RStudio

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212

Email: info@rstudio.com
Web: http://www.rstudio.com

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.

RStudio

250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212

Email: info@rstudio.com
Web: http://www.rstudio.com

How to start with Shiny, Part 3

How to customize appearance

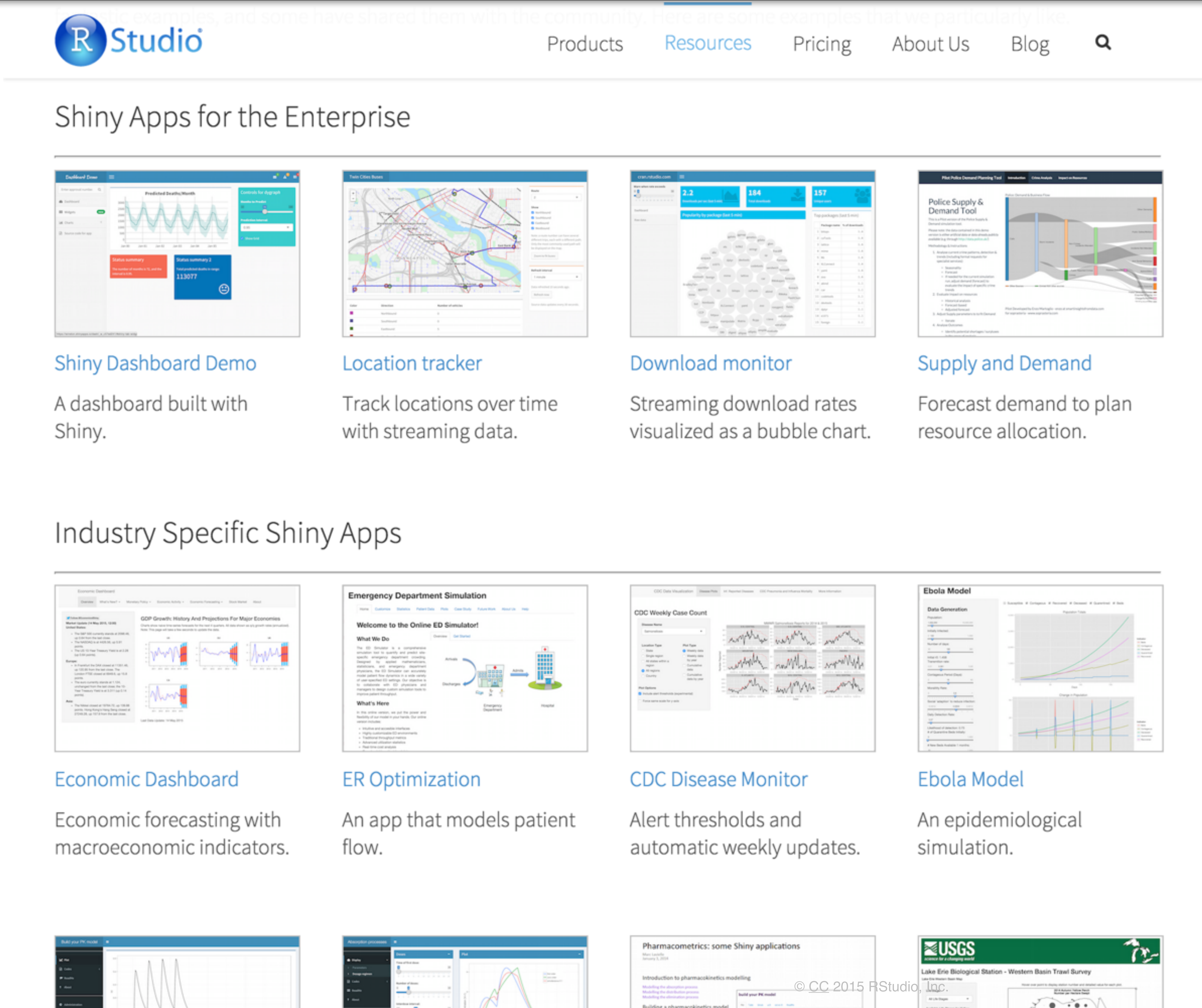
Garrett Grolemund
Data Scientist and Master Instructor
May 2015
Email: garrett@rstudio.com
Twitter: @StatGarrett

Code and slides at:

bit.ly/shiny-quickstart-3

Shiny Showcase

www.rstudio.com/products/shiny/shiny-user-showcase/



tic examples, and some have shared them with the community. Here are some examples that we particularly like.

Shiny Apps for the Enterprise

Shiny Dashboard Demo
A dashboard built with Shiny.

Location tracker
Track locations over time with streaming data.

Download monitor
Streaming download rates visualized as a bubble chart.

Supply and Demand
Forecast demand to plan resource allocation.

Industry Specific Shiny Apps

Economic Dashboard
Economic forecasting with macroeconomic indicators.

ER Optimization
An app that models patient flow.

CDC Disease Monitor
Alert thresholds and automatic weekly updates.

Ebola Model
An epidemiological simulation.

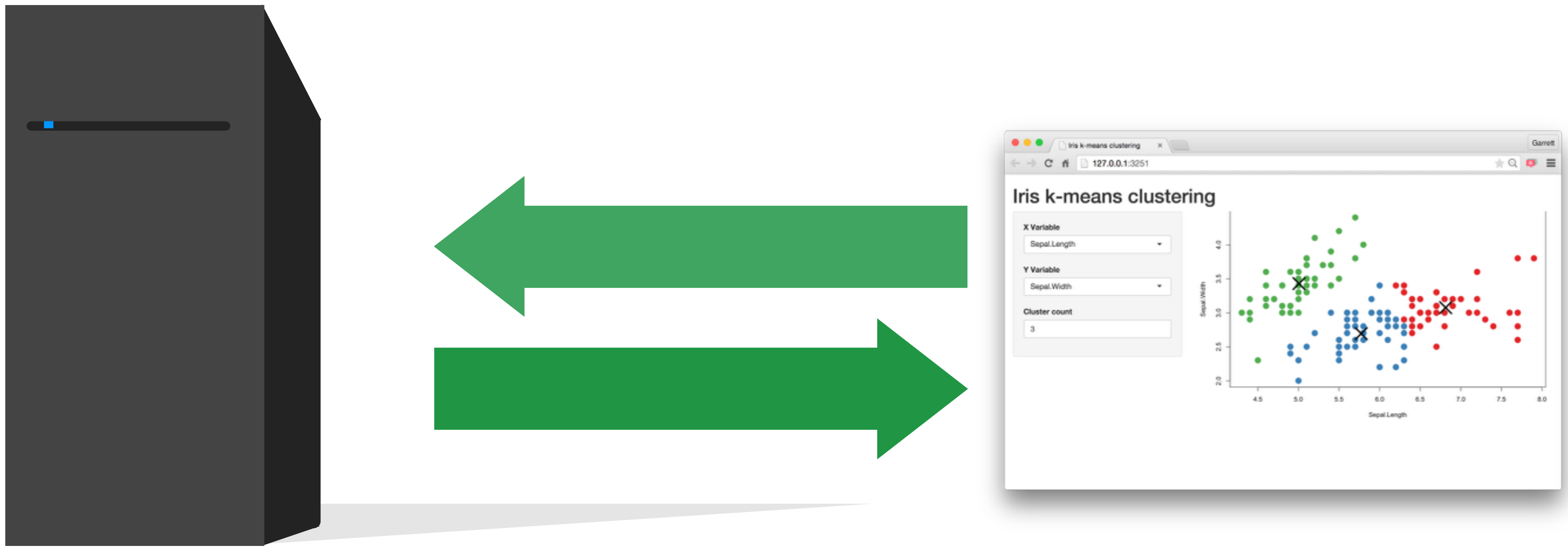
Pharmacometrics: some Shiny applications
USGS
Lake Erie Biological Station - Western Basin Trawl Survey

How to start with Shiny

1. How to build a Shiny app (www.rstudio.com/resources/webinars/)
2. How to customize reactions (www.rstudio.com/resources/webinars/)
3. How to customize appearance (Today)

The story so far

Every Shiny app is maintained by a computer running R



Iris k-means clustering

X Variable: Sepal.Length

Y Variable: Sepal.Width

Cluster count: 3

Sharing apps

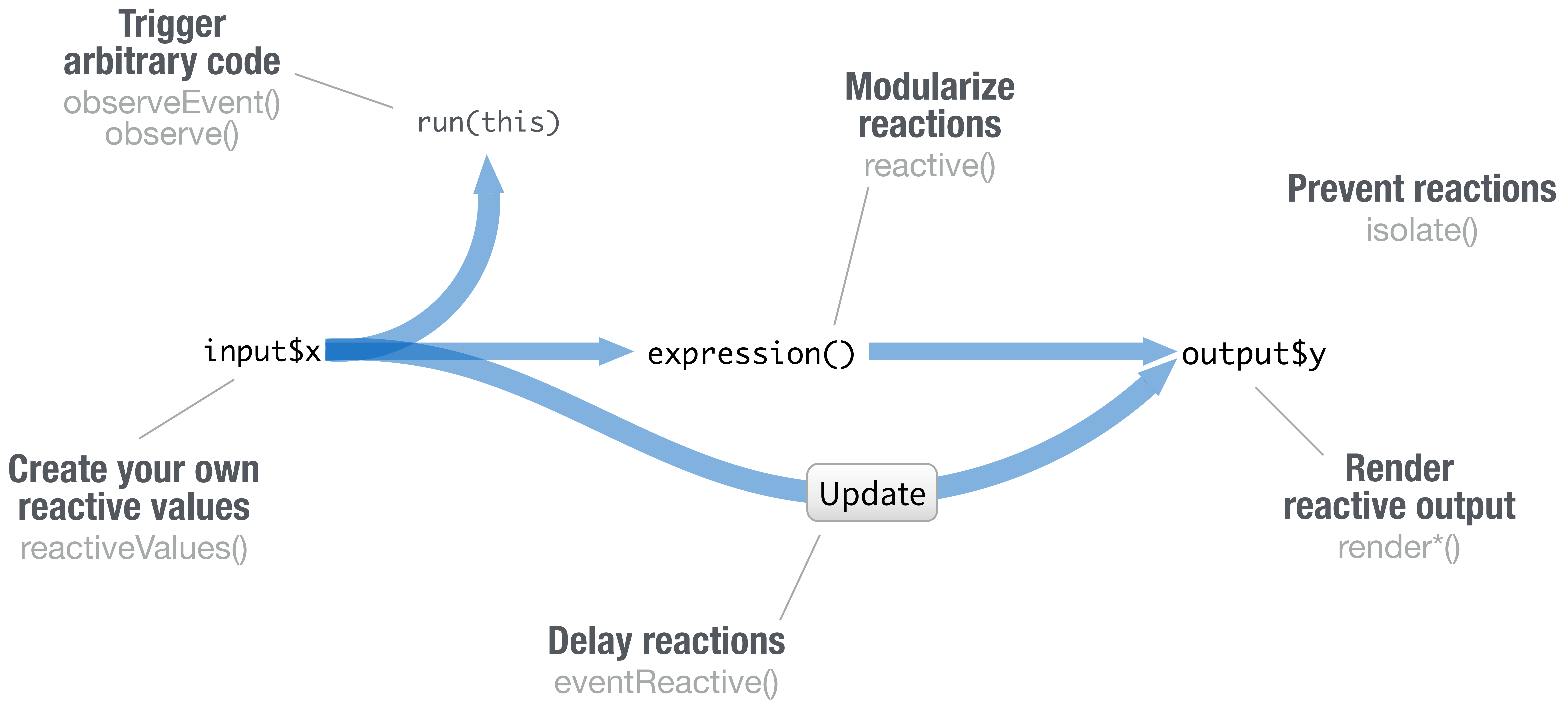
Shiny Server (Pro)

App template

The shortest viable shiny app

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


server <- function(input, output) {}


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

Reactivity



Trigger arbitrary code

observeEvent()
observe()

run(this)

Modularize reactions

reactive()

Prevent reactions

isolate()

input$x

expression()

output$y

Update

Create your own reactive values

reactiveValues()

