保存一个 ui.R 文件和 server.R 文件来创建一个 Shiny 应用。每一个应用都需要自己独特的存放目录。

运行应用的方法是在函数 runApp 中置入目录名称。例如你的应用存放的目录名称为 my_app，那么键入以下代码可以执行应用：

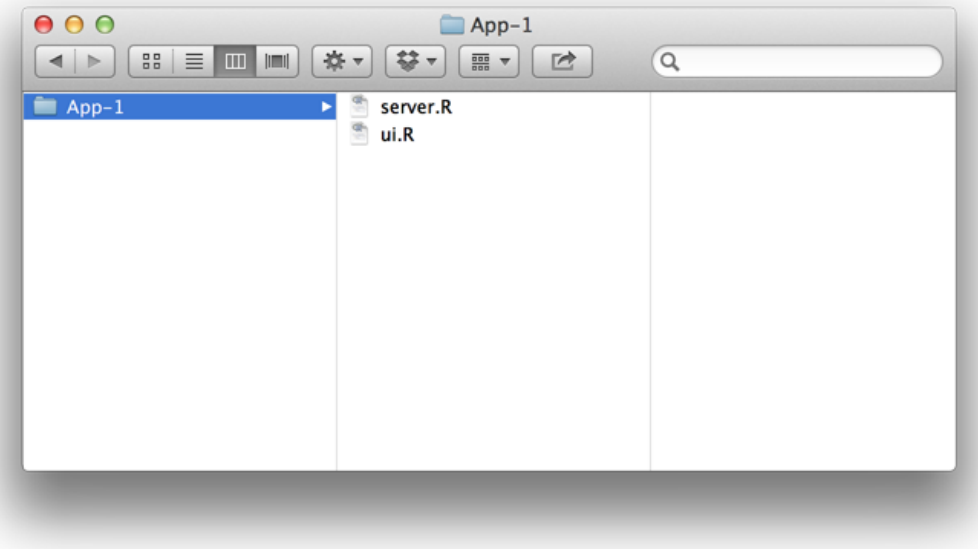
```
>library(shiny)
>runApp("my_app")
```

注意：runApp 与 read.csv、read.table 以及 R 中其他的一些命令类似。runApp 中的第一个参数是从你的工作目录到应用存放目录的一个文件路径。以上的代码假设 my_app 是一个相对路径。在这一情况下，文件路径与目录名称相同。

（以免你会疑惑，Hello Shiny 应用中的文件存放在一个特殊的系统目录"01_hello"中。这个目录可以直接通过 runExample ("01_hello")来予以调用。）

尝试

在你的工作目录下创建一个名为"App-1"的目录。然后复制粘贴 Hello Shiny 中两个脚本的代码到自己的文件目录中。当你完成这一步操作后，你的目录状态应该如下：



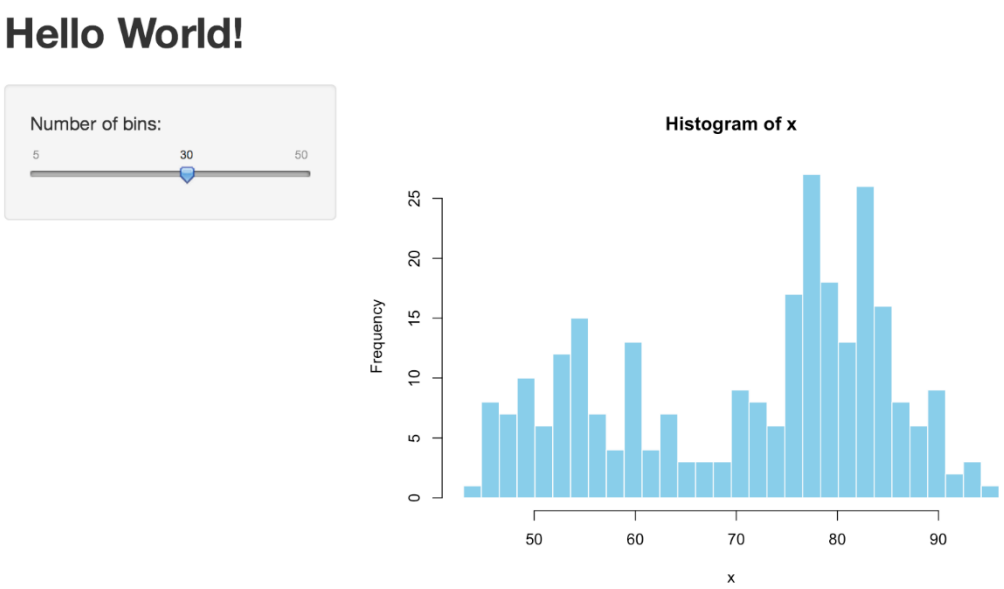
通过输入"runApp("App-1")"来运行你的应用，然后点击退出并对自己的应用作如下更改：

将标题从"Hello Shiny!"改为"Hello World!"。

将滚动条的最小值设置为 5.

将直方图颜色由"darkgray"改为"skyblue"。

当你完成后，再次运行你的应用。你的新应用显示应该如下图所示。



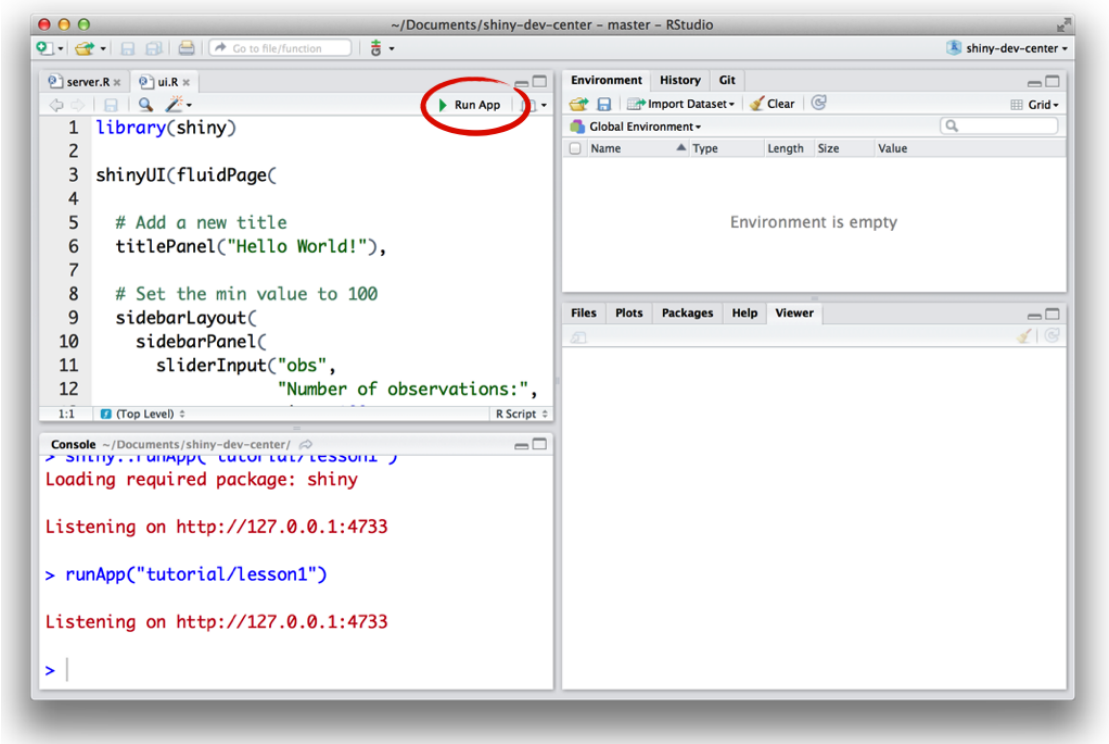
默认情况下，Shiny 应用采用"normal"模式显示，恰如以上应用截图所示。Hello Shiny 和其他内置的例子的展示都采用"showcase model"，这是一种不同的展示模式。

如果你希望自己的应用也以 showcase 模式进行展示，可以执行如下命令：runApp("App-1",display.mode = "Showcase").

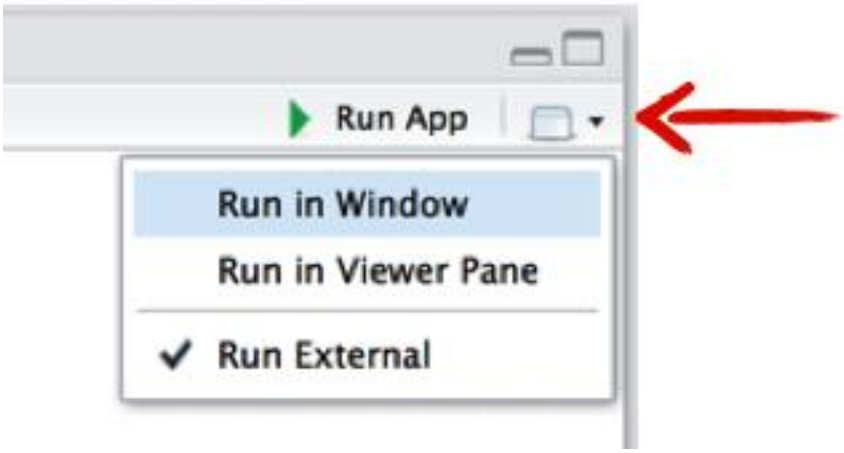
重新运行应用

重新运行运用的方法有两种：

- 一种是执行 runApp("App-1")命令，另外一种是
- 在 Rstudio 编辑器中打开 ui.R 或者 server.R 脚本，Rstudio 会自动识别 Shiny 脚本并且提供一个 Run App 的按钮（在编辑器右上角）。你既可以点击按钮进行执行操作，也可以使用键盘快捷键：Command+Shift+Enter on iOS，Control+Shift+Enter on Windows.



默认情况下，Rstudio 会在新窗口中运行应用，但是你也可以对展示窗口进行设置，在上图 Run App 的按钮右边，可以进行选择，一共有三种展现形式。



扼要重述

创建自己的 Shiny 应用，你需要：

为自己的应用建立一个目录；

在你的目录下保存 server.R 和 ui.R 脚本；

通过 runApp 或者快捷键来运行应用；

通过点击 escape 退出 Shiny 应用。

内容拓展

你可以通过复制和更改现有的 Shiny 应用来创建自己的 Shiny 应用。Shiny Gallery（http://shiny.rstudio.com/gallery ）提供很多好例子，或者可以使用下列 11 种预建的 Shiny 示例。

```
system.file("examples", package="shiny")
```

```
runExample("01_hello") # a histogram
runExample("02_text") # tables and data frames
runExample("03_reactivity") # a reactive expression
runExample("04_mpg") # global variables
runExample("05_sliders") # slider bars
runExample("06_tabsets") # tabbed panels
runExample("07_widgets") # help text and submit buttons
runExample("08_html") # Shiny app built from HTML
runExample("09_upload") # file upload wizard
runExample("10_download") # file download wizard
runExample("11_timer") # an automated timer
```