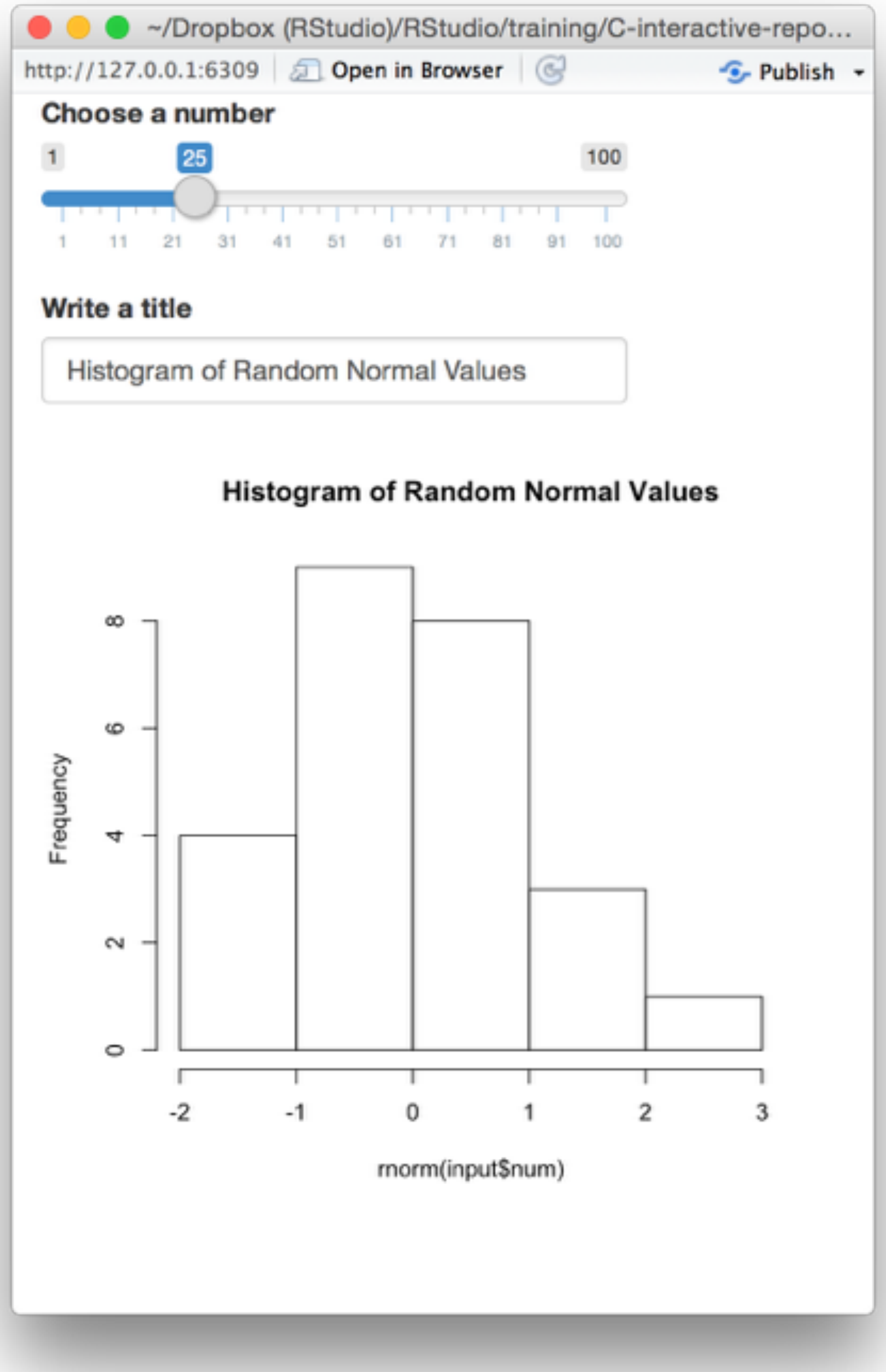
Render reactive output

render*()

Delay reactions

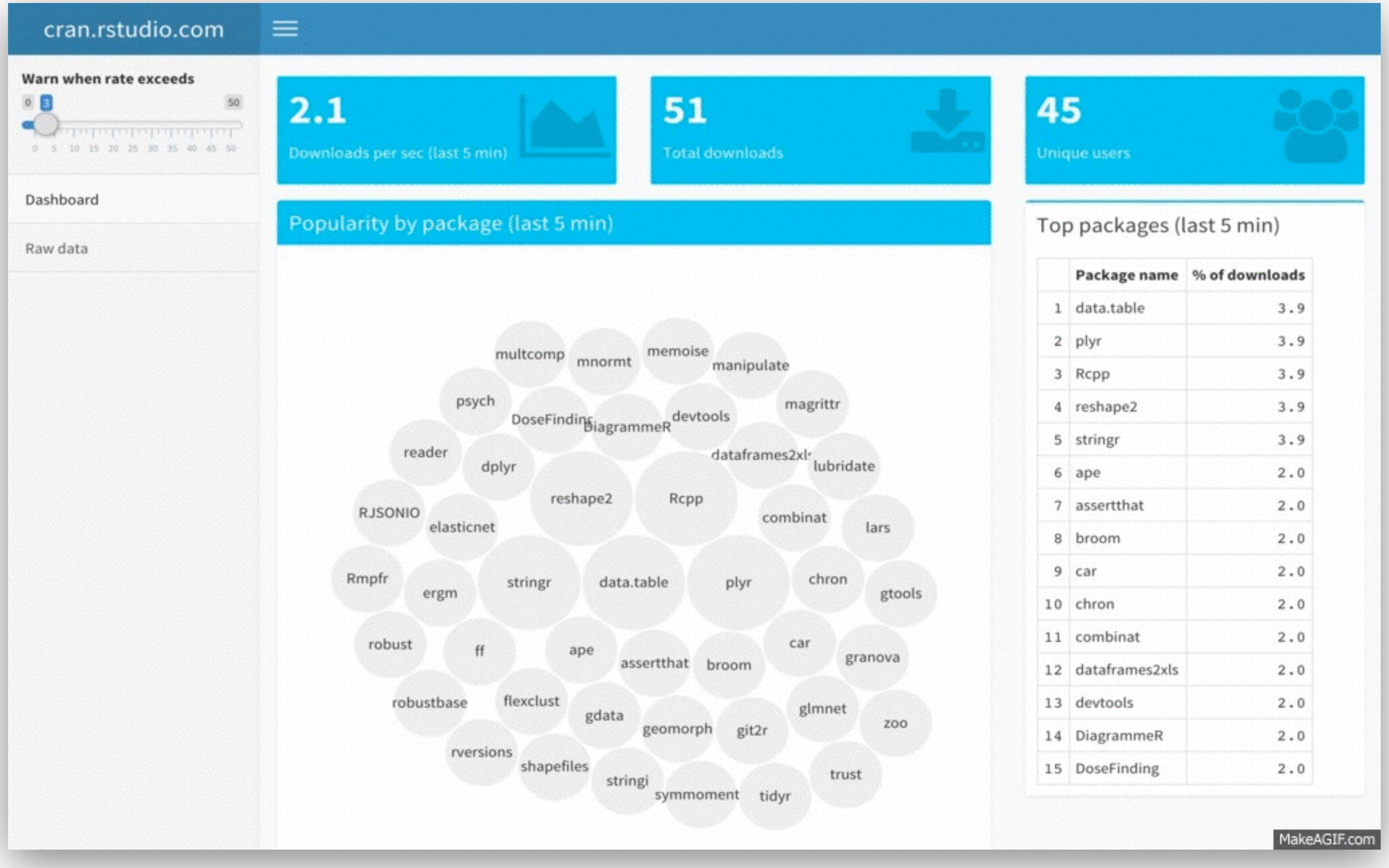
eventReactive()

User Interface



Choose a number: 25

Write a title: Histogram of Random Normal Values



Warn when rate exceeds: 3

2.1 Downloads per sec (last 5 min)

51 Total downloads

45 Unique users

Popularity by package (last 5 min)

Top packages (last 5 min)

	Package name	% of downloads
1	data.table	3.9
2	plyr	3.9
3	Rcpp	3.9
4	reshape2	3.9
5	stringr	3.9
6	ape	2.0
7	assertthat	2.0
8	broom	2.0
9	car	2.0
10	chron	2.0
11	combinat	2.0
12	dataframes2xls	2.0
13	devtools	2.0
14	DiagrammeR	2.0
15	DoseFinding	2.0

Work with the HTML UI

The HTML that builds the user interface for your app

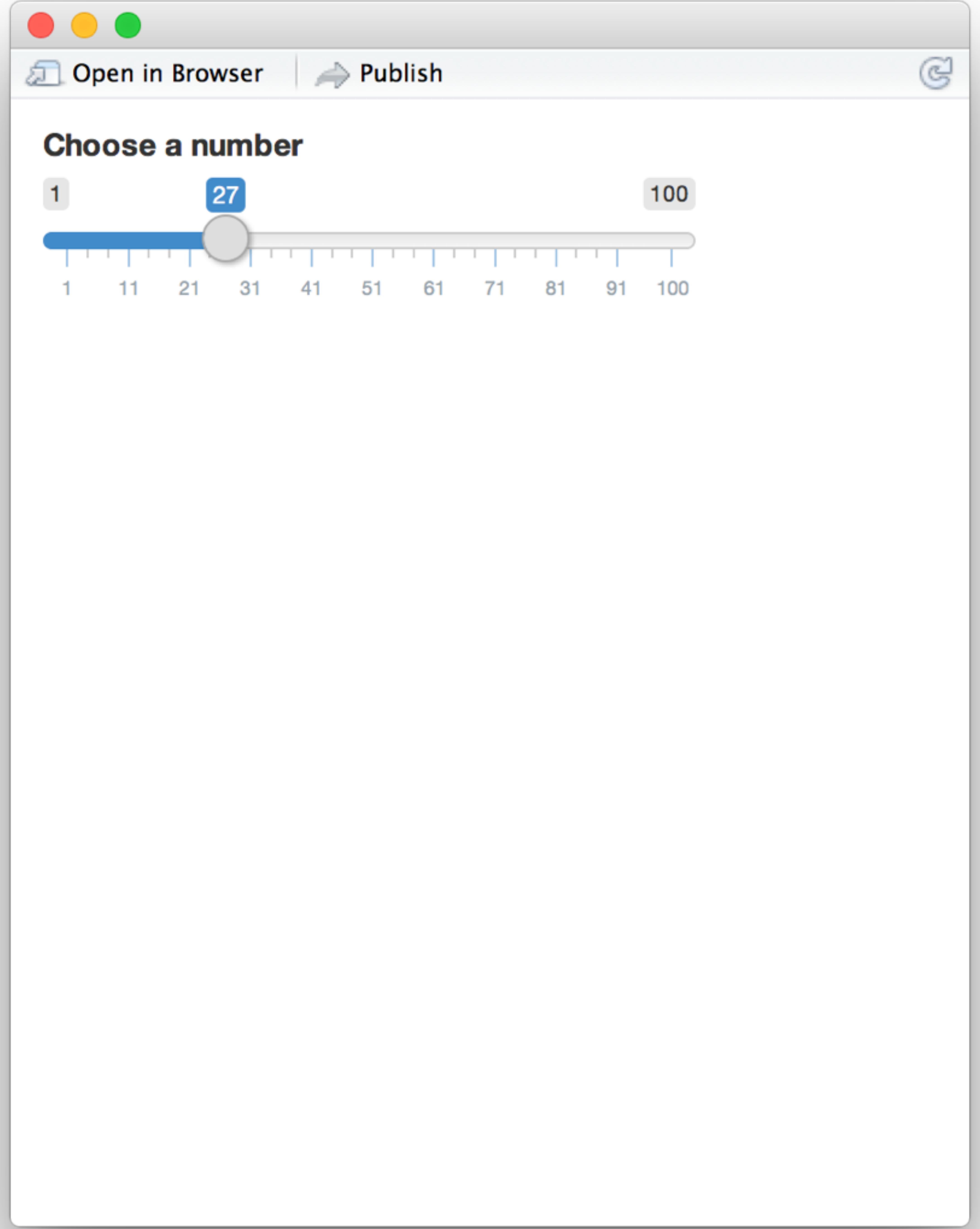
```
ui <- fluidPage()
```

```
fluidPage()
```

```
<div class="container-fluid"></div>
```

```
library(shiny)

ui <- fluidPage(



)

server <- function(input, output) {


}

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

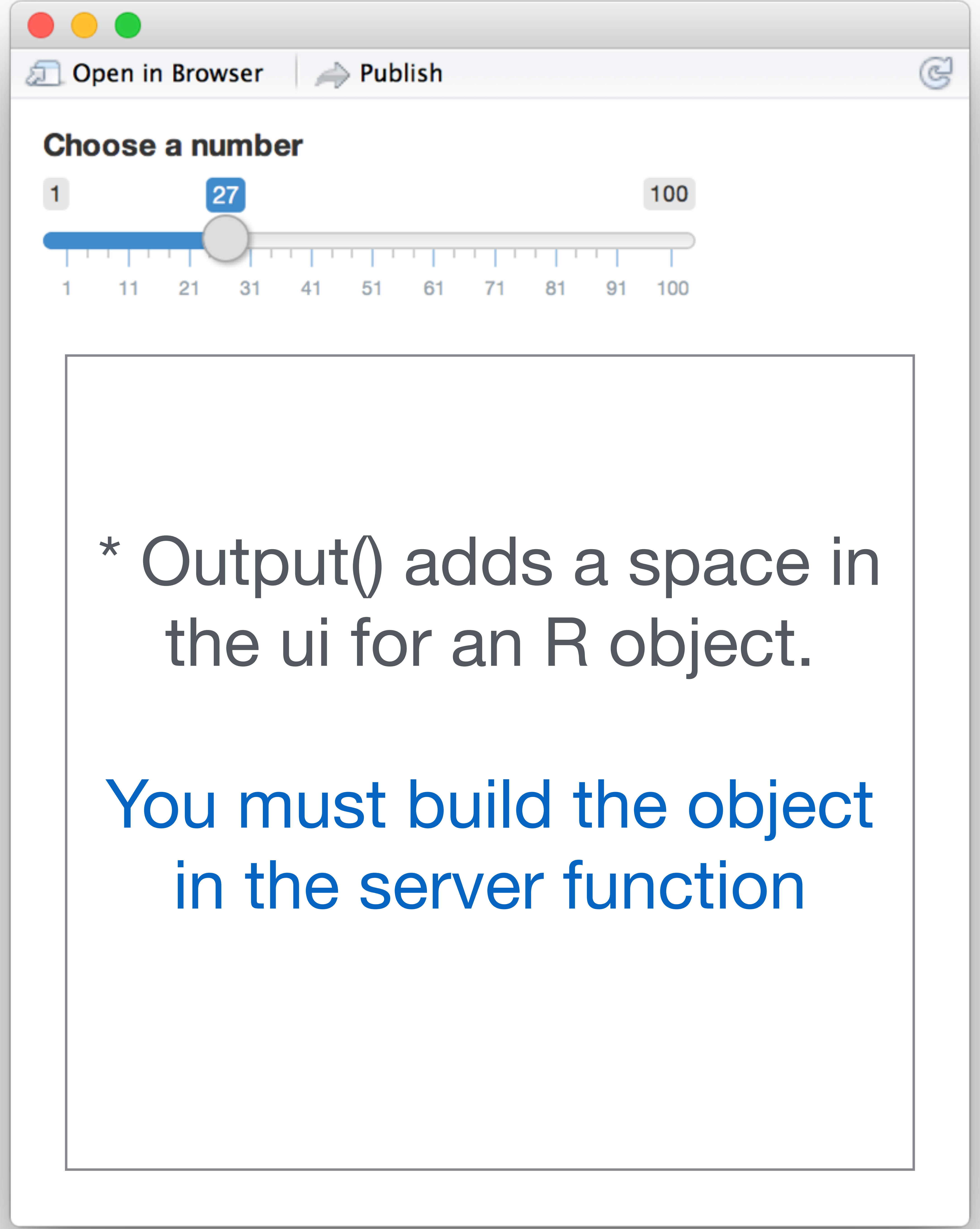
```
library(shiny)