课程 2-构建一个用户界面

现在你已经明白了 Shiny 应用的基本结构，你可以着手构建你的第一个应用了。

本课程将向你展示如何为自己的应用构建一个用户界面。你将学习到用户界面如何布局，然后如何添加文字、图片、以及其他的 HTML 元素到应用中。

我们将利用到你在课程 1 中建构的应用 App-1。开始学习之前，打开它的 server.R 和 ui.R 文件。像如下代码一样编辑自己的脚本。

Ui.R

```
ShinyUI(fluidPage(
))
```

Server.R

```
shinyServer(function(input,output){
})
```

这些代码是创建一个 Shiny 应用的最低要求。以上代码的运行结果是一个有空白用户界面的空应用，作为本课时的开始最合适不过。

布局

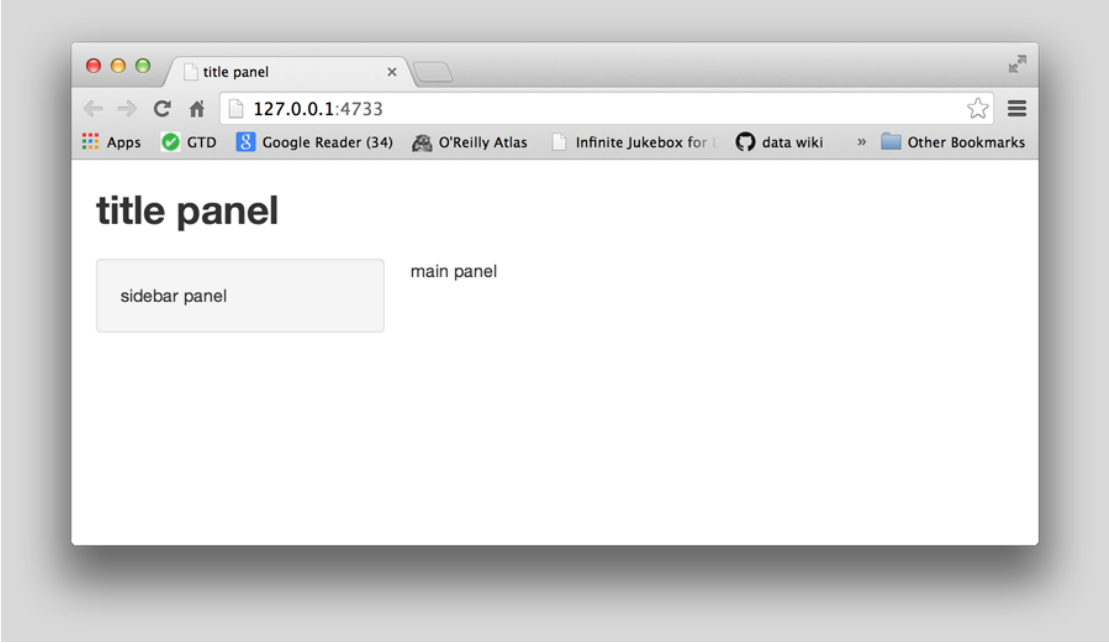
Shiny 的 ui.R 脚本利用函数 fluidPage 来创造一个自适应用户浏览器窗口的展示。应用的布局与展示一定要添加 fluidPage 函数。

例如，以下的 ui.R 脚本创造这样的一个用户界面：包含一个标题面板和一个侧边栏布局，后者包括一个侧边栏面板和一个主面板。这些元素都放置在 fluidPage 函数内部。

```
# ui.R

shinyUI(fluidPage(
  titlePanel("title panel"),

  sidebarLayout(
    sidebarPanel( "sidebar panel"),
    mainPanel("main panel")
  )
))
```



titlePanel 和 sidebarLayout 是 fliudPage 中最重要的两个元素。他们协同创造出了基本的带有侧边栏的 Shiny 应用。

sidebarLayout 一般携带两个参数：

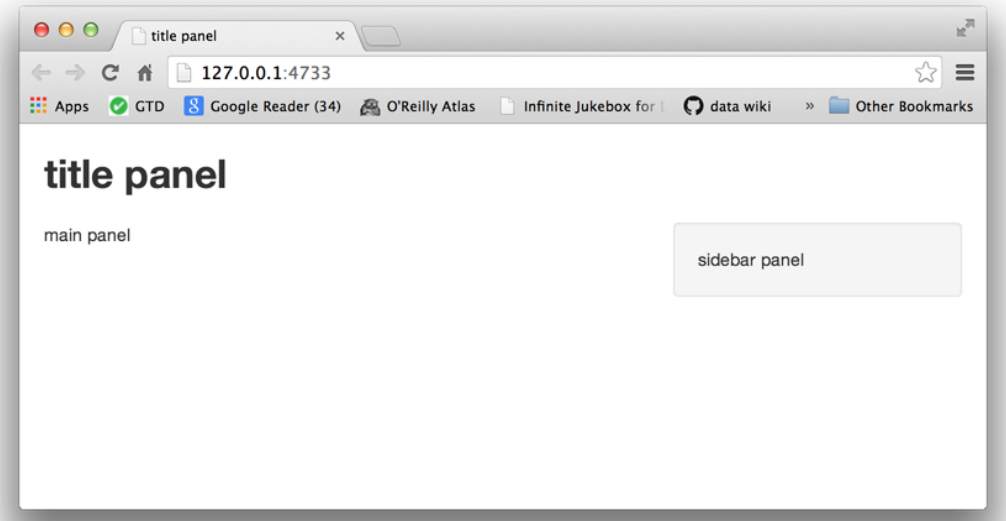
- sidebarPanel
- mainPanel

这些函数保证你可以在侧边栏或者是主面板添加内容。默认情况下，侧边栏面板出现在应用的左侧。你可以将其移动到右侧，具体做法是在 sidebarLayout 中输入选择参数 position = "right".

```
# ui.R

shinyUI(fluidPage(
  titlePanel("title panel"),

  sidebarLayout(position = "right",
    sidebarPanel( "sidebar panel"),
    mainPanel("main panel")
  )
))
```



titlePanel 和 sidebarLayout 为你的 Shiny 应用构成基本的布局，但你可以创建更高级的布局。例如，你可以利用 navbarPage 来让你的应用展示出导航栏，实现多页面的用户界面。或者你可以利用 fluidRow 和 column 来构建布局网格系统。如果你想要更多地了解高级的设置，请阅读 Shiny Application Layout Guide（http://shiny.rstudio.com/articles/layout-guide.html ）。在这一课程中，我们将继续讲解 sidebarLayout。

HTML 内容

你可以在 Shiny 应用中添加内容，做法仅仅是在＊Panel 函数中添加相关元素。例如 sidebarPanel("sidebar panel")，"sidebar panel"就显示在侧边栏面板中。原理同样适用于标题面板和主面板。

想要添加更多的内容，使用 Shiny HTML 标签中的任何一个函数。这些函数与常见的 HTML5 标签大同小异，我们可以尝试一下。

Shiny 函数与 HTML5 的等价列举
p 〈p〉 一段文字
h1 〈h1〉 级别 1 的标题
h2 〈h2〉 级别 2 的标题
h3 〈h3〉 级别 3 的标题
h4 〈h4〉 级别 4 的标题
h5 〈h5〉 级别 5 的标题
h6 〈h6〉 级别 6 的标题
a 〈a〉 一个超链接
br 〈br〉 一个换行符（如：一个空行）
div 〈div〉 具有某种统一风格的文本的区域
span 〈span〉 具有某种统一风格的文本的行内区域
pre 〈pre〉 具有固定宽度的字体
code 〈code〉 一个格式化的代码块

img 〈img〉 一张图片
strong 〈strong〉 粗体字
em 〈em〉 斜体字
HTML 直接以 HTML 代码样式通过的字符串

标题

建立一个标题元素：

- 选择一个标题函数（例如 h1 或者 h5）
- 输入你想在标题中希望看到的文本

例如，你可以创建第一级别的标题，利用 h1(“My title”)，显示字符为“My title”。如果你在命令行中输入以下命令，你会注意到实际上 Shiny 中的函数产生了 HTML 代码。

```
>library(shiny)
>h1(“My title”)
<h1>My title</h1>
```

如何将这一元素放入应用中：

将 h1("My title")作为一个变量放入 titlePanel，sidebarPanel 或者 mainPanel 中。

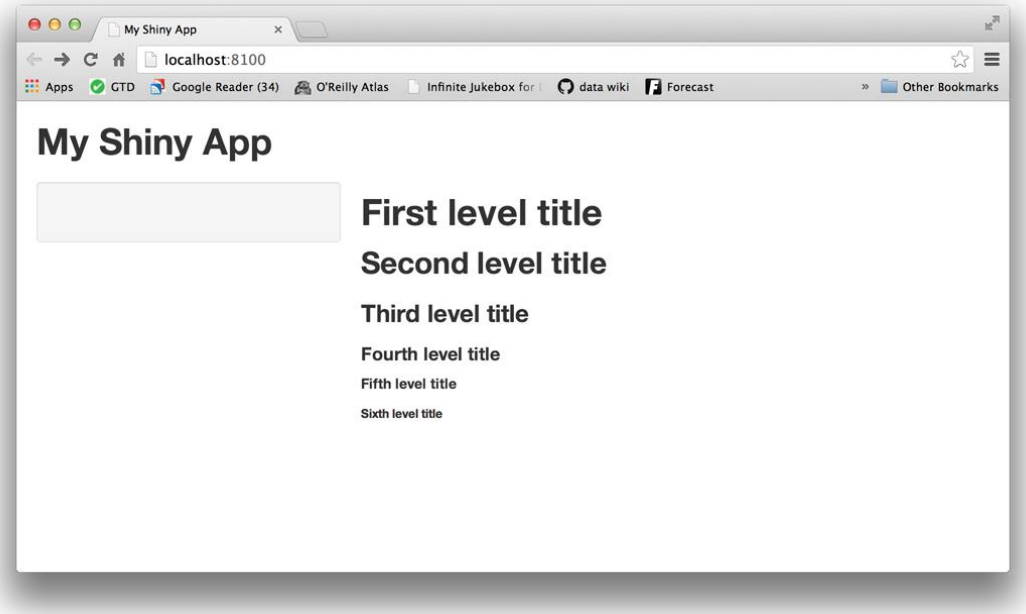
这个文本会显示在相应的面板中。你可以在同一个面板中放入多个元素，前提是用逗号区分它们。

尝试一下。下面的新脚本包含六个级别的标题。升级你的 ui.R 脚本然后重新运行你的应用。（执行 runApp（“App-1”）或者利用上一课程中介绍的按钮与快捷键）

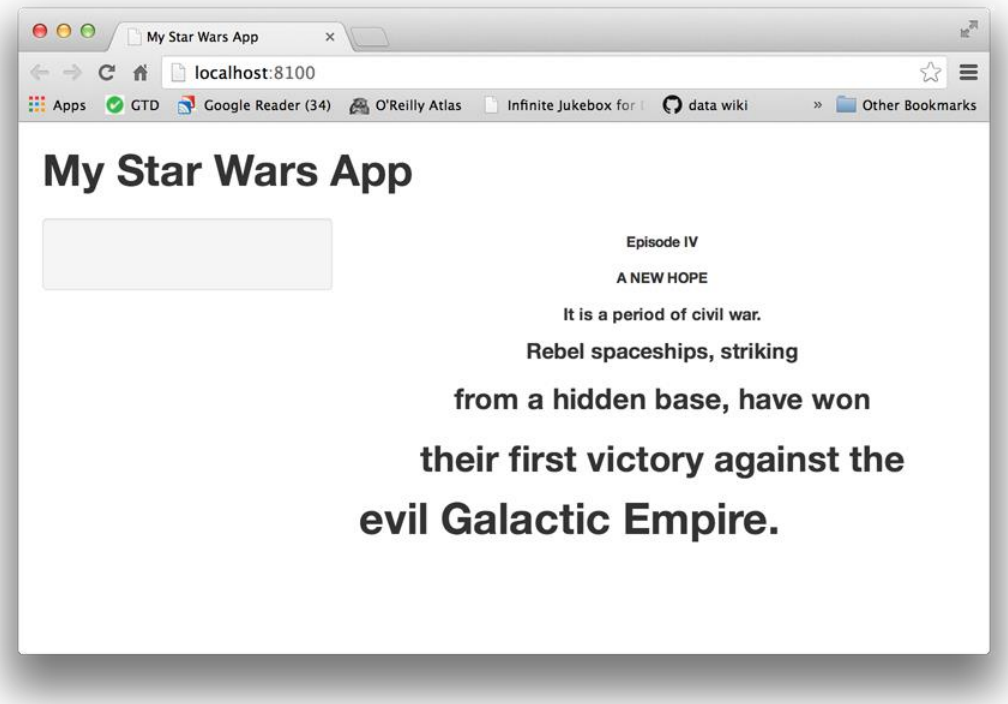
```
# ui.R

shinyUI(fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      h1("First level title"),
      h2("Second level title"),
      h3("Third level title"),
      h4("Fourth level title"),
      h5("Fifth level title"),
      h6("Sixth level title")
    )
  )
))
```