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

)

server <- function(input, output) {



}

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



Choose a number

27

```r
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

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

```
<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>
```

```
plotOutput("hist")
```

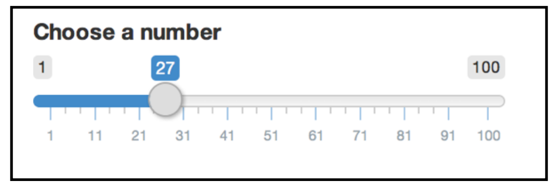
```
<div id="hist" class="shiny-plot-output" style="width: 100% ; height: 400px"></div>
```

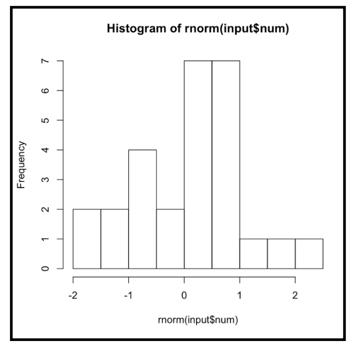
Recap: User Interface

Call R functions to assemble HTML

`fluidPage()`

Use **fluidPage()** to quickly set up a UI





Use **input*()** and **output*()** functions to add reactive content.

`ui <-`

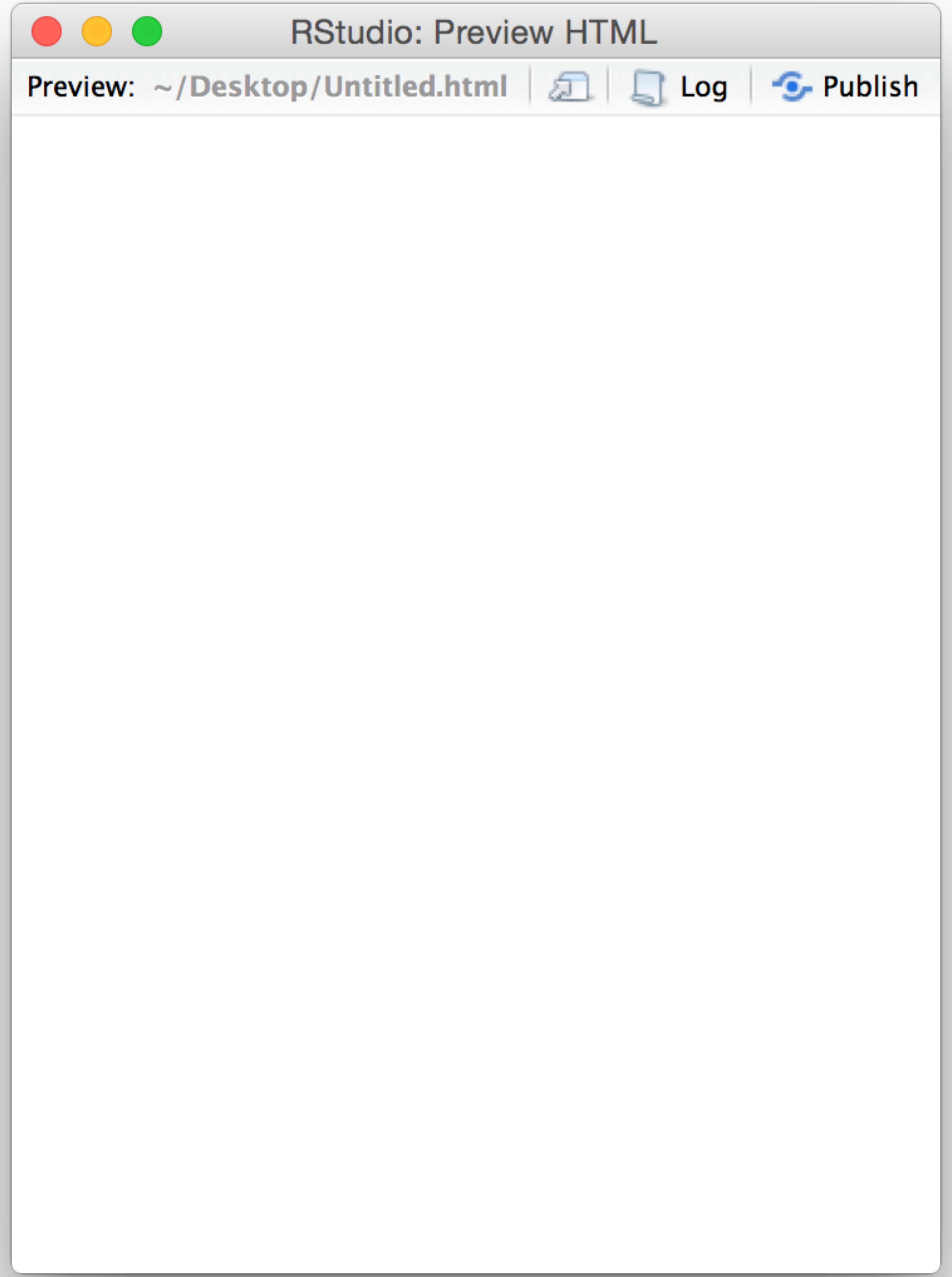
Save the output to **ui**

Add

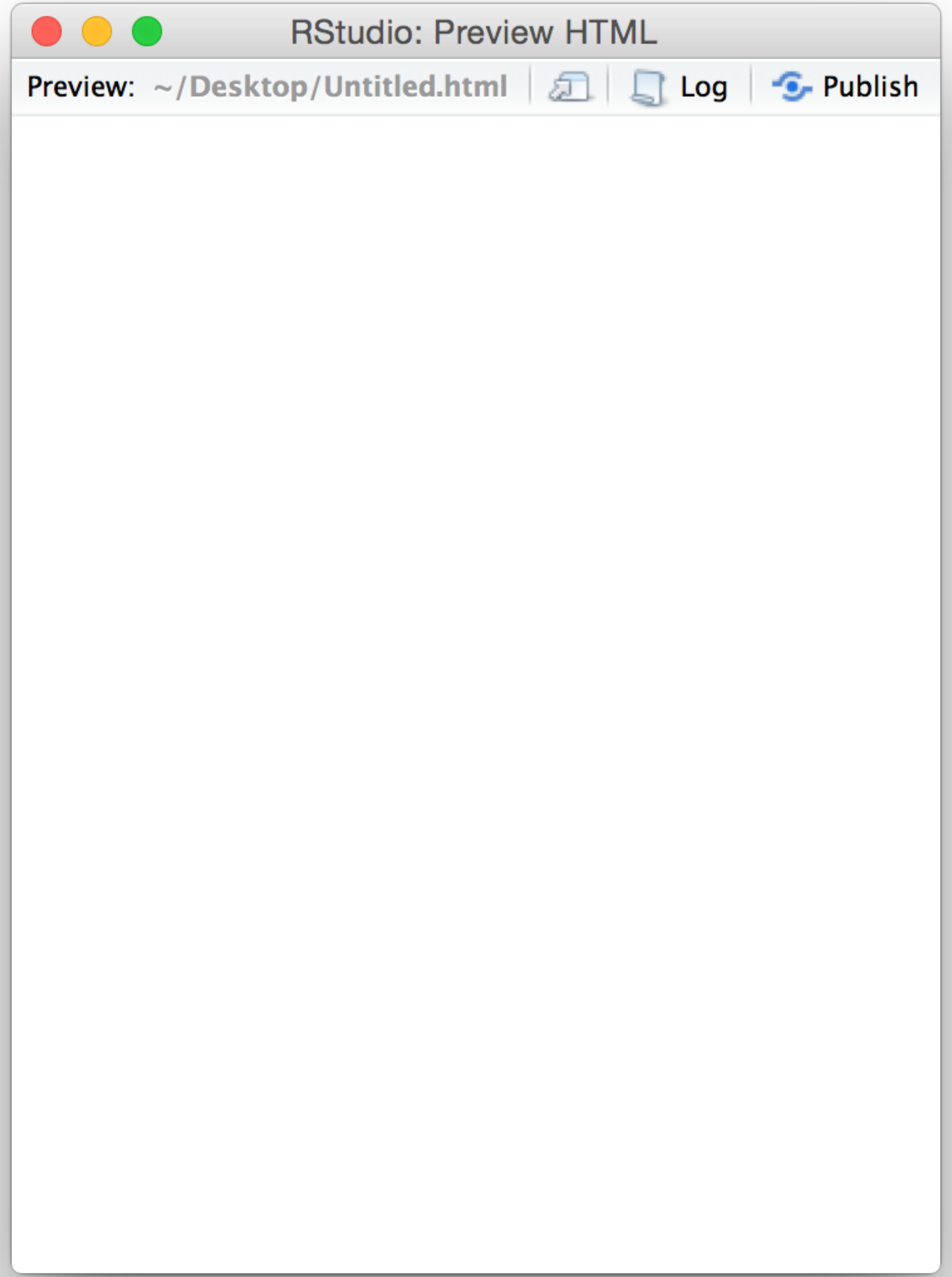
static content

How do you add content to a web page?

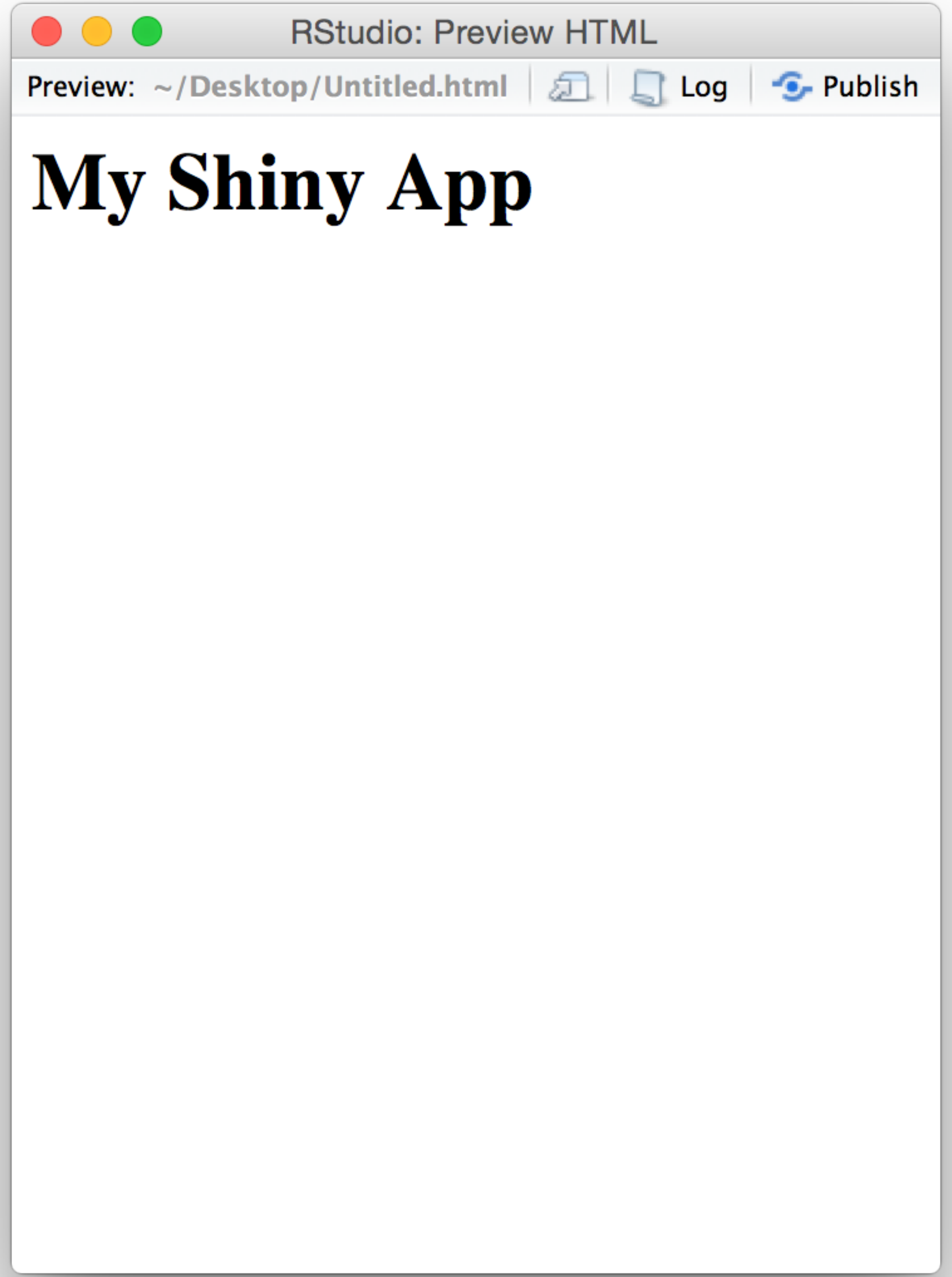
```
<div class="container-fluid"></div>
```


RStudio: Preview HTML
Preview: ~/Desktop/Untitled.html
Log
Publish

```
<div class="container-fluid">


</div>
```

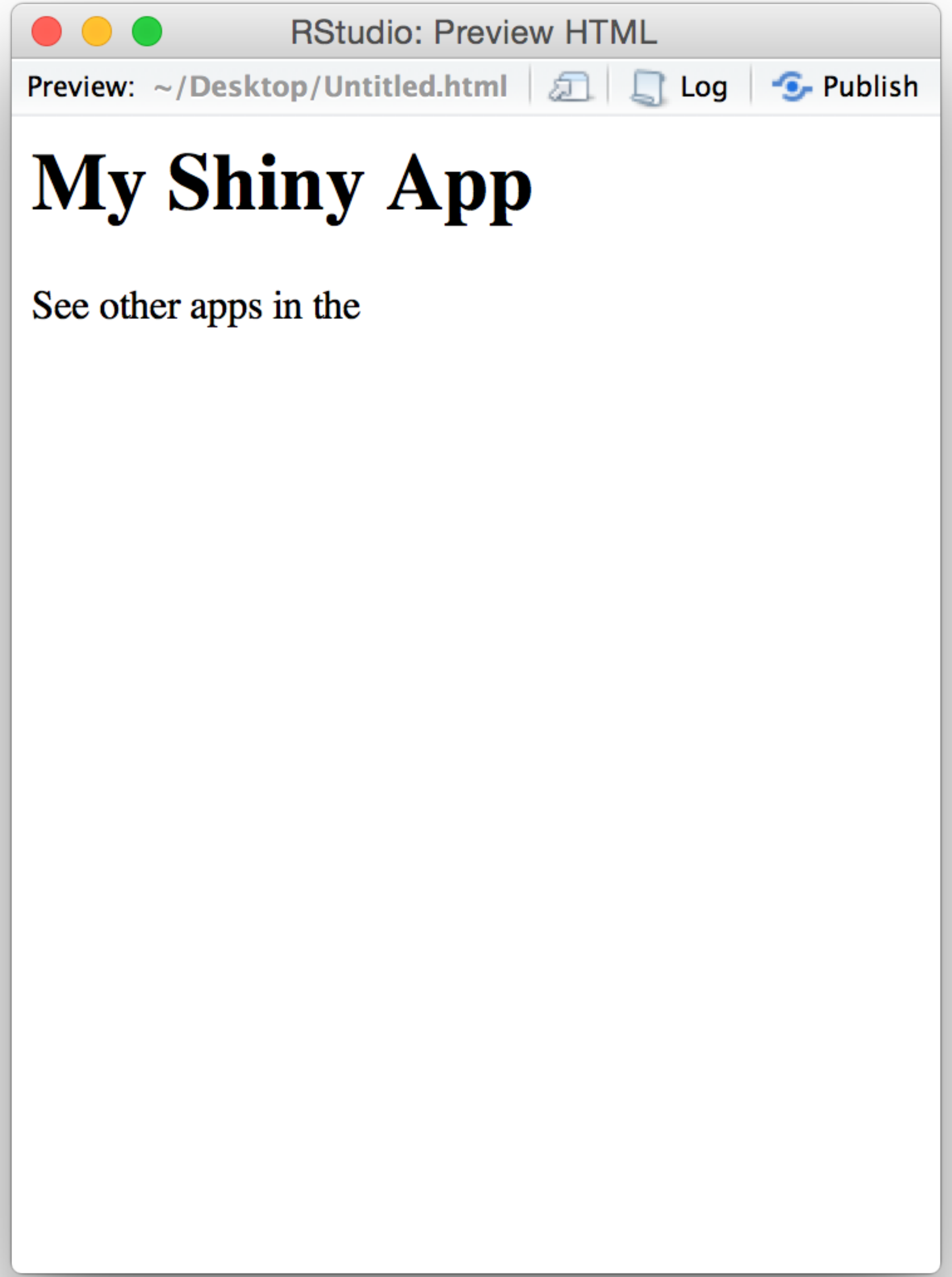

RStudio: Preview HTML
Preview: ~/Desktop/Untitled.html
Log
Publish

```
<div class="container-fluid">
  <h1>My Shiny App</h1>
</div>
```



My Shiny App

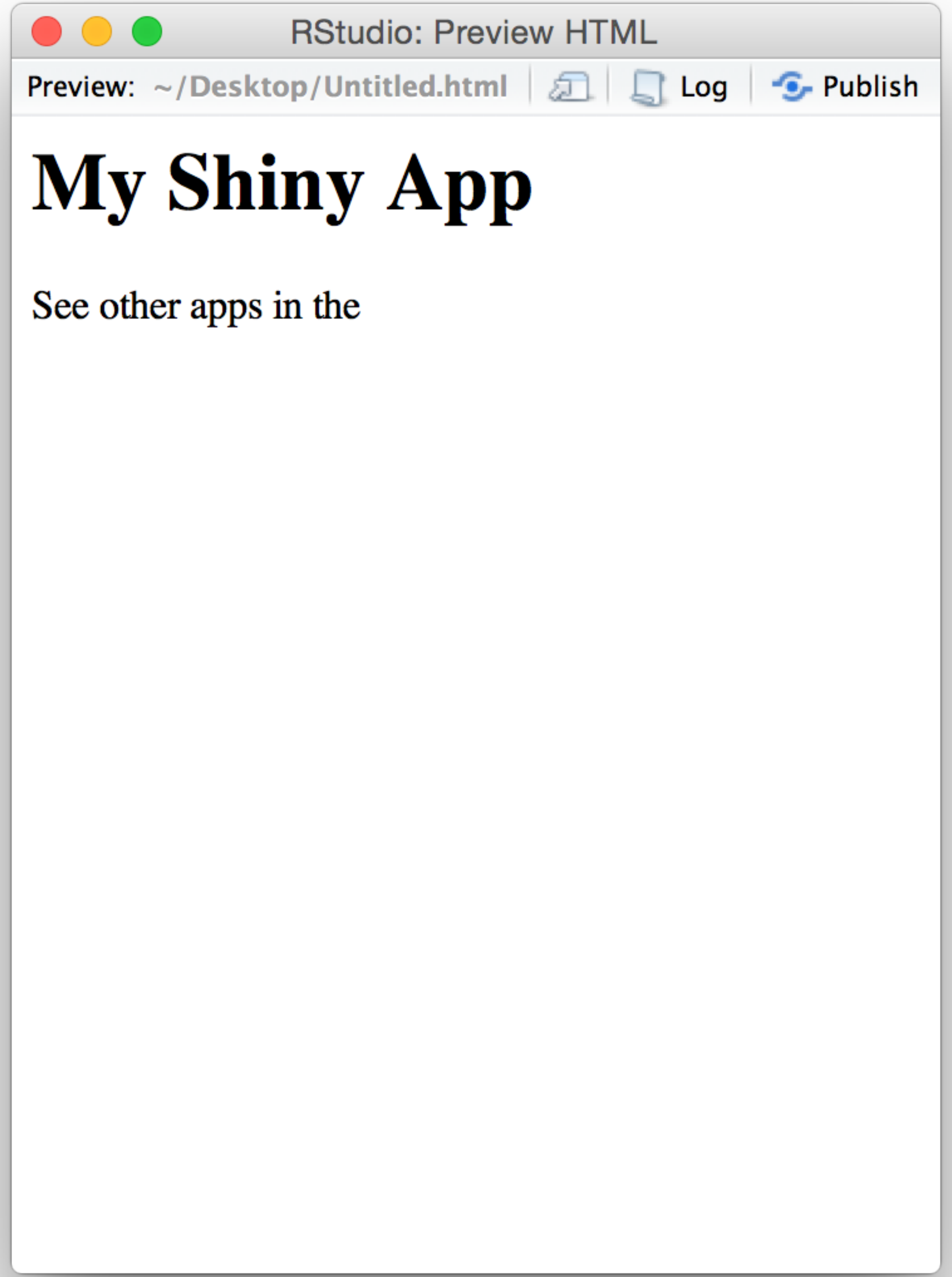
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p>See other apps in the</p>
</div>
```



My Shiny App

See other apps in the

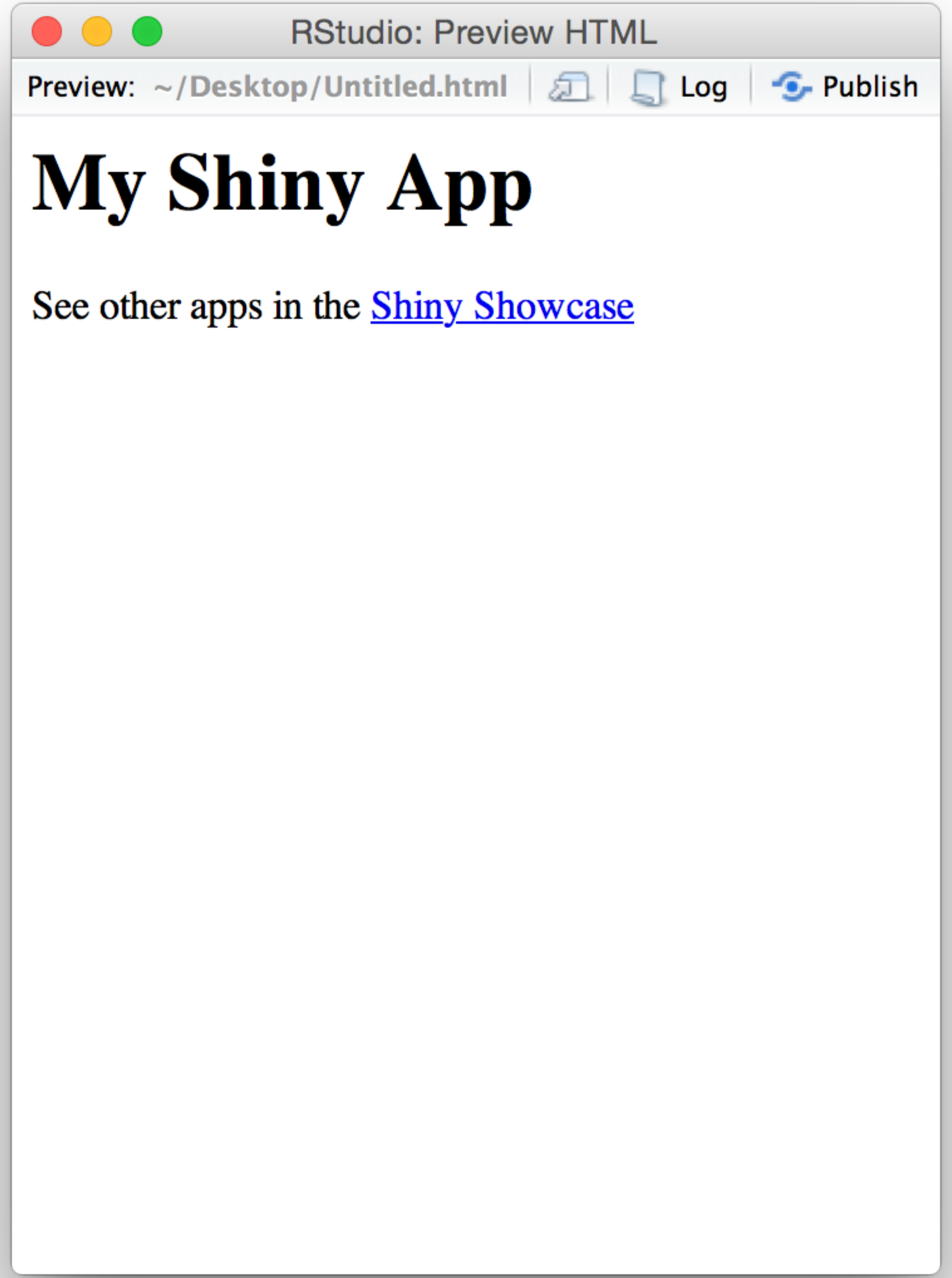
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p>See other apps in the