现在你的应用应如下图所示。



如果是 George Lucas（译者注：George Lucas 是著名电影《星球大战》的导演）制作的应用，将会像下图一样。



为了达到这个效果，你可以使用 align = "center"，就如 h6("Episode IV",align = "center")。

通常来说，任何 HTML 标签都可以作为参数添加到 Shiny 的标签函数中。

如果你对于 HTML 标签的种种属性较为陌生，你可以通过一些免费的在线 HTML 资源学习，例如 w3schools(http://www.w3schools.com/tags/tag_hn.asp)。

以下是《星球大战》用户界面的代码：

```
# ui.R

shinyUI(fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
        h6("Episode IV", align = "center"),
        h6("A NEW HOPE", align = "center"),
        h5("It is a period of civil war.", align = "center"),
        h4("Rebel spaceships, striking", align = "center"),
        h3("from a hidden base, have won", align = "center"),
        h2("their first victory against the", align = "center"),
        h1("evil Galactic Empire.")
    )
  )
))
```

对文本设置格式

Shiny 为文本格式设置提供了许多标签式函数。了解它们的最简单方法是通过案例实践。

粘贴以下的ui.R脚本到你自己的ui.R脚本中然后加以保存。如果你的Shiny应用还在运行，你可以直接刷新你的网页或者预览窗口，即可显示新变化。如果应用已关闭，请再次运行。

可以通过比较用户界面的变化来掌握怎样在 Shiny 应用中对文本设置格式。

```
# ui.R

shinyUI(fluidPage(
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      p("p creates a paragraph of text. Note: this paragraph is followed by br(), which makes a blank line."),
      p("A new p() command starts a new paragraph. Supply a style attribute to change the format of the entire paragraph", style = "font-family: 'times'; font-si16pt"),
      strong("strong() makes bold text."),
      em("em() creates italicized (i.e, emphasized) text."),
      br(),
      code("code displays your text similar to computer code"),
      div("div creates segments of text with a similar style. This division of text is all blue because I passed the argument 'style = color:blue' to div", style = "color:blue"),
```

```
      br(),
      p("span does the same thing as div, but it works with",
        span("groups of words", style = "color:blue"),
        "that appear inside a paragraph.")
    )
  )
))
```

图像

图像可以提升你的应用显示的美观程度，并且帮助你的用户更好地理解内容。Shiny 利用 img 函数来在应用中添加图像。

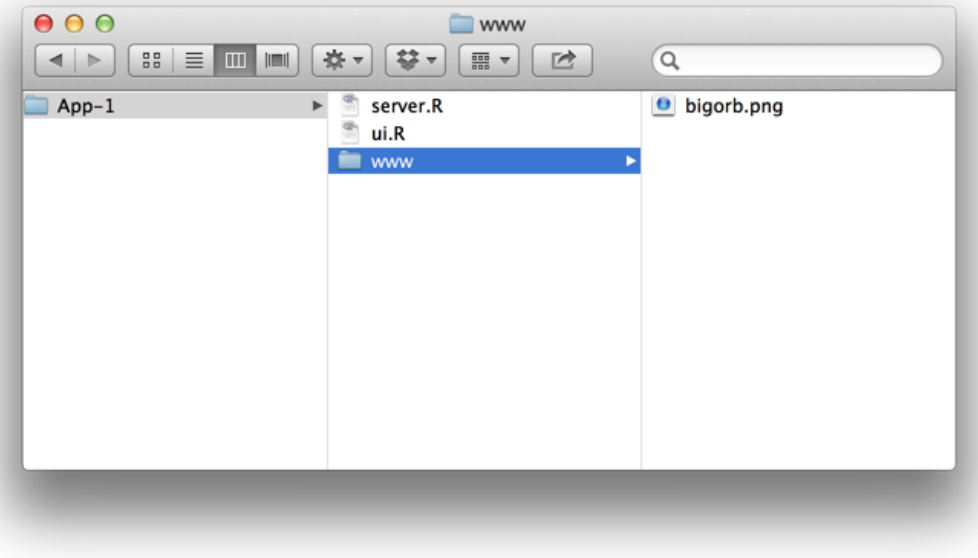
插入一个图像的具体做法是，给出 img 函数，再在 src 参数中输入图像文件的名称(例如，img(src = "my_image.png"))。

你还可以添加其他 HTML 中的参数，例如高（height）和宽（width）等。注意宽、高的数值都是指像素值。

```
img(src = "my_image.png",height = 72, width = 72)
```

img 函数将在特定的地方寻找你的图像文件。你的文件必须存放在与 ui.R 脚本相同目录的路径中，并且文件夹要以 www 命名。Shiny 将会在网络浏览器中分享 www 文件夹里的所有文件，www 文件夹是一个存放图片、体例等元素的理想位置，你构建 Shiny 应用的所有元素都可以包含于内。

*所以如果你想使用名为"bigorb.png"的图片，（http://shiny.rstudio.com/tutorial/lesson2/www/bigorb.png） 你的 App-1 目录应该如下（iOS 系统中）：



利用这种文件布置，以下的 ui.R 脚本可以创造这个应用。下载 bigorb.png 并且尝试。

```
# ui.R

shinyUI(fluidPage(
```

```
  titlePanel("My Shiny App"),
  sidebarLayout(
    sidebarPanel(),
    mainPanel(
      img(src="bigorb.png", height = 400, width = 400)
    )
  )
))
```

其它标签

本次课程中涵盖最流行的 Shiny 标签函数，但仍有很多其他的函数等待你的使用。你可以通过这两个链接了解其他的标签函数。Customize your UI with HTML（http://shiny.rstudio.com/articles/html-tags.html ）和 Shiny HTML Tags Glossary（http://shiny.rstudio.com/articles/tag-glossary.html）。

扼要重述

伴随这些你已掌握的新技能，你可以：

- 创建一个带有 fluidPage、titlePanel 和 sidebarLayout 的用户界面
- 依靠 Shiny 的标签函数创造 HTML 元素
- 根据每个标签函数的参数来设置 HTML 的标签属性
- 将你需要添加的元素放入 titlePanel、sidebarLayout 或者是 mainPanel 中
- 在任何一个面板中添加多种元素，利用逗号区隔它们
- 将你的图像放在 Shiny 应用文件目录下的 www 文件夹中，并在应用中利用 img 函数来添加图像

注意

当你在写 code(‘install.packages("shiny")’),
请严格注意双引号与单引号的区别与兼容性，绝对不能出现双引号中包含双

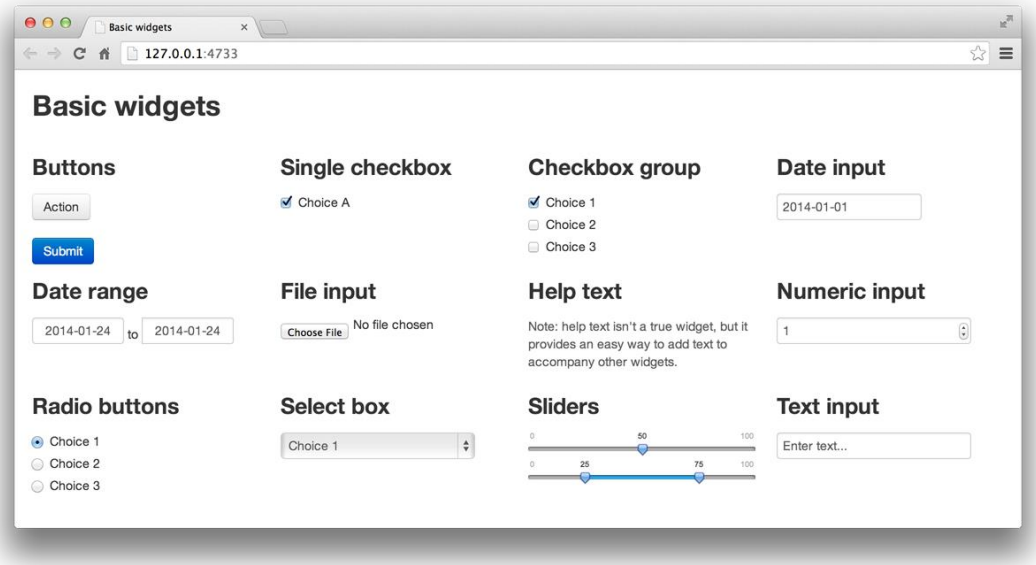
引号。

课程 3-添加控制部件（widgets）

本课程将向你展示如何在 Shiny 应用中添加控制部件。什么是部件？就是一个用户可以进行交互的网络元素。部件为你的用户提供了一种能够向应用发送信息指令的途径，

Shiny 的部件收集用户输入的一系列数值。当一名用户更改了部件，用户界面上的数值也会相应进行更改。我们将会在课程 4 中对这一点进行详细讨论。

控制部件



Shiny 与生俱来的特征就是自带内置部件，每个部件都有一个 R 函数与之对应。例如，Shiny 提供的一个函数名曰 actionButton，这个函数可以创建一个动作按钮、一个函数名为"sliderInput"，它用来创建滑动条。

标准的 Shiny 部件为：

函数 部件
actionButton 动作按钮
checkboxGroupInput 一个复选框（check boxes）系列
checkboxInput 一个单一复选框
dateInput 一个日期选择控件
dateRangeInput 两个日历选择器，用以确定一个时间范围
fileInput 文件上传的控件向导
helpText 帮助文本
numericInput 键入数值的区域
radioButtons 一组单选按钮
selectInput 一个可供选择的待选盒
sliderInput 滑动条
submitButton 一个呈交按钮
textInput 输入文本的区域

这些部件来自于 Twitter Bootstrap（http://getbootstrap.com/2.3.2/ ）项目，这是一个风靡世界的用于搭建用户界面的开源架构。

添加部件

正如在课程 2 中添加 HTML 内容一致，你也可以用这种方法向你的网页添加部件。应用中的部件可以添加在 ui.R 文件中的 sidebarPanel 或 mainPanel。

每一个部件都需要几个参数。每一个部件的前两个参数是：

- **部件的名称（name for the widget）**.用户不会看见部件名称，但是你可以利用部件名称来接触部件的数值（译者注：这一点在 server.R 中会用到）。部件名称必须是字符串类型。
- **标签（label）**.标签会与部件一起显示在应用里。它也必须是一个字符串，但是可以为空字符串。

在本例中，部件名称为 action，标签为 Action：

```
actionButton("action",label = "Action")
```

其余的参数随部件的不同而各有变化，这完全取决于部件实现功能的自身需求。参数包括部件实现功能所必需的起始值、定义域和增量等。你可以在部件帮助页面找到具体的参数要求。例如：

```
?selectInput
```

以下的 ui.R 脚本构造了上一张图片所示的景况。把你的 App-1 中的 ui.R 脚本修改一下达到图片所示的效果，然后执行 runApp("App-1").