  </p>
</div>
```



My Shiny App

See other apps in the

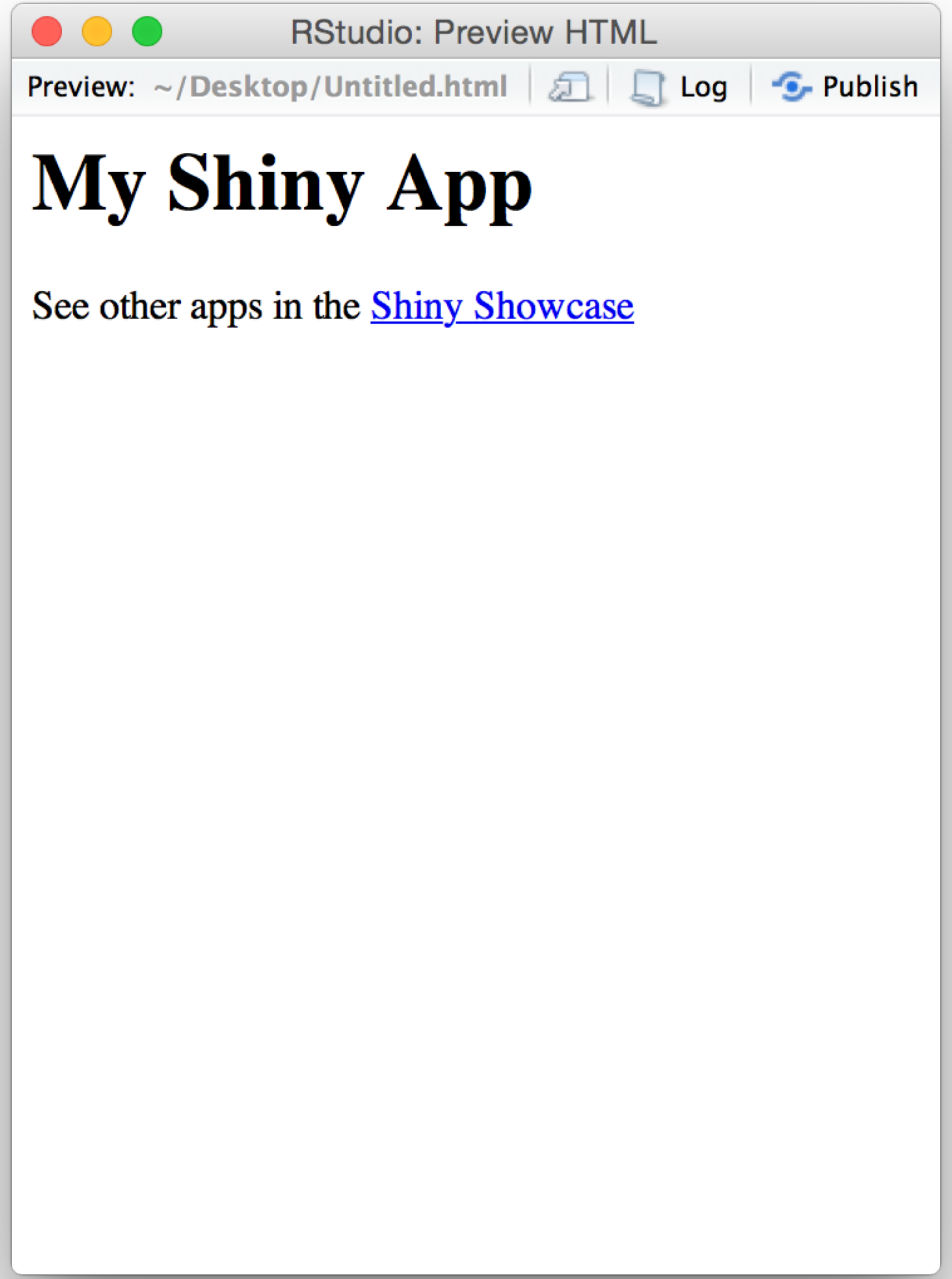
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p>See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



My Shiny App

See other apps in the Shiny Showcase

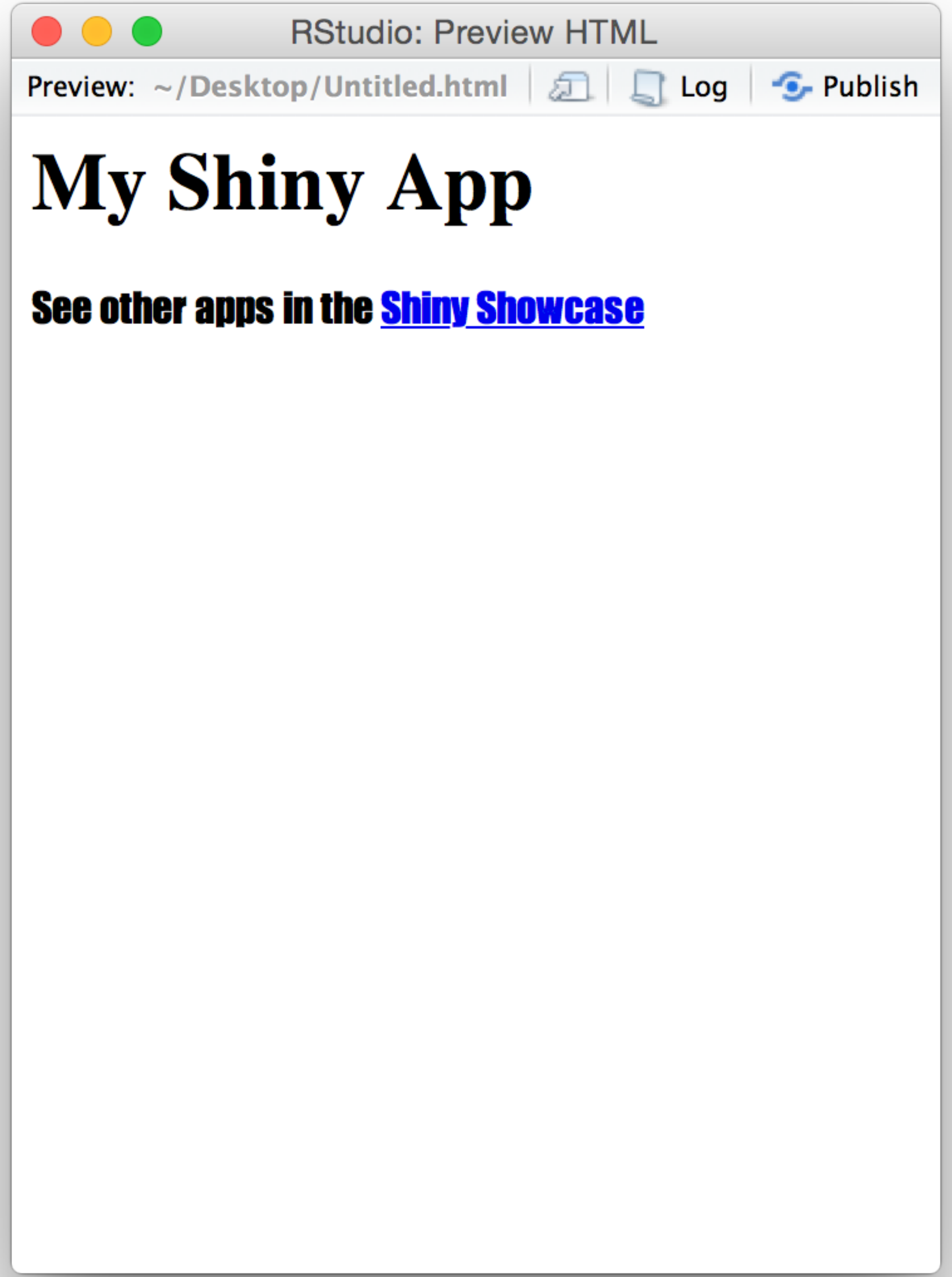
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p>See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



My Shiny App

See other apps in the Shiny Showcase

```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



My Shiny App

See other apps in the Shiny Showcase

How do you add content to a web page?

When writing HTML, you add content with **tags**.

```
<h1></h1>
<a></a>
```

How do you add content to a web page?

When writing R, you add content with **tags** functions.

```
tags$h1() ⟷ <h1></h1>
tags$a() ⟷ <a></a>
```

Shiny HTML tag functions

Shiny provides R functions to recreate HTML tags.

`tags$h1()` ↔ `<h1></h1>`

`tags$a()` ↔ `<a></a>`

tags

Each element of the tags list is a function that recreates an html tag.

```
names(tags)
##  [1] "a"           "abbr"        "address"     "area"       
##  [5] "article"     "aside"       "audio"       "b"          
##  [9] "base"        "bdi"         "bdo"         "blockquote" 
## [13] "body"        "br"          "button"      "canvas"     
## [17] "caption"     "cite"        "code"        "col"        
## [21] "colgroup"    "command"     "data"        "datalist"   
## [25] "dd"          "del"         "details"     "dfn"        
## [29] "div"         "dl"          "dt"          "em"         
## [33] "embed"       "eventsource" "fieldset"    "figcaption" 
## [37] "figure"      "footer"      "form"        "h1"         
## [41] "h2"          "h3"          "h4"          "h5"         
```

A list of functions

```
tags$h1
```

```
function (...)
tag("h1", list(...))
<environment: namespace:htmltools>
```

```
tags$h1()
```

```
<h1></h1>
```

tags syntax

```
tags$a(href = "www.rstudio.com", "RStudio")
```

the list named tags

the function/tag name (followed by parentheses)

named arguments appear as tag attributes (set boolean attributes to NA)