尝试每一个部件寻找感觉，了解它们的功能及运行机制。修改部件函数的数值并观察效果变化。如果你对 Shiny 应用的布局计划该兴趣，敬请阅览 application layout guide（http://shiny.rstudio.com/articles/layout-guide.html ）中的详细描述。这些内容在本课程中将不一一赘述。

```
# ui.R

shinyUI(fluidPage(
  titlePanel("Basic widgets"),

  fluidRow(

    column(3,
      h3("Buttons"),
      actionButton("action", label = "Action"),
      br(),
      br(),
      submitButton("Submit")),

    column(3,
      h3("Single checkbox"),
      checkboxInput("checkbox", label = "Choice A", value = TRUE)),

    column(3,
      checkboxGroupInput("checkGroup",
```

```
      label = h3("Checkbox group"),
      choices = list("Choice 1" = 1,
        "Choice 2" = 2, "Choice 3" = 3),
      selected = 1)),

  column(3,
    dateInput("date",
      label = h3("Date input"),
      value = "2014-01-01"))
),

fluidRow(

  column(3,
    dateRangeInput("dates", label = h3("Date range"))),

  column(3,
    fileInput("file", label = h3("File input"))),

  column(3,
    h3("Help text"),
    helpText("Note: help text isn't a true widget,",
      "but it provides an easy way to add text to",
      "accompany other widgets.")),

  column(3,
    numericInput("num",
      label = h3("Numeric input"),
      value = 1))
),

fluidRow(

  column(3,
    radioButtons("radio", label = h3("Radio buttons"),
      choices = list("Choice 1" = 1, "Choice 2" = 2,
                     "Choice 3" = 3),selected = 1)),

  column(3,
    selectInput("select", label = h3("Select box"),
      choices = list("Choice 1" = 1, "Choice 2" = 2,
                     "Choice 3" = 3), selected = 1)),

  column(3,
```

```
      sliderInput("slider1", label = h3("Sliders"),
        min = 0, max = 100, value = 50),
      sliderInput("slider2", "",
        min = 0, max = 100, value = c(25, 75))
      ),

    column(3,
      textInput("text", label = h3("Text input"),
        value = "Enter text..."))
  )

))
```

扼要重述

在你的应用中添加功能强大的各种部件是非常容易的。

- Shiny 提供了一系列函数来创建这些部件
- 每一个函数都需要一个名称（name）以及标签（label）
- 部分部件的功能实现需要特殊指令，需要详细阅读相关说明文档
- 往应用中加入部件的操作方法恰如你往应用中添加各种各样的 HTML 内容(可参见课程 2)

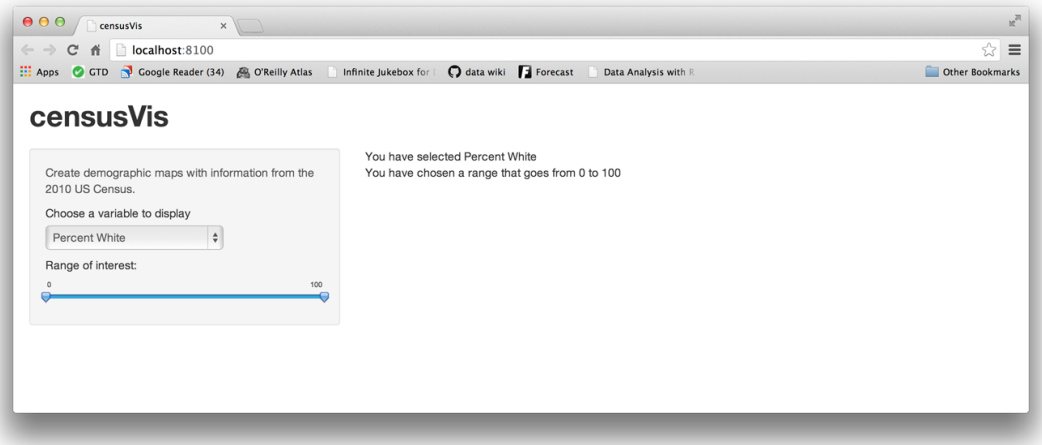
内容拓展

*Shiny Widgets Gallery（http://shiny.rstudio.com/gallery/widget-gallery.html ）提供各种部件的样板，只需点击每个部件下面的"See Code"即可查看源代码。操作简单。

课程 4-响应式输出

是时候让你的 Shiny 应用动起来了！本课程将教授你如何在你的应用中建立响应输出。当你的用户触发部件时，响应输出会自动做出触发回应。

到本课程结束之时，你将会明白如何做出一个携带两个响应式文本行的简单 Shiny 应用。每个文本行都会显示用户输入的相应部件数值。



这个新的 Shiny 应用需要自己独立的、全新的目录。在你的工作目录中新建一个文件夹，名称为"census-app"。这个文件夹将会是你存放 ui.R 和 server.R 的地方。

两个步骤

你可以通过两个步骤来创建响应输出。

1. 在 ui.R 脚本中，在你的用户界面中添加一个 R 对象（R object）
2. 告诉 Shiny 如何在 server.R 中与 ui.R 的对象进行关联。如果它的相关代码调用了部件数值，那么对象将会作出响应。

步骤 1：向 UI 中加入 R 对象

Shiny 提供了一系列函数，用以在用户界面中将 R 对象转化为输出。每一个函数都创造一个特定类型的输出。

输出函数	创造结果
htmlOutput	HTML 代码
imageOutput	图像
plotOutput	图表
tableOutput	表格
textOutput	文本
uiOutput	原始的 HTML
verbatimTextOutput（verbatim：逐字的）	文本

正如你向用户界面添加 HTML 元素和部件一样，你可以用类似做法向 UI 中添加输出。将输出函数置入 ui.R 脚本的 sidebarPanel 或者 mainPanel 中。、

例如，以下的 ui.R 文件使用 textOutput 来向主要面板中添加响应文本行，最终效果如上一张图片所示。

```
# ui.R

shinyUI(fluidPage(
  titlePanel("censusVis"),

  sidebarLayout(
    sidebarPanel(
      helpText("Create demographic maps with
        information from the 2010 US Census."),

      selectInput("var",
        label = "Choose a variable to display",
        choices = c("Percent White", "Percent Black",
          "Percent Hispanic", "Percent Asian"),
        selected = "Percent White"),

      sliderInput("range",
        label = "Range of interest:",
        min = 0, max = 100, value = c(0, 100))
    ),

    mainPanel(
      textOutput("text1")
    )
  )
))
```

请注意 textOutput 利用了一个参数，也就是本例中的字符串"text1"。每一个＊Output 函数都需要一个单一参数：一个字符串，Shiny 会将其作为相应元素的名称。用户并不会看见这个名称，但你在之后的 server.R 中将会使用到它。

步骤 2：提供 R 代码来建立对象

在 ui.R 中加入函数告诉 Shiny 你应该在何处展示自己的对象。接下来，你要告诉 Shiny 如何来搭建这个对象。

这一步的做法是在 server.R 中加入建构对象的 R 代码。在你的 server.R 脚本中的 shinyServer 中代码应该是以未命名函数的形式完成的。

在 Shiny 的运行过程中，未命名函数起特别的作用；它建立一个名称为 output 的 list-like 对象，对象中包含所有更新你应用中 R 对象所需的代码。每一个 R 对象有对应于自己的条目列表。

您可以通过定义一个新元素为未命名函数中的 output 创建一个条目，就像以下的代码一般。元素名称要与你在 ui.R 中创造的相应元素名称一致。

以下脚本中，output$text1 与 ui.R 脚本中的 textOutput("text1")是相互契合的。

```
# server.R
```

```
shinyServer(function(input, output) {

    output$text1 <- renderText({
        "You have selected this"
    })

  }
)
```

你无需在最后一行代码中为匿名函数进行安排让它返回到 output。R 会自动根据参考类语义来更新 output。

每个 output 中的条目应该包含任一个 Shiny 中 render＊函数中的输出。这些函数捕获了一个 R 表达式并对表达式做一些预处理。使用 render＊函数可以与你正在制作的响应式对象类型契合。

render 函数	创造结果
renderImage	图像（保存为与源文件的链接）
renderPlot	图表
renderPrint	任何打印输出
renderTable	数据框、矩阵以及其他表格结构
renderText	字符串
renderUI	Shiny 标签对象或者是 HTML

每一个 render*函数携带一个单独的参数：一个由大括号括起来的 R 表达式，{}.这个表达式可以是一个单一的文本行，或者可以包含多行代码，就好像它是一个复杂函数的调用。

将 R 表达式想象为一系列你给予 Shiny 的指令。Shiny 将会在你第一次运行应用时运行这些指令，在未来每一次需要更新对象时 Shiny 都会重复运行这些指令。

为此，你的表达式应该返回到你预先在脑海中设定好的对象（一段文本、一个图表、一个数据框等）。如果表达式未返回到某一个的对象或者返回到错误的对象类型，你就会得到 R 的报错（error report）。

使用部件数值

当你运行以上的 server.R 脚本，Shiny 应用就会在主面板中显示"You have selected this"。然而，文本却非响应式的。当你对应用中的部件进行操纵时，这些文字并不会随之变化。

要想使之变化，我们来看看如何加以实现。

看一看 server.R 中的第一行代码。你是否注意到未命名函数提到两个参数：input 与 output？你已经看到在你的应用中为建立 R 对象，output 作为 list-like 对象存储了相应的指令。

Input 是第二个 list-like 对象。它存储了应用中所有部件的当前值。这些值将存放在 ui.R 的部件名称下。

例如，我们的应用有两个部件，一个名称为"var"另一个名称为"range"（详见课程 3 中的链接）。"var"和"range"的数值将以 input$var 与 input$range 的形式存储在 input 之中。因为滑块部件有两个数值（一个最小值和一个最大值），input$range 也同样是包含两个数值的向量。

如果对象使用 input 数值，那么 Shiny 会自动制造一个响应对象。例如，server.R 文件通过调用下拉选框部件的数值创造一个响应文本行。

```
# server.R

shinyServer(
  function(input, output) {

    output$text1 <- renderText({
      paste("You have selected", input$var)
    })

  }
)
```

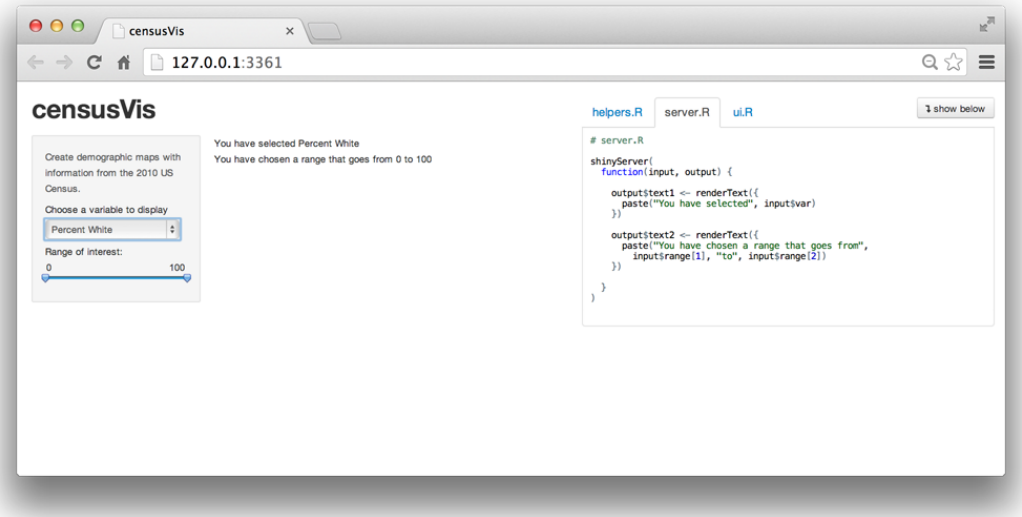
Shiny 根据不同部件进行不同的输出。当用户更改一个部件数值，Shiny 会依据部件更改重建所有输出，恰如依据定义域的变化而引发值域变化。最终，重建的对象会完全更新。

这就是 Shiny 中创建响应的步骤，即通过把 input 中的数值与 output 中的对象连接起来。

运行你的应用并且观看响应式输出

把你的 server.R 和 ui.R 更新成以上形式，然后运行你的应用，代码是 runApp("censusVis",display.mode = "showcase")，你的应用外观将如下图所示，当你的下拉选框部件数值更改后其余部分也会立刻更新。

观察一下 server.R 脚本。当 Shiny 重建一个输出时，相应的代码部分会在运行时高亮。临时的高亮显示会帮助你更好地了解 Shiny 生成响应式输出的运行原理。