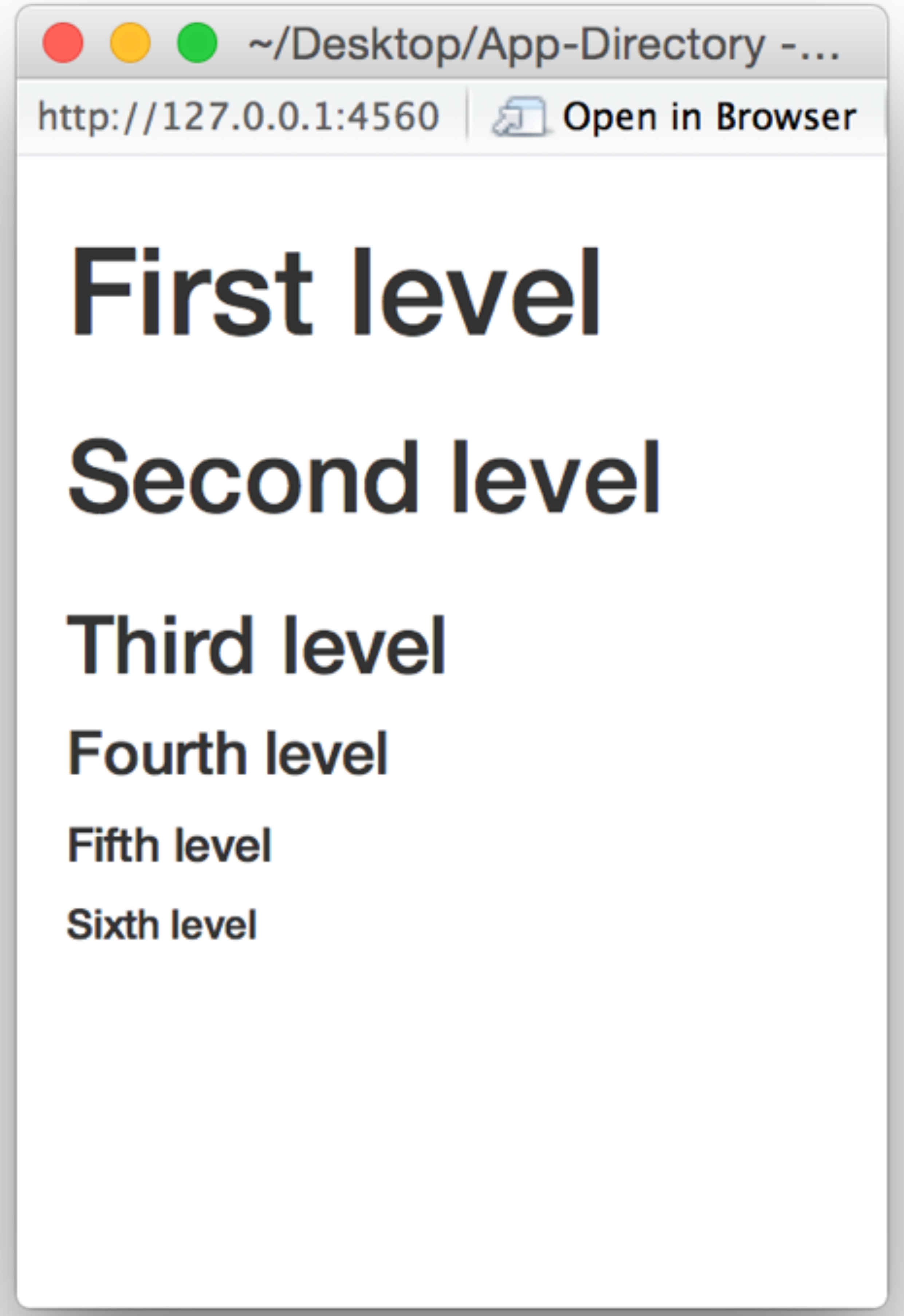
unnamed arguments appear inside the tags (call tags$...() to create nested tags)

```
<a href="www.rstudio.com">RStudio</a>
```

h1() - h6()

Headers

```
fluidPage(
   tags$h1("First level"),
   tags$h2("Second level"),
   tags$h3("Third level"),
   tags$h4("Fourth level"),
   tags$h5("Fifth level"),
  tags$h6("Sixth level")
)
```



First level

Second level

Third level

Fourth level

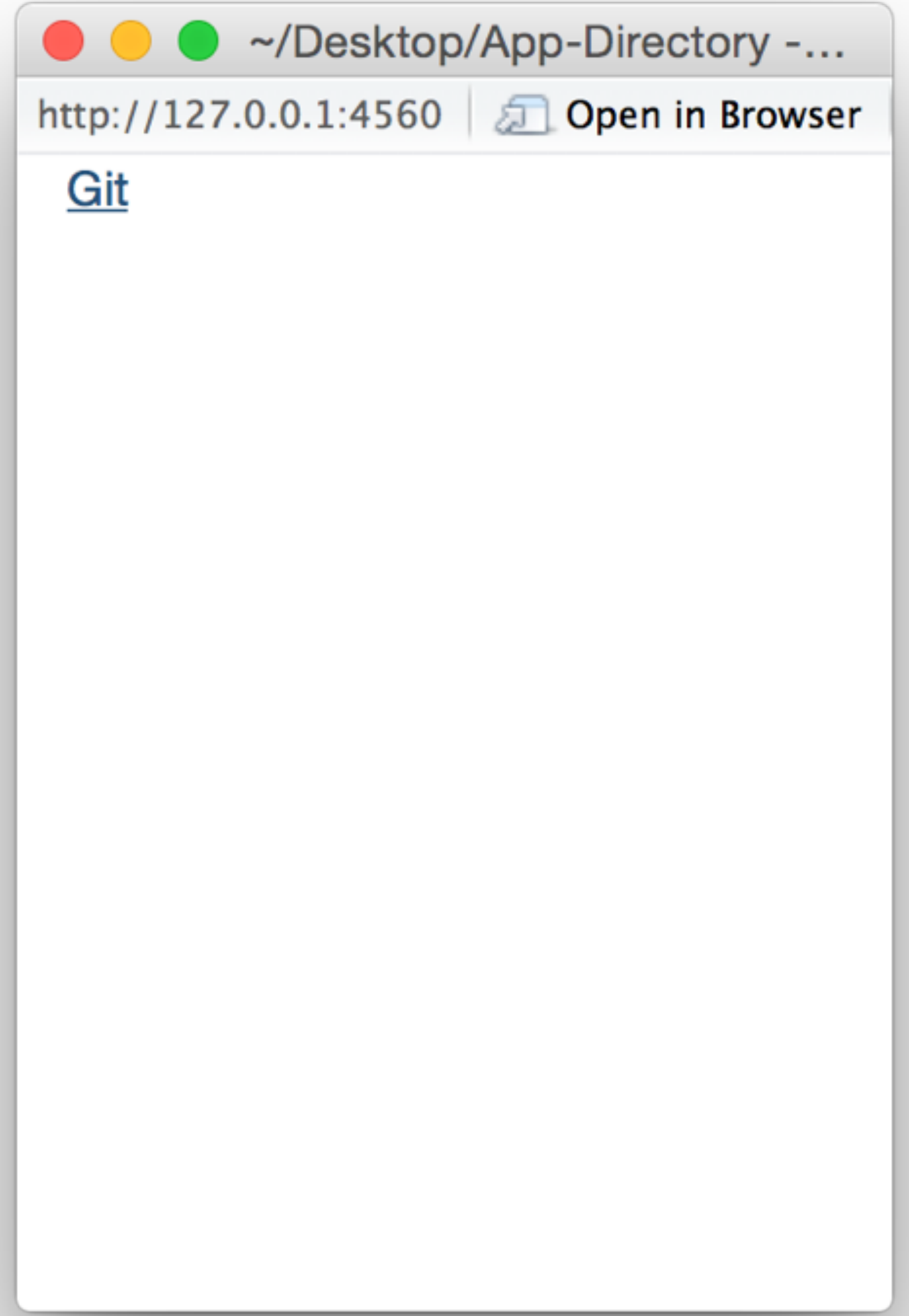
Fifth level

Sixth level

a()

hyperlink with the href argument

```
fluidPage(
  tags$a(href= "http://www.git.com",
    "Git")
)
```

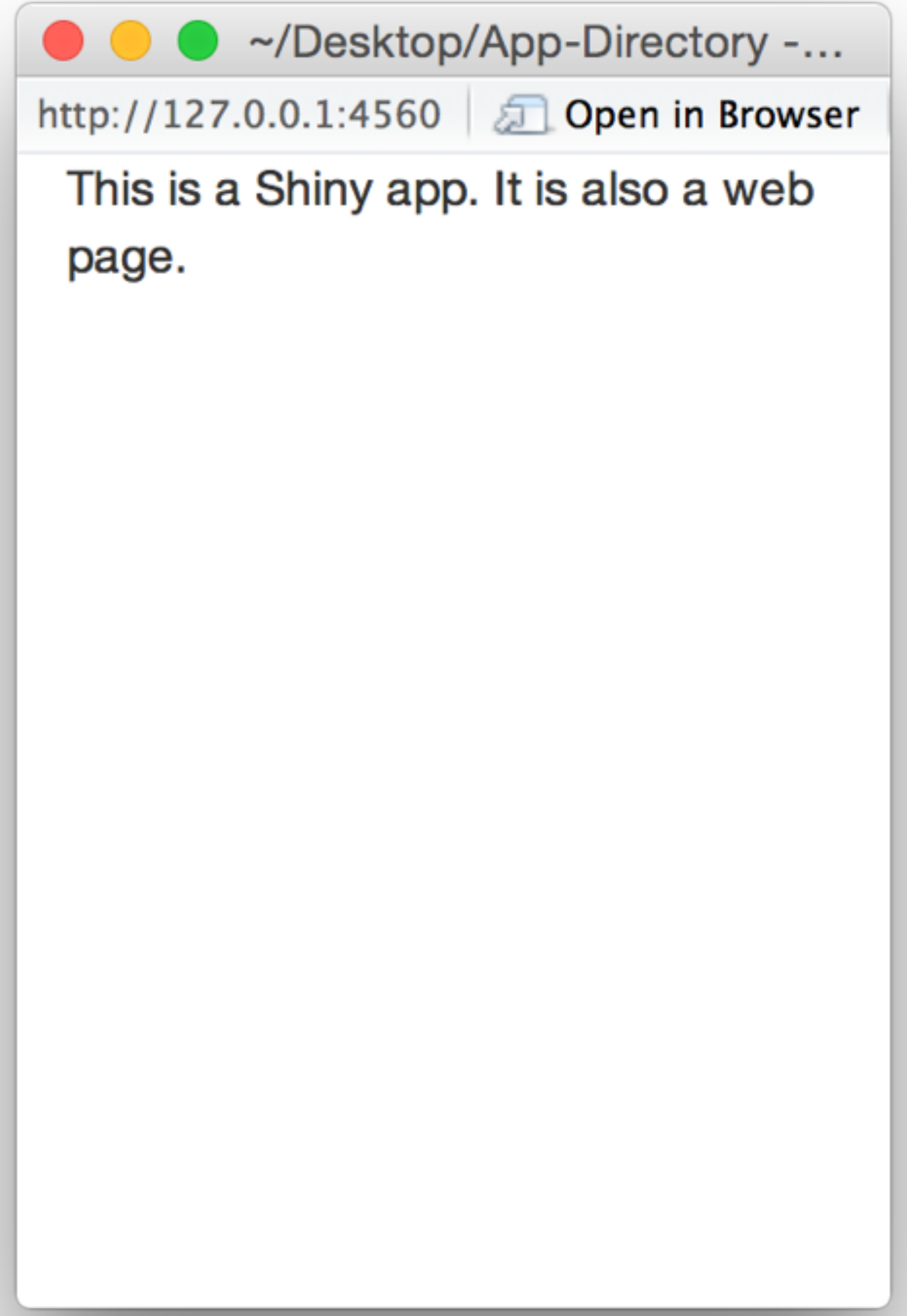


Git

text

Character strings do not need a tag.

```
fluidPage(
 "This is a Shiny app.",
 "It is also a web page."
)
```

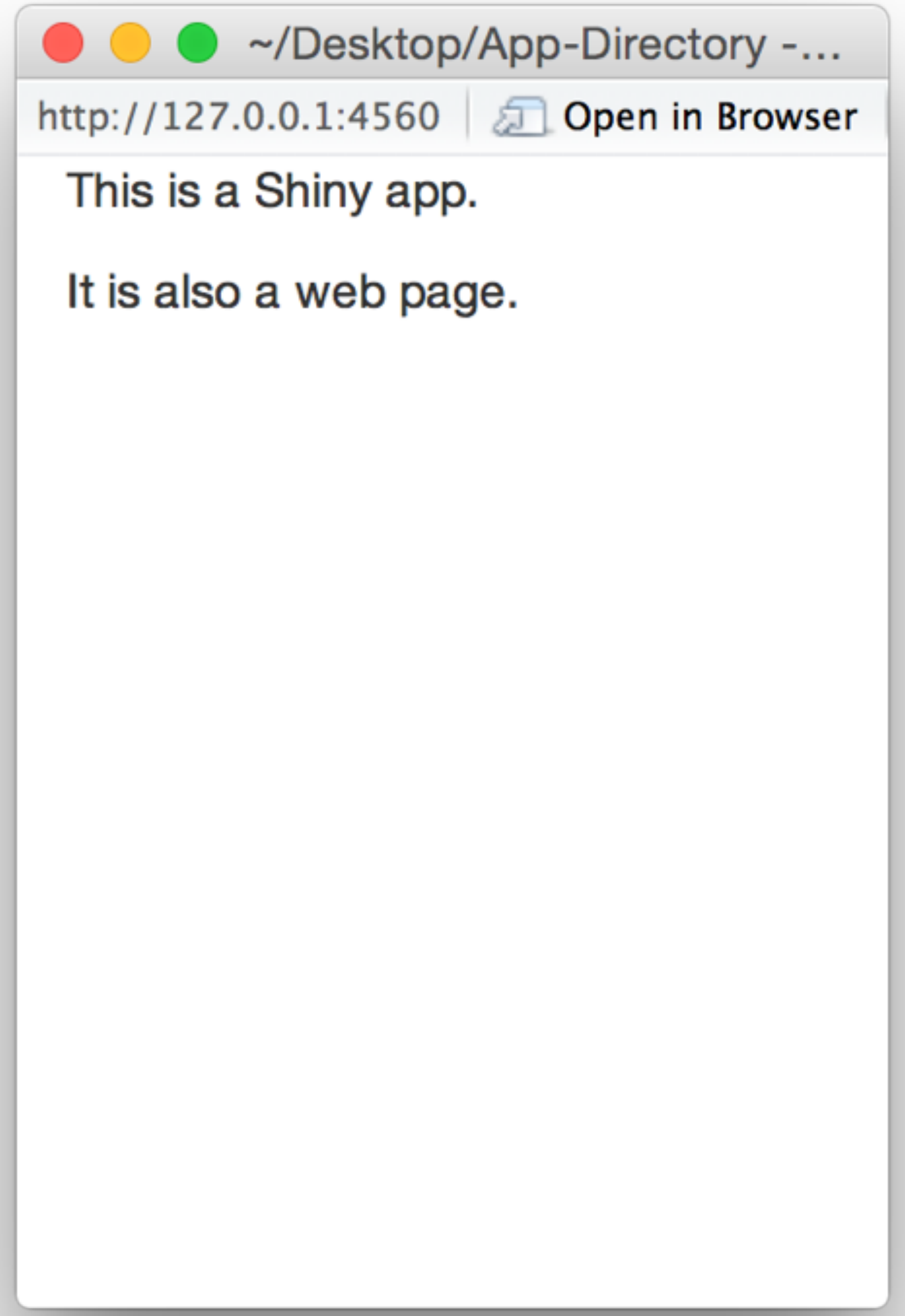


This is a Shiny app. It is also a web page.

p()

A new paragraph

```
fluidPage(
 tags$p("This is a Shiny app."),
 tags$p("It is also a web page.")
)
```



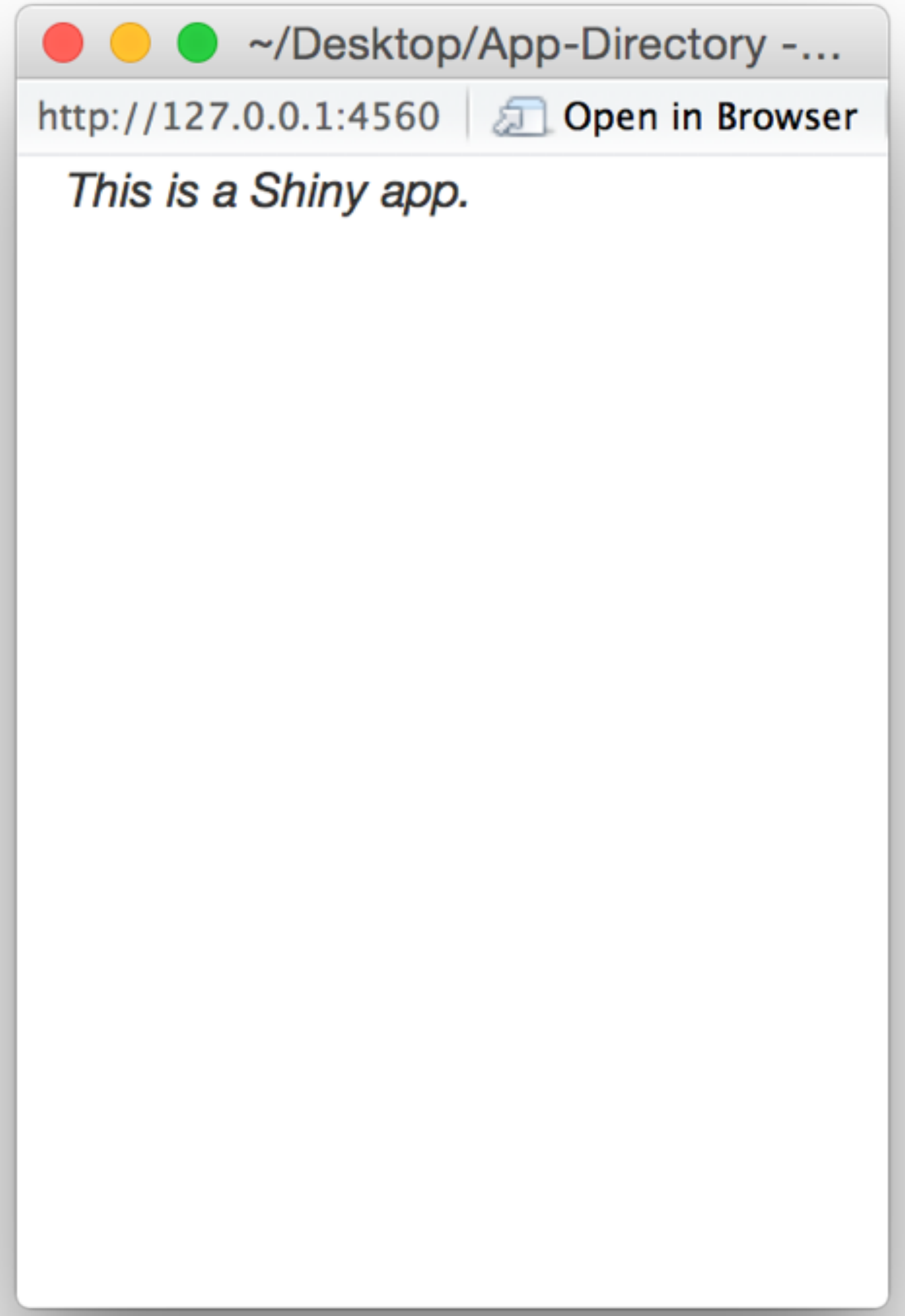
This is a Shiny app.

It is also a web page.

em()

Emphasized (*italic*) text

```
fluidPage(
  tags$em("This is a Shiny app.")
)
```

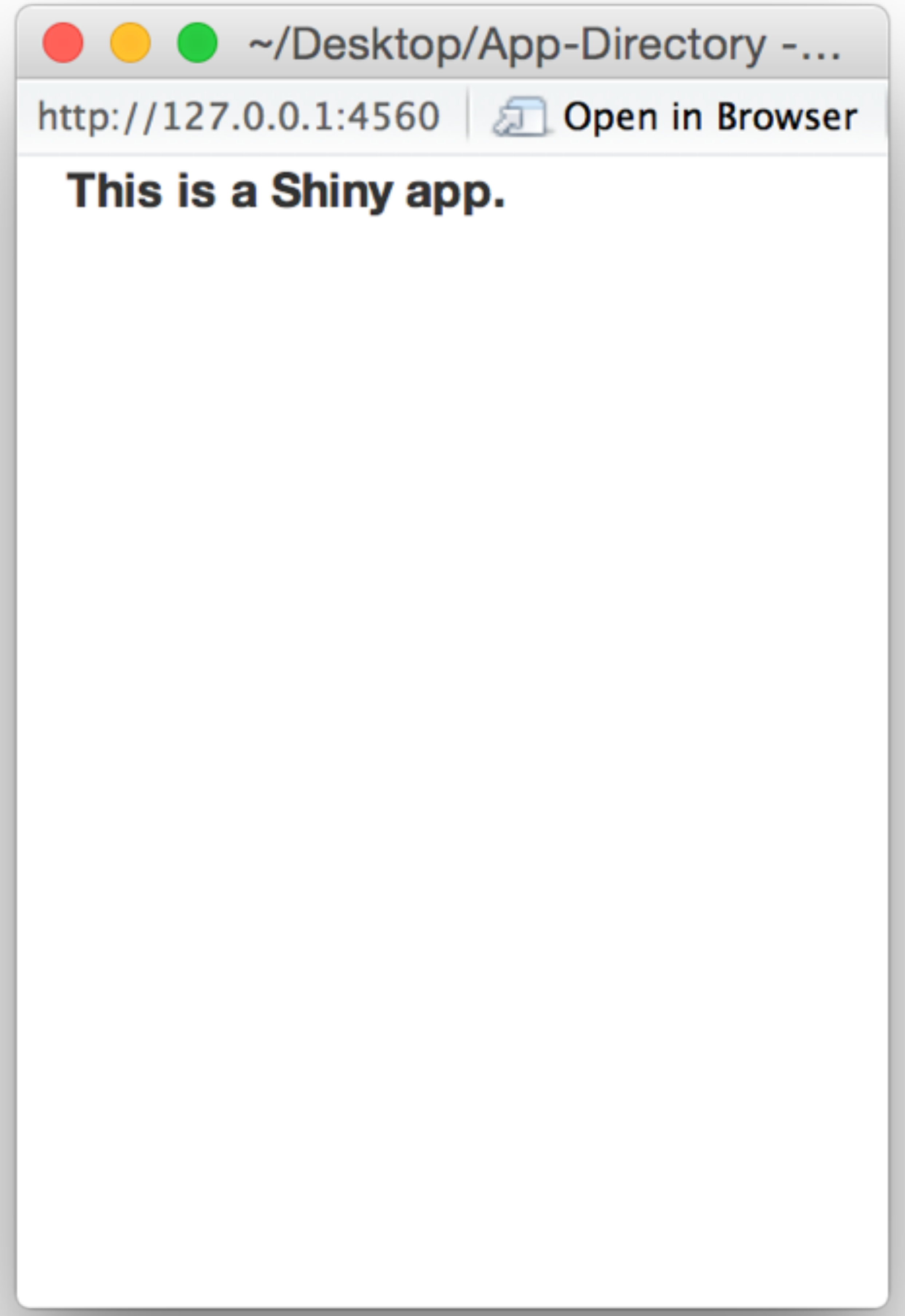


This is a Shiny app.

strong()

Strong (**bold**) text

```
fluidPage(
  tags$strong("This is a Shiny app.")
)
```

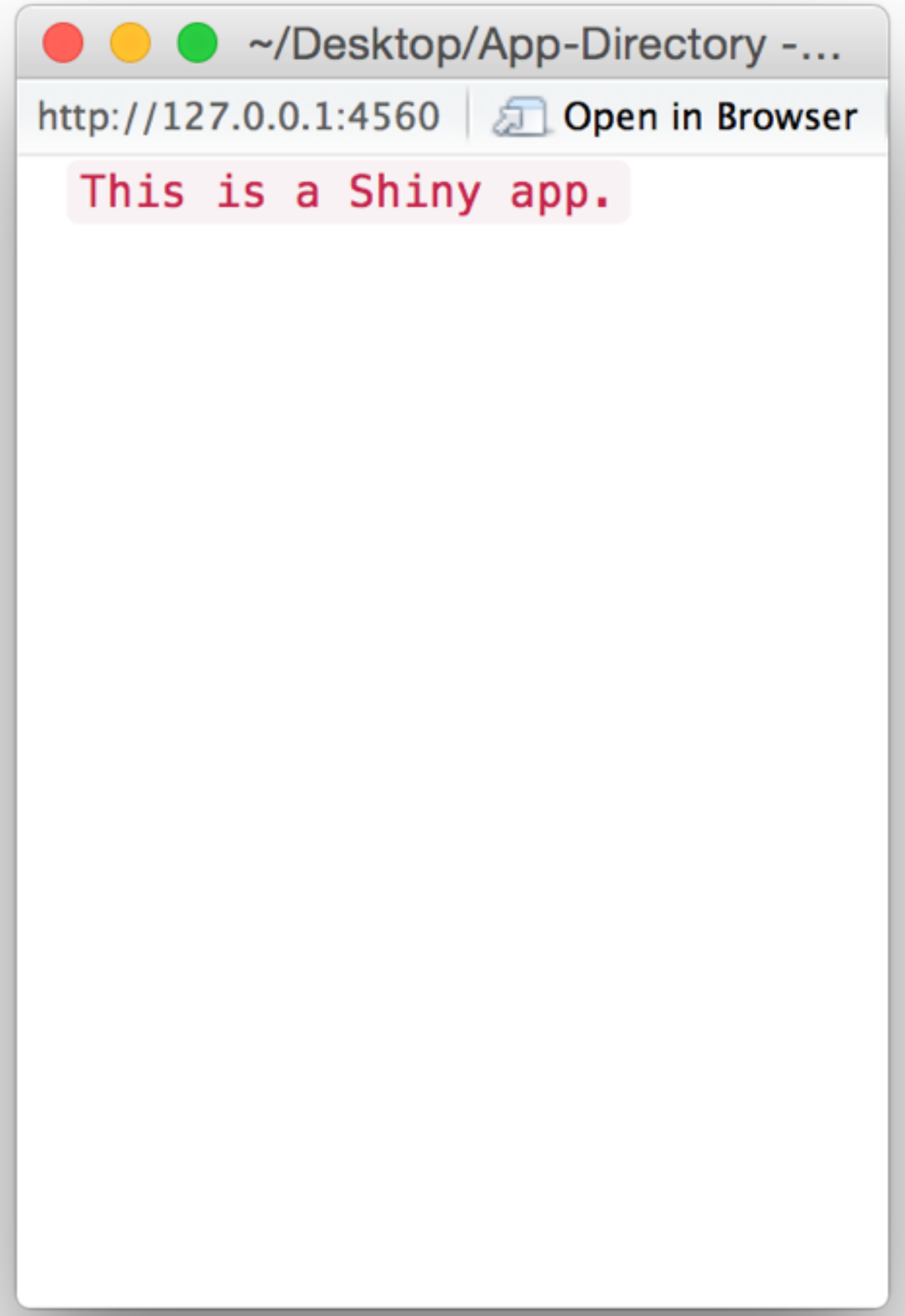


This is a Shiny app.