扼要重述

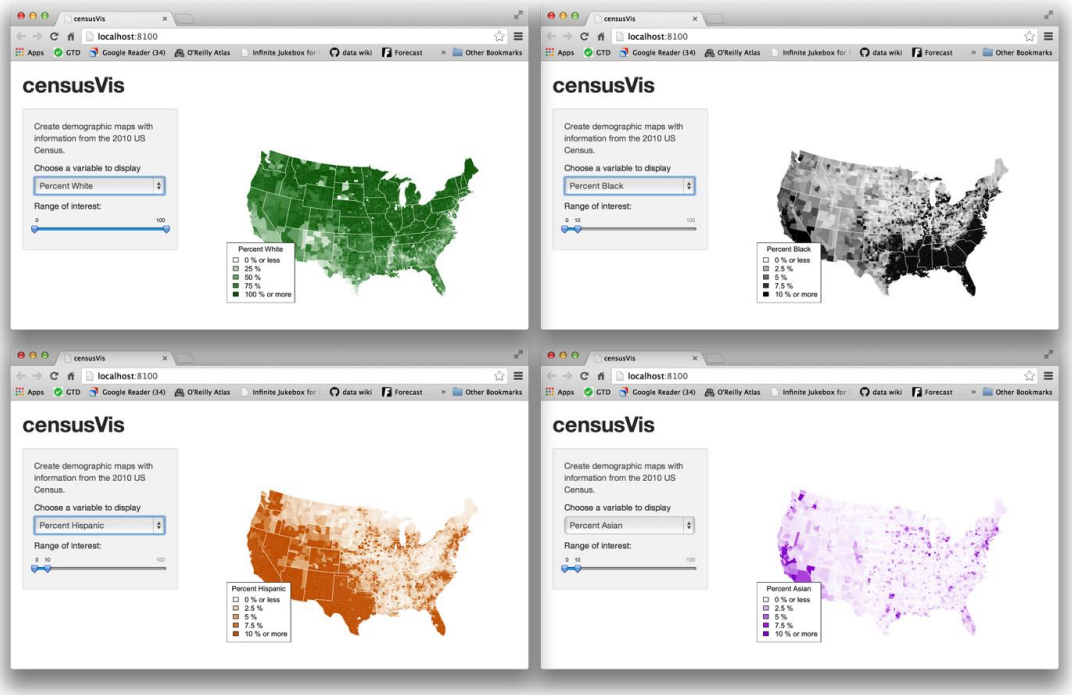
本课程中，你可以创建自己首个响应式 Shiny 应用。

- 在 ui.R 脚本中使用＊Output 函数来在 Shiny 应用中置入响应式对象

- 在 server.R 脚本中使用 render＊函数来告诉 Shiny 如何搭建你的对象
- 在每个 render＊函数中用大括号{}包容 R 表达式
- 在 output 中保存你的 render＊表达式，应用中的条目与每个响应式对象一一对应
- 在 render＊表达式中通过包含 input 数值来实现响应

课程 5-使用 R 脚本与数据

本课程将告诉你如何加载 Shiny 应用中所需的数据、R 脚本与 R 包。同时，你将会构建一个复杂的、对美国人口普查数据进行可视化的应用。



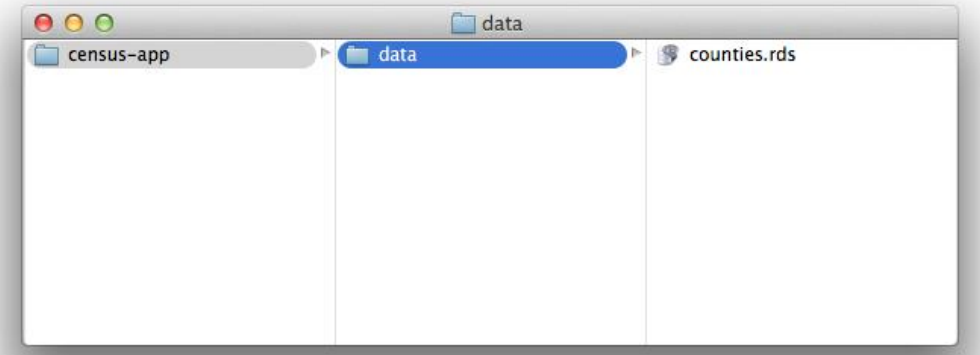
conuntries.rds

conuntries.rds 是一个包含美国各郡（county）人口统计资料的数据集，收容在 Uscensus2010 的 R 包之中。你可以从此处（http://shiny.rstudio.com/tutorial/lesson5/census-app/data/counties.rds ）下载它。

当你下载完毕后，

- 在你的 census-app 目录下创建一个名为 data 的新文件夹。
- 将 countries.rds 挪至 data 文件夹中。

当你完成这两步后，你的 census-app 文件夹将会如下图所示。



countries.rds 数据集中包含：

- 美国每个郡的名称
- 每个郡的总人口数
- 各郡中白种人、黑种人、西班牙裔人、亚洲人占总人口的比例

```
counties <- readRDS("census-app/data/counties.rds")
head(counties)
```

	name	total.pop	white	black	hispanic	asian
1	alabama,autauga	54571	77.2	19.3	2.4	0.9
2	alabama,baldwin	182265	83.5	10.9	4.4	0.7
3	alabama,barbour	27457	46.8	47.8	5.1	0.4
4	alabama,bibb	22915	75.0	22.9	1.8	0.1
5	alabama,blount	57322	88.9	2.5	8.1	0.2
6	alabama,bullock	10914	21.9	71.0	7.1	0.2

3082 observations of 6 variables

	row.names	name	total.pop	white	black	hispanic	asian
1	1	alabama,autauga	54571	77.2	19.3	2.4	0.9
2	2	alabama,baldwin	182265	83.5	10.9	4.4	0.7
3	3	alabama,barbour	27457	46.8	47.8	5.1	0.4
4	4	alabama,bibb	22915	75.0	22.9	1.8	0.1
5	5	alabama,blount	57322	88.9	2.5	8.1	0.2
6	6	alabama,bullock	10914	21.9	71.0	7.1	0.2
7	7	alabama,butler	20947	54.1	44.2	0.9	0.8
8	8	alabama,calhoun	118572	73.6	22.2	3.3	0.7
9	9	alabama,chambers	34215	58.1	39.9	1.6	0.5
10	10	alabama,cherokee	25989	92.1	6.1	1.2	0.2
11	11	alabama,chilton	43643	81.1	10.9	7.8	0.3
12	12	alabama,choctaw	13859	55.6	43.8	0.5	0.1
13	13	alabama,clarke	25833	54.0	44.6	1.0	0.3
14	14	alabama,clay	13932	80.3	16.5	2.9	0.2

Displayed 1000 rows of 3082 (2082 omitted)

helpers.R

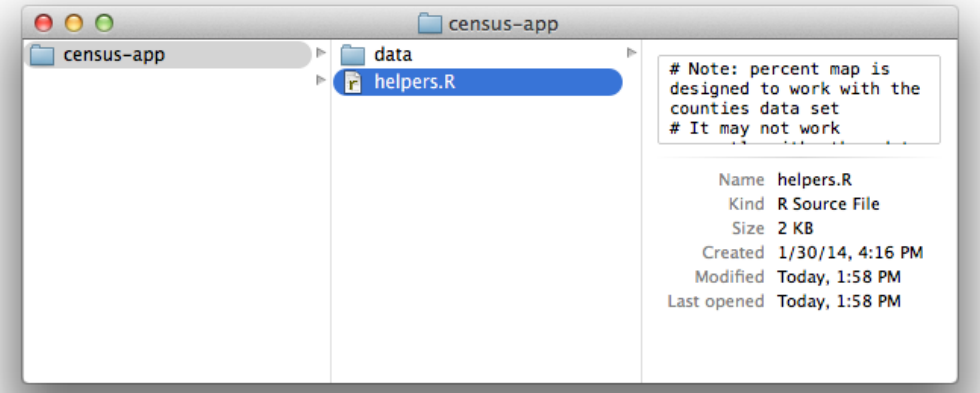
helper.R 是一个 R 脚本，可帮助你制作等值线地图，比如本课程中第一张图所示。一张等值线地图是使用颜色来显示变量的地区差别的一种地图类型。在本案例中，将会运用 helpers.R 中的 percent_map，该函数将对 counties.rds 的数据*进行地图可视化。你可以在此

（http://shiny.rstudio.com/tutorial/lesson5/census-app/helpers.R ）下载 helpers.R

helper.R 使用 maps 和 mapproj 两个 R 包。如果你之前从未安装过这两个 R 包，在你制作应用之前需要执行以下代码：

```
>install.packages(c("maps","mapproj"))
```

将 helpers.R 如下图所示的那样存放在你的 census-app 目录中。



helpers.R 中的 percent_map 函数携带五个参数：

Argument Input

var counties.rds 数据集中的列向量（a column vector from the counties.rds dataset）

color colors()颜色名称（any character string you see in the output of colors()）

legend.title 作为图表图例标题的字符串（A character string to use as the title of the plot’s legend）

max （控制阴影范围的一项参数(默认到 100))A parameter for controlling shade range (defaults to 100)

min （控制阴影范围的一项参数（默认到 0)) A parameter for controlling shade range (defaults to 0)

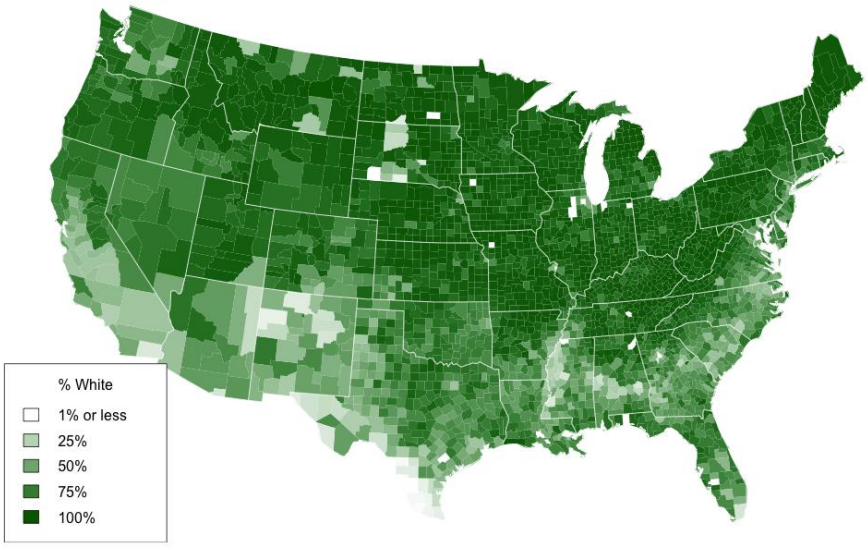
你可以在命令行中使用 percent_map 来把 counties data 绘制为等值线地图，就如以下这样。

```
library(maps)
library(mapproj)
source(“census-app/helpers.R”)
counties <- readRDS(“census-app/data/counties.rds”)
percent_map(counties$white, “darkgreen”, “% white”)
```

注意：以上代码的有效运行时以 census-app 目录是你的工作目录的一个子目录为前提。请确保将你的工作目录设置为 census-app 的父目录。如果需要更改你的工作目录位置，点击 RStudio 菜单栏中的 Session>Set Working Directory>Choose Directory…进行设置。

percent_map 将 counties data 绘制为等值线地图（地区分布图），以下是上

一段代码执行后的效果图，也即用深绿色表示每个郡中白种人居民的比例。



加载文件和文件路径

看一看上面涉及到的代码。为了使用 percent_map，我们首先利用 source 函数执行了 helpers.R，然后利用 readRDS 函数加载 counties.rds 数据集。我们同样运行了 library(maps)和 library(mapproj).

在应用中使用 percent_map 之前，你需要向 Shiny 请求调用相同的函数，但是不同之处在于：你如何编写这些函数直接影响到后续功能的实现。source 和 readRDS 都需要一个文件路径，文件路径在执行时不会与它们在命令行时一样。当 Shiny 执行 server.R 的命令时，它默认为所有文件路径都与 server.R 的目录一致。换言之，你存放 server.R 的目录会变成你 Shiny 应用的工作目录。

既然你把 helpers.R 与 server.R 存放在同一个目录下，那么你可以请求 Shiny 加载它，请用如下命令：

```
source("helpers.R")
```

既然你已经将 counties.rds 保存在 server.R 所在目录的子目录下（data 文件夹中），你可以这样加载它：

```
counties <- readRDS("data/counties.rds")
```

加载 maps 和 mapproj 包只需按照普通的方式即可：

```
library(maps)
library(mapproj)
```

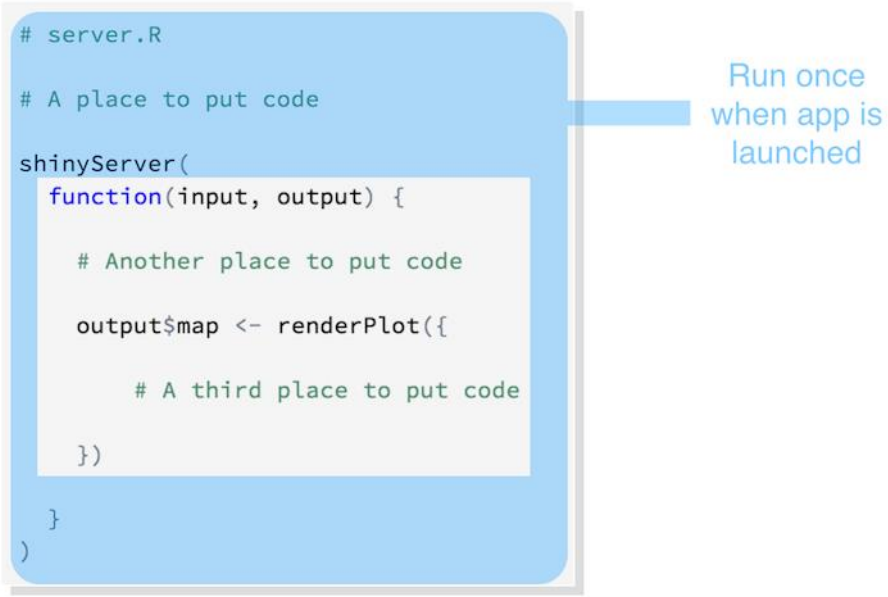
执行（Execution）

Shiny 将会执行所有存放在 server.R 脚本中的命令。然而，你把它们在 server.R 置入的位置将会决定它们运行（或再运行）的次数，而这将会影响到你应用的表现情况。

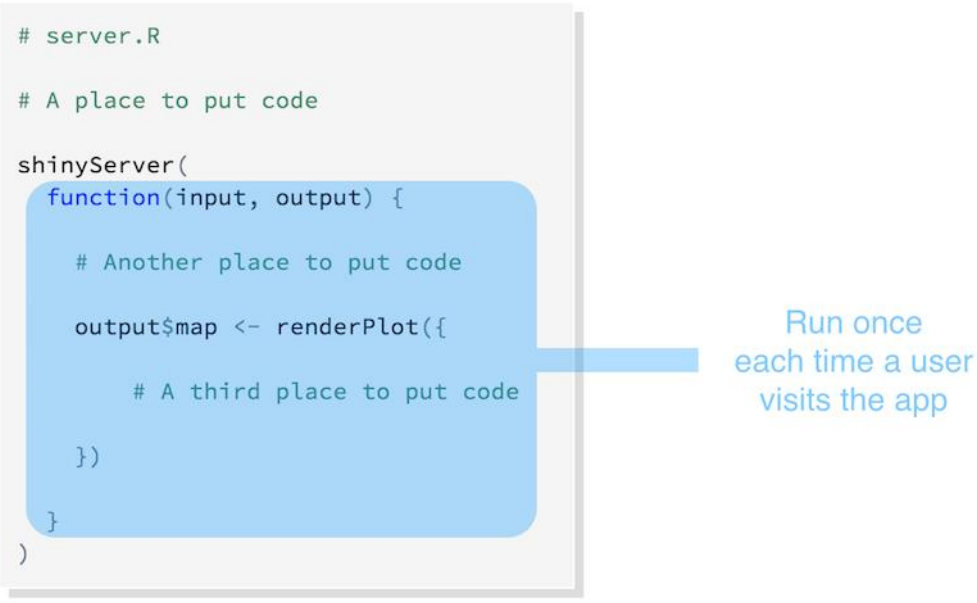
Shiny 对于 server.R 中的一些部分比其他部分运行的更为频繁。

当你首次调用 runApp 时，Shiny 会执行所有脚本。这致使 Shiny 执行

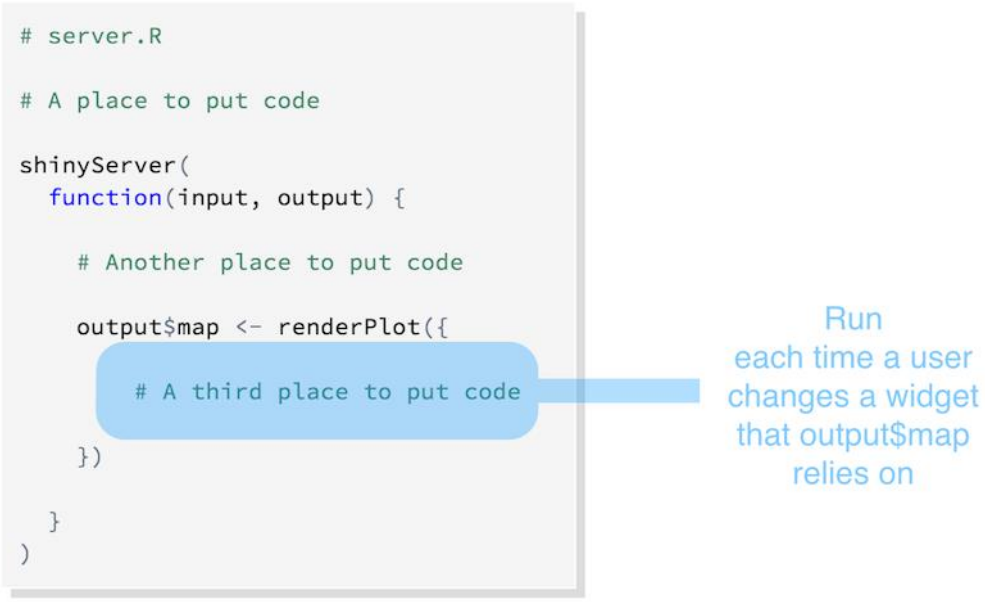
shinyServer 中的代码。



直到新用户来临前，Shiny 会保存这个未命名函数。每次一位新用户访问你的应用，Shiny 就会在此执行一次未命名函数。函数帮助 Shiny 为每位用户建立不同的响应式对象组。



当用户更改部件后，Shiny 会为每个分配的响应式对象重新执行 R 表达式。如果你的用户使用非常频繁，那么这些表达式很可能在一秒内就执行许多许多次。



我们目前学到的知识有：

- 当你运行应用时，server.R 脚本运行一次
- shinyServer 中的未命名函数当用户每次访问你的应用时运行一次
- render＊函数中的 R 表达式会运行许多次。当用户每次更改部件后，Shiny 都要运行它们。

你应该如何使用这些信息呢？

在 shinyServer 函数外的 server.R 开始导入源脚本，加载 libraries，读取数据集。Shiny 仅对这些代码执行一次，这是保证你的 server 能够执行 shinyServer 中的 R 表达式的所有要求。

在 shinyServer 的未命名函数中定义用户特别对象，但是这些定义会在任何 render＊调用之外完成。例如，一个记录用户的 session information（http://shiny.rstudio.com/articles/client-data.html ）的对象。这些代码会根据每位用户的运行各执行一次。

Shiny 需要在一个 render＊函数下建立一个对象。当用户改变代码块中的任何一个部件，那么 Shiny 会重新运行 render＊代码块中的所有代码。这是非常频繁的。

在一般情况下，你应该尽量避免把不必要的代码放置在 render 函数中，否则代码的多次重复执行会放缓整个应用的运行速度。

注意

这一章课程中的两个 Your Turn 对学习 Shiny 应用建构非常有帮助，建议大家尝试进行操作，但是这些 Your Turn 的操作任务提示不包含在本翻译的范畴内。链接为：http://shiny.rstudio.com/tutorial/lesson5/

完成应用

这个 censusVis 拥有一个响应式对象，一个名为"map"的图表。这个图表是

在 percent_map 函数的基础上建构的，该函数携有 5 个参数。

- 前三个参数：var，color，legend.title 取决于下拉选框部件的数值。
- 后两个参数：max，min，分别是滑动条部件的最大值与最小值。

以下的 server.R 脚本反映精巧建构 percent_map 中响应式参数的一种方法。R 的 switch 函数可以将下拉选框部件中的输出转化为你想要的任何东西。但是，这个脚本是缺失的，它尚未提供 color，legend.title，max 或者 min 的数值。注意：这个脚本不能正常运行。你需要在运行前补全这个脚本，这也是你在 Your Turn 2 中面临的任务。

```
# server.R

library(maps)
library(mapproj)
counties <- readRDS("data/counties.rds")
source("helpers.R")

shinyServer(
  function(input, output) {
    output$map <- renderPlot({
      data <- switch(input$var,
        "Percent White" = counties$white,
        "Percent Black" = counties$black,
        "Percent Hispanic" = counties$hispanic,
        "Percent Asian" = counties$asian)

      percent_map(var = data, color = ?, legend.title = ?, max = ?,
min = ?)
    })
  }
)
```

扼要重述

你可以通过加载 R 脚本、R 包和数据集来创造更多复杂的应用。

请铭记于心：

- Server.R 所在的目录会成为 Shiny 应用的工作目录
- Shiny 会执行 server.R 开头部分的代码（在 shinyServer 之前的部分），这只会在应用的生命过程中执行一次
- Shiny 会多次执行在 shinyServer 里的代码，这将会减慢应用的运行速度。

你已认识到 switch 是一个操纵 Shiny 选择部件的有用的工具。当你把部件的数值换成 R 表达式时，请使用 switch。

当你的应用变得趋于复杂时，它们会变得效率低下且速度滞缓。课程 6 会向你展示如何运用响应式的表达式来搭建快速的、模块化的应用。

教程 6-使用响应式表达式

Shiny 应用总能让你的用户啧啧称奇，惊艳于它的运行速度之快，简直达到 instantly fast 的迅捷程度。但是，如果你的应用需要做很多缓慢的计算时应该怎么办？

这个命题是恒久远的，本教程就会向你展示如何让你的 Shiny 应用在响应式表达式的协助下变得更为流畅（streamline）。响应式表达式让你自由操控应用中需要更新的部分，并且阻止一些不必要的工作。

开始之前，我们需要做的准备工作有：

- 在你的工作目录中新建一个文件夹，命名为“stockVis”。
- 下载三个文件并将它们放置于 stockVis 中，这三个文件分别是：ui.R（http://shiny.rstudio.com/tutorial/lesson6/stockVis/ui.R），server.R（http://shiny.rstudio.com/tutorial/lesson6/stockVis/server.R），helpers.R（http://shiny.rstudio.com/tutorial/lesson6/stockVis/helpers.R）。
- 通过 runApp(“stockVis”)来运行应用。

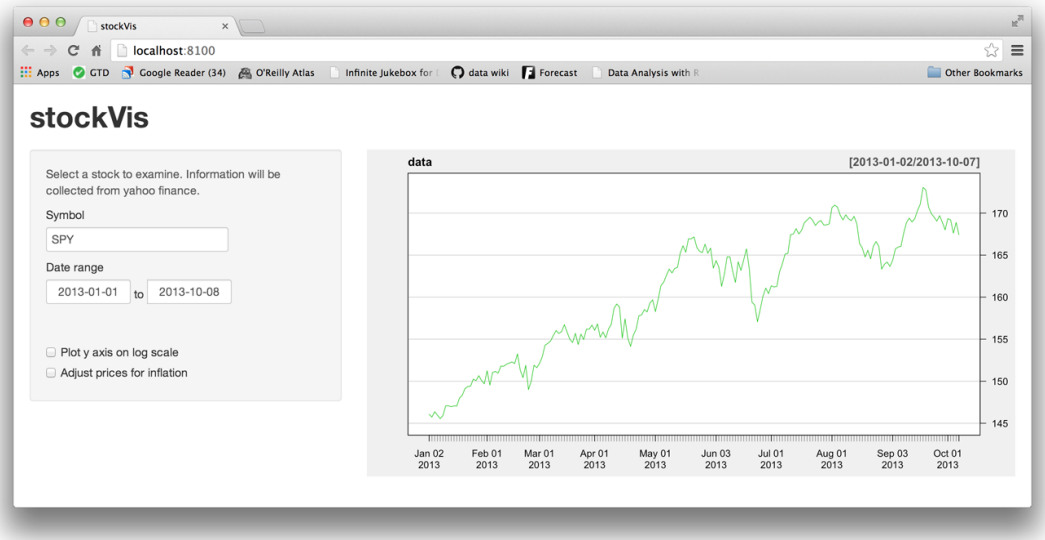
stockVis 利用 R 中的 quantmod 包（译者注：quantmod 是 R 中的一款金融分析包），如果你尚未安装此包你需要现在安装 quantmod 包，安装方法：

```
install.packages(“quantmod”)
```

一个新的应用程序：stockVis

stockVis 应用根据股票代号（ticker symbol）来寻找股票价格，并且以折线图的形式来展现结果。这个应用可以让你：

1. 选择一个股票来加以检验（examine）；
2. 选择一个时间段来加以阅览；
3. 选择 y 轴上的数据展示形式：是绘制股票价格值还是股票价格的对数（log）值；
4. 决定是否要出于通货膨胀的考量来修改股票价格。



请注意图中的“Adjust prices for inflation”复选框尚不能正常工作。我们本次

教程的任务之一就是修正这个复选框。

默认情况下，stockVis 显示 SPY 股票代码（完整的 S&P 500 的指数）。想要查找不同的股票，请键入一个雅虎财经（Yahoo finance）能够识别的股票代码。你可以从这里（http://finance.yahoo.com/lookup）寻找一连串的雅虎股票代码。一些常用的代码有 GOOG（Google），AAPL（Apple）和 GS（Goldman Sachs）。

StockVis 主要依赖 quantmod 包里的两个功能：

1. getSymbols，用来从 Yahoo finance 和 Federal Reserve Bank of St. Louis 等类似网站直接下载财经数据到 R 中；
2. chartSeries，用一种有吸引力的图表形式来展现价格。

StockVis 还依赖一个名为 helpers.R 的 R 脚本（译者注：在前几节教程中已有所涉及），里面包含调控股票价格的一个函数。

复选框与日期范围（Check boxes and date ranges）

stockVis 使用几个新的部件：

- 一个日期范围选择器，由 dataRangeInput 创建；
- 一对复选框，由 checkboxInput 创建。复选框部件非常简单。当框被选中时，返回 TRUE 值，当框处于未选中状态时，处于 FALSE 值。

在 ui.R 脚本中，复选框分别被命名为 log 和 adjust，这表明你可以在 server.R 脚本中以 input$log 和 input$adjust 来寻找相应复选框。如果你想要回顾如何使用部件以及它们的数值，请参阅教程 3 与教程 4.

简化计算（Streamline computation）

stockVis 应用存在一个问题。

尝试一下，当你点击“Plot y axis on the log scale”时会发生什么。Input$log 的数值将会变更，这就造成 renderPlot 的全部表达式再次运行。

```
output$plot <- renderPlot({
  data <- getSymbols(input$symb, src = "yahoo",
    from = input$dates[1],
    to = input$dates[2],
    auto.assign = FALSE)

  chartSeries(data, theme = chartTheme("white"),
    type = "line", log.scale = input$log, TA = NULL)
})
```

每次 renderPlot 重新运行时，

1. 它重新根据 getSymbols 从 Yahoo finance 上再次抓取数据；
2. 以更改后的坐标轴来重新绘制图表。

这并不符合我们的预期，因为你不需要再次抓取数据、也不希望再次绘制图表，Yahoo finance 将会因为你过于频繁地抓取数据而切断你的正常访问（因为你的行为像一只马蝇（bot））。但更为重要的是，重复运行 getSymbols 是无用功，这只会减慢应用的运行速度，消耗服务器的带宽。

响应式表达式

响应式表达式帮你限制部分代码在响应过程中的重复运行。

响应式表达式是一种 R 表达式类型，它使用部件值输入并且返回到一个值。当原有部件进行变更时响应式表达式会自动更新数值。

创建响应式表达式需要使用 reactive 函数，它被大括号{}包围（就如 render* 函数一样）。

例如，以下是利用 stockVis 部件从 Yahoo 上抓取数据的响应式表达式。

```
dataInput <- reactive({
  getSymbols(input$symb, src = "yahoo",
    from = input$dates[1],
    to = input$dates[2],
    auto.assign = FALSE)
})
```

当你运行这段表达式时，它会运行 getSymbols 然后返回结果，结果就是一个价格数据的数据框（data frame）。你可以在 renderPlot 中调用 dataInput()来利用表达式接触价格数据。

```
output$plot <- renderPlot({
  chartSeries(dataInput(), theme = chartTheme("white"),
    type = "line", log.scale = input$log, TA = NULL)
})
```

响应式表达式比常规的 R 函数更为灵活。它们缓存自己的数值并且知道在何时数据开始过时。这意味着什么？当你初次运行一个响应式表达式，这个表达式就会将自己的运行结果存放在你的电脑内存中。下一次你再次调用这个响应式表达式，它会在不执行任何计算的前提下返回已保存的结果（这样保证了应用速度的迅速）。

响应式表达式只会返回保存的结果如果它知道结果是最新的。如果响应式表达式发现结果是过时的（因为部件已经更改），表达式会重新计算结果。它会返回新结果并且保存一份新的副本。响应式表达式会使用新的副本直到新副本再次过时。

让我们来总结一下这种操作行为：

- 首次运行时，响应式表达式就保存结果；
- 下一次调用响应式表达式时，它会检查上一次保存的数值是否过时（也即，它所依赖的部件是否做出改变）；
- 如果数值过时，响应式对象就是重新执行计算（然后保存新的结果）；
- 如果数值保持更新，响应式表达式会在不执行任何计算的前提下返回已保留的数值。

你可以利用这种操作行为来阻止 Shiny 运行不必要的代码。看看以下代码，思考一下响应式表达式究竟是如何工作的。

```
# server.R

library(quantmod)
source("helpers.R")

shinyServer(function(input, output) {

  dataInput <- reactive({
```

```
    getSymbols(input$symb, src = "yahoo",
        from = input$dates[1],
        to = input$dates[2],
        auto.assign = FALSE)
  })

  output$plot <- renderPlot({
    chartSeries(dataInput(), theme = chartTheme("white"),
        type = "line", log.scale = input$log, TA = NULL)
  })
})
```

当你点击“Plot y axis on the log scale”时，input$log 会进行变更，renderPlot 会重复执行，现在：

1. renderPlot 调用 dataInput()；
2. dataInput 会检查日期与 symb 部件是否有过变更；
3. dataInput 会在不重复从 Yahoo 抓取数据的前提下返回到预先保存的股票价格数据集；
4. renderPlot 会以更改后的坐标轴来重绘图表。

相关性（Dependencies）

如果你的用户在 symb 部件中更改了股票代码呢？

这就使 renderPlot 绘制的图表过时了，但是 renderPlot 不再调用 input$symb。Shiny 会知道 input$symb 已经让图表过时了吗？

是的，Shiny 会知道相关情况并且重绘图表。Shiny 会与依靠对象的 output 的响应式表达式保持同步，当部件输入变动时就开始同步。Shiny 在以下情况中会自动重建一个对象：

- 在对象中的 render*函数中一个 input 数值发生改变；
- 在对象中的 render*函数中响应式表达式过时。

你可以加以想象：响应式表达式就如连接 input 数值与 output 对象之间的一个链条。Output 中的对象将会对链条下游的任何改动做出响应。（你可以在脑海中构建一个长链条，因为响应式表达式可以调用其他的响应式表达式）。

只有在 reactive 或者 render*函数中的响应式表达式才能够被调用。这是为何呢？因为只有这些 R 函数才能赋予响应式输出的功能，并且改变尽在不言中。事实上，Shiny 会阻止你在这些函数之外调用响应式表达式。

扼要重述

利用响应式表达式，你可以对代码进行模块化，从而保证自己的应用运行更加快速。

- 一个响应式表达式包含 input 数值，或者来源于其他响应式表达式的数值，并且返回一个新数值；
- 响应式表达式保存自己的运行结果，当且仅当输入变化时才重新执行计算；
- 用 reactive{}来创建响应式表达式；
- 在表达式名称后插入()来调用响应式表达式；
- 只有从其他响应式表达式或者 render*函数中调用响应式表达式。

你现在可以创造复杂的，运行流畅的 Shiny 应用。最后一个教程将向你展示如何与他人分享你的 Shiny 运用程序。

注意

本教程中的联系包含 Warm up 与 Your Turn，非常具有指导意义，可以自行尝试。

由于每一处的联系都有标准答案代码（点击联系下方的 Reveal answer 即可），本处不予翻译。请自行查看：http://shiny.rstudio.com/tutorial/lesson6/

课程 7-分享你的应用

现在你可以创建一个有用的 Shiny 应用了，但怎样与小伙伴们一同分享自己的应用呢？本课程将向你展示与他人分享 Shiny 应用的几种方法。

两个基本的分享选择是：

1. **以两个文件 server.R 和 ui.R 的形式来分享你的 Shiny 应用**。这是最简单的分享方式，但是仅当你的用户在电脑上安装了 R，并且知晓如何使用 R 后，这才得以实现。用户可以在自己的电脑上运行这些脚本来打开程序，就像你平常运行应用程序一样。
2. **以网页形式分享你的 Shiny 应用**。这无疑是分享应用最友善的途径。用户可以通过互联网打开浏览器对应用进行操作。他们会发现你的应用完全渲染呈现、即刻更新，并且操作自如。

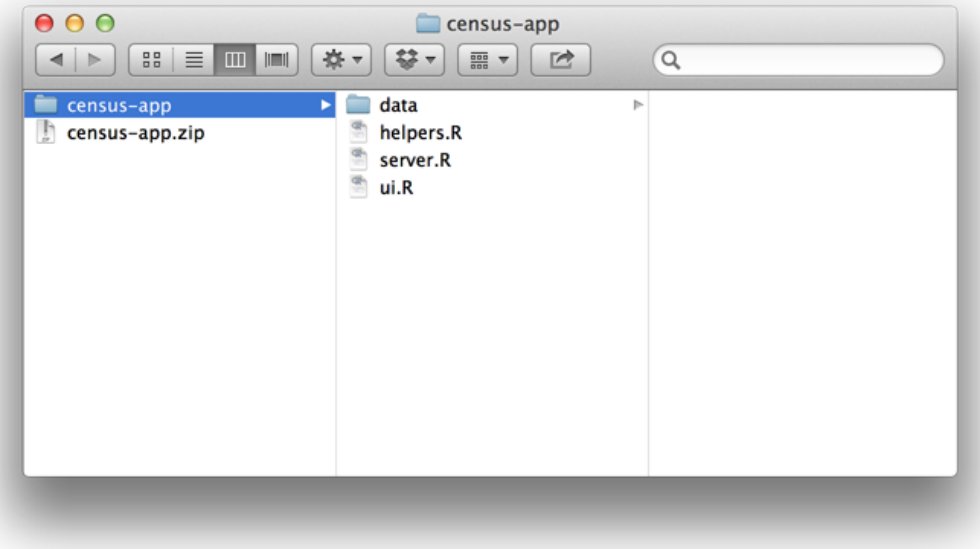
两个 R 文件的共享形式

拥有 R 语言的任何人都可以运行你的应用。他们仅需要你的 server.R 和 ui.R 的一份副本（copy）即可，同时在一些情况下也需要一些应用中的补充材料（如 www 文件夹或者是 helpers.R 文件）。

可以通过发送 email(以 zip 文件的形式)或者在线共享来传播你的文件。

你的用户可以将文件存放在一个他们自己的目录中。仅需在命令行中输入以下命令即可运行，与你在电脑上的操作别无二致。

```
#install.packages("shiny")
library(shiny)
runApp("census-app")
```



Shiny 拥有三个内置命令可以使打开在线文件变得轻松快捷，它们是：runUrl, runGitHub 和 runGist.

runUrl

runUrl 会直接从网络链接中下载运行一个 Shiny 应用。

使用 runUrl 的方法是：

- 将你的 Shiny 应用目录保存为 zip 文件
- 在网页中上传 zip 文件获得超链接，任何人都可以访问链接并且加以运行，只需在 R 命令行中输入：

```
library(shiny)
runUrl("<the web link>")
```

runGitHub

如果你没有属于自己的网页来托管文件，你可以在 www.github.com 上免费托管自己的文件。

使用 GitHub，你需要注册（sign up）并且选取一个用户名（user name）。

通过 GitHub 来分享自己的应用，请在 GitHub 上创建一个项目存储库（project repository），然后在存储库中保存你的 server.R 和 ui.R 文件，以及一些运行程序所需的补充文件。

你的用户可以通过以下代码运行你的 Shiny 应用。

```
runGitHub("<Your repository name>","<your user name>")
```

runGist

如果你希望以匿名形式发布你的在线文件，GitHub 提供了一个 pasteboard 服务，详情访问（http://gist.github.com/ ）。你不需要注册 GitHub 来使用它的常规服务。即使你已经拥有一个 GitHub 账户，Gist 仍然可以作为一种简单、快捷的 Shiny 项目共享方式。

- 复制粘贴你的 server.R 和 ui.R 文件到 Gist 的网页中
- 注意 GitHub 给你的 Gist 生成的 URL

一旦你制作了一个 Gist,你的用户可以通过输入 runGist("<gist number>") 来运行应用。"<gist number>"显示的是你的 Gist 网址中后部分的数字。

例如 `runGist(“3239667”)`

以网页形式分享应用

以上的分享方式有着同样的限制条件，他们需要你的用户事先在电脑上安装好 R 与 Shiny 包。

然而，Shiny 提供了绝好的分享方式，使得用户安装 R 语言不是必须的硬性要求。你的 Shiny 应用会用世界上最流行的传播工具：网页，来进行分享。如果你有自己应用的 URL，用户可以访问这个应用程序（并且丝毫不必担心代码的问题）。

如果你对网页保存应用驾轻就熟，或者你有权限接触某个 IT 部门，你可以自己主宰自己的 Shiny 应用。

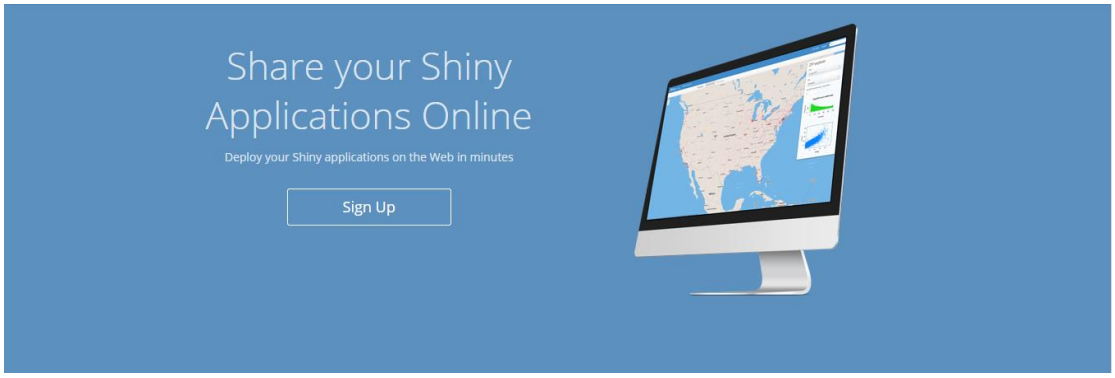
如果你期待一个更为简便的方式或者需要专家支持，RStudio 提供三种方法来将你的 Shiny 应用放入网页中，这三种方法是：

1. Shinyapps.io
2. Shiny Server
3. Shiny Server Pro

Shinyapps.io

Shinyapp.io 是将你的 Shiny 应用变成网页形式的最简易方法，这是 Rstudio 为 Shiny 应用提供的一项托管服务。

Shinyapp.io 让你直接从 R 任务中上传自己的应用到 RStudio 的服务器中。你可以在服务器管理器具的帮助下完全控制你的应用。更多信息，详见 shinyapps.io（http://shinyapps.io/ ）.



Shiny Server

Shiny Server 是 Shiny 的一个伙伴项目，为 Shiny 应用的托管建立一个网络服务器。它是免费的、开源的，并且在 GitHub 上可用。

Shiny Server 的特性是：Linux 服务器可以将 Shiny 应用置换为网页。要使用 Shiny Server，你首先要有一个 Liunx 服务器， Ubuntu 12.04 或者更高版本（64 位）以及 CentOS/RHEL 5（64 位）。如果你没有使用显性的支持分布架构，你仍然可以从源代码建构来使用 Shiny 服务器。

利用同一个 Shiny 服务器，你可以托管多个 Shiny 应用在多个网页中，并且你可以从防火墙后部署应用程序。

欲知详情（如何安装、配置 Shiny 服务器），请访问 Shiny Server guide（https://github.com/rstudio/shiny-server/blob/master/README.md）.

Shiny Server Pro

Shiny Server 可以将你的应用全部放到网络上并且照顾你的 Shiny 发布需要。然而，如果你的 Shiny 应用意图赢利，那么你可能想给自己的服务器工具配置大多数常见的付费的服务器程序，比如：

- 密码认证（Password authentification）
- SSL（加密套接字协议层（一种加密的通讯协定，用在使用者与网服器之间，Security Socket Layer））支持（SSL support）
- 管理员工具（Administrator tools）
- 优先级支持（Priority support）
- 更多 and more.

如果是这样的话，请查看 Shiny Server Pro（http://www.rstudio.com/shiny/server/ ），这是 RStudio 针对 Shiny Server 的付费专业版本。

扼要重述

Shiny 应用的分享十分简单。你可以共享的方式有：一对 R 脚本，或带有 URL

的全功能 web 应用程序。每一类方法都各有所长。

你已经学习到：

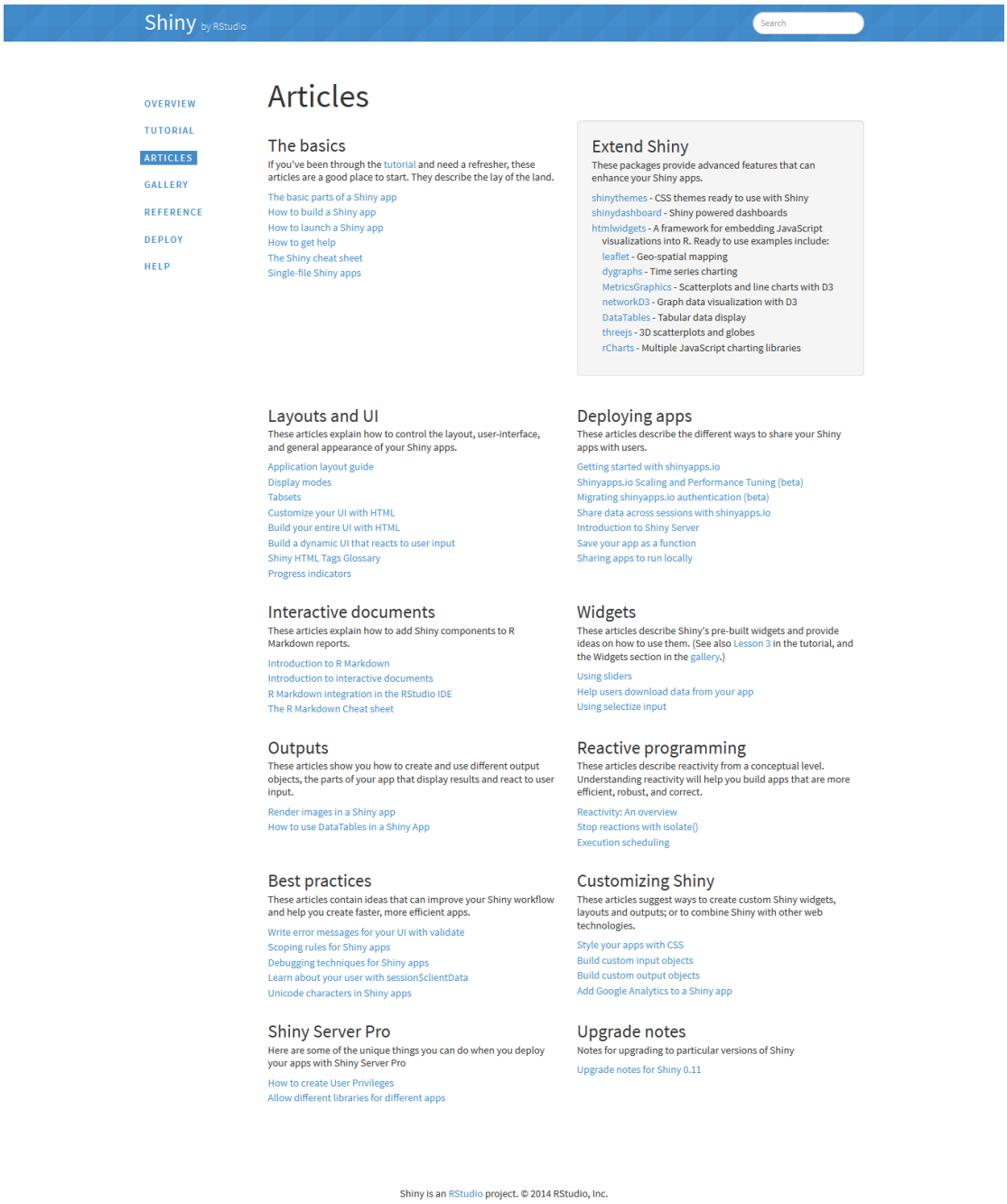
- 每个人都可以运行你的应用程序，只要他们拥有 R 语言，shiny 包和你应用程序的复制版本。
- runUrl，runGitHub，runGist 让分享变得简易快捷，并且保证用户能从网络链接中检索到 Shiny 文件。
- 你可以把你的应用变成灵动的网络应用程序，仅需要使用 shinyapps.io 获取专属 URL。
- 你可以使用开源的 Shiny Server 来搭建 Linux server 来托管 Shiny 应用。
- 如果你需要对应用程序进行密切控制，或者意图管理流量，你可以从 RStudio 中购买 Shiny Server Pro。

祝贺你，你已经学习了完整的 Shiny 开发过程。你现在可以建构一个复杂的、响应式的应用，并且部署它，与他人共享你的成果。用户可以与你的数据进行交互并且以一种新的方式来追随你的故事。

下一步便是勤学苦练，然后继续探索 Shiny 更多的先进功能。

Shiny Dev Center（http://shiny.rstudio.com/ ）可以伴你一路同行，它托管了大量应用程序的 Gallery（http://shiny.rstudio.com/gallery/ ），并且提供了这些应用程序的代码。

同样地，Center 中亦包含文章部分（articles，http://shiny.rstudio.com/articles/ ）供学习者进行深入研究。每一篇文章都深入检视一个中级到高级的 Shiny 主题。



现在的你已具备足够丰富的建造 Shiny 应用程序的知识，开始吧。See what you can do!

附：

Shiny 论坛：Shiny Discussion Forum

https://groups.google.com/forum/#!forum/shiny-discuss

如何获取帮助：How to get help?

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