code()

Monospaced text (code)

```
fluidPage(
  tags$code("This is a Shiny app.")
)
```

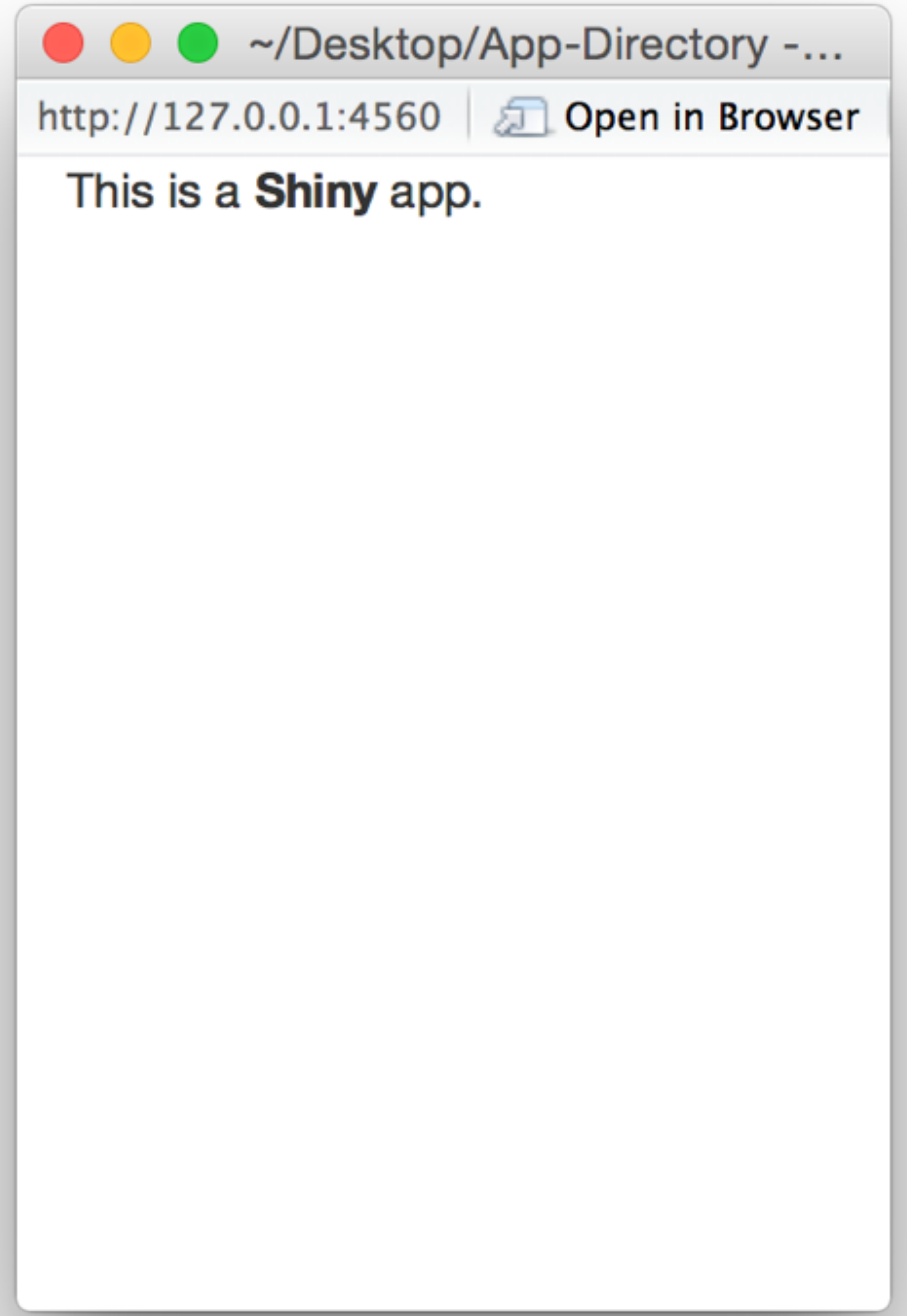


```
This is a Shiny app.
```

nesting

You can also nest functions inside of others

```
fluidPage(
  tags$p("This is a",
    tags$strong("Shiny"),
    "app.")
)
```

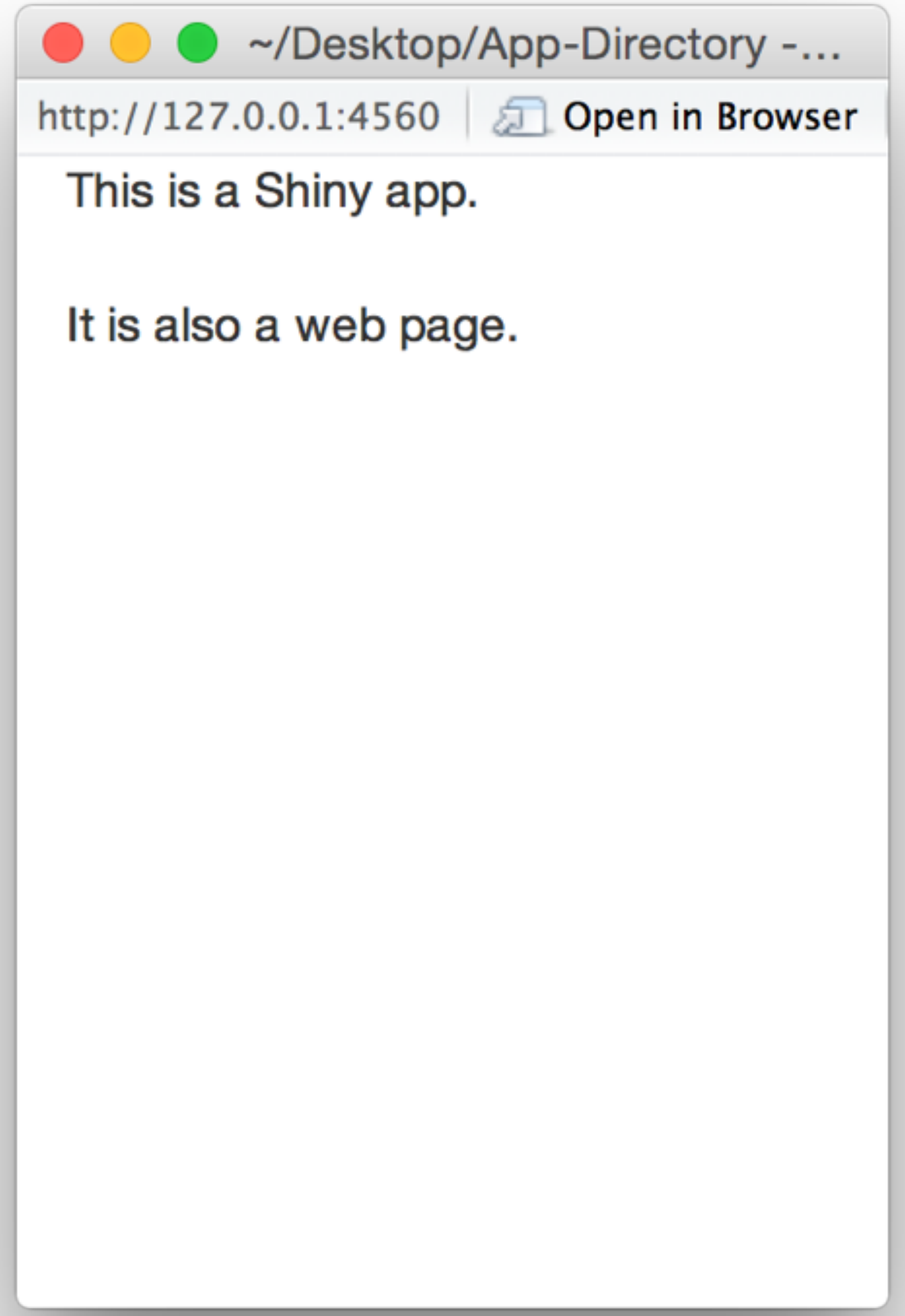


This is a **Shiny** app.

br()

A line break

```
fluidPage(
  "This is a Shiny app.",
  tags$br(),
  "It is also a web page."
)
```



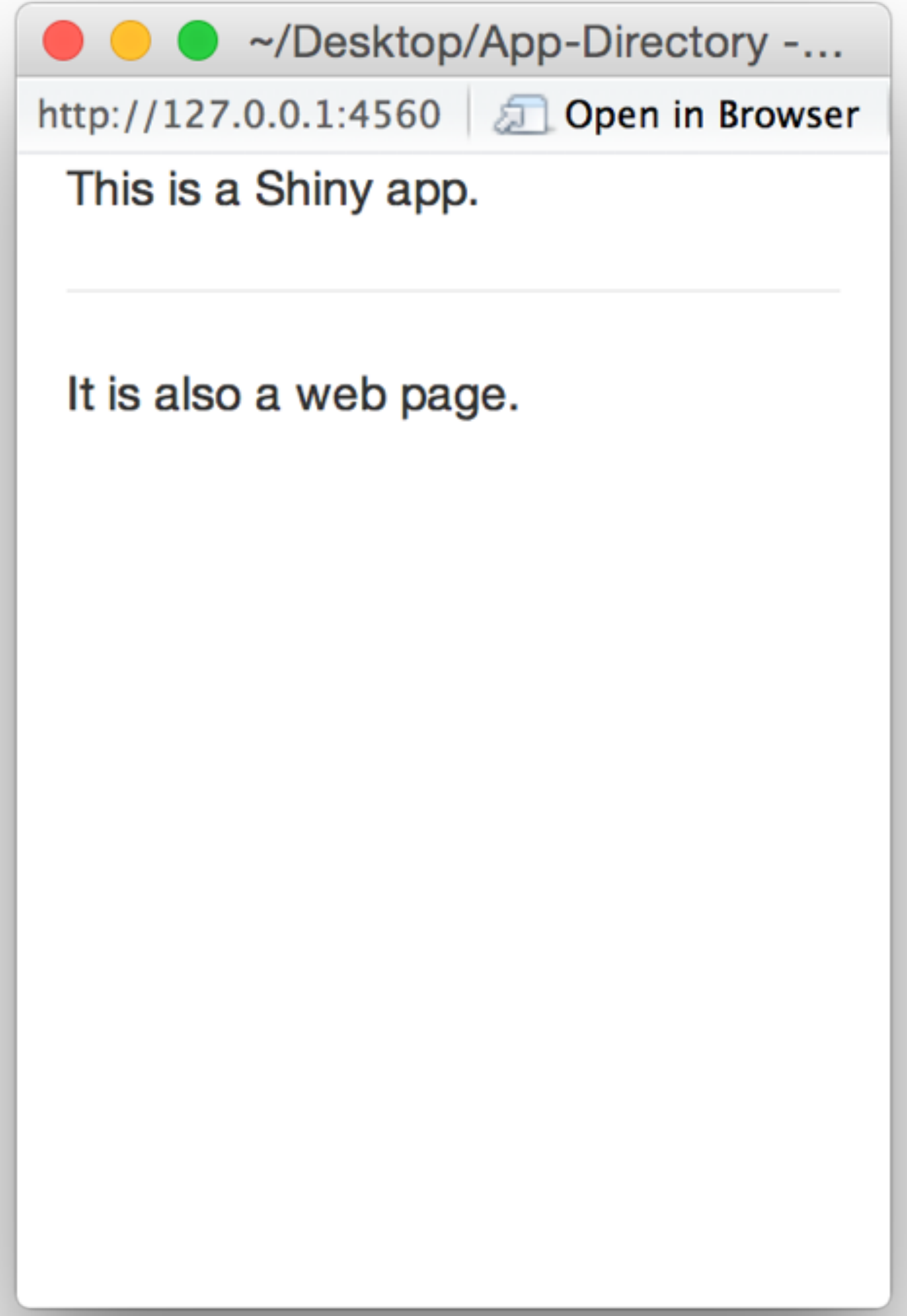
This is a Shiny app.

It is also a web page.

hr()

A horizontal rule

```
fluidPage(
  "This is a Shiny app.",
  tags$hr(),
  "It is also a web page."
)
```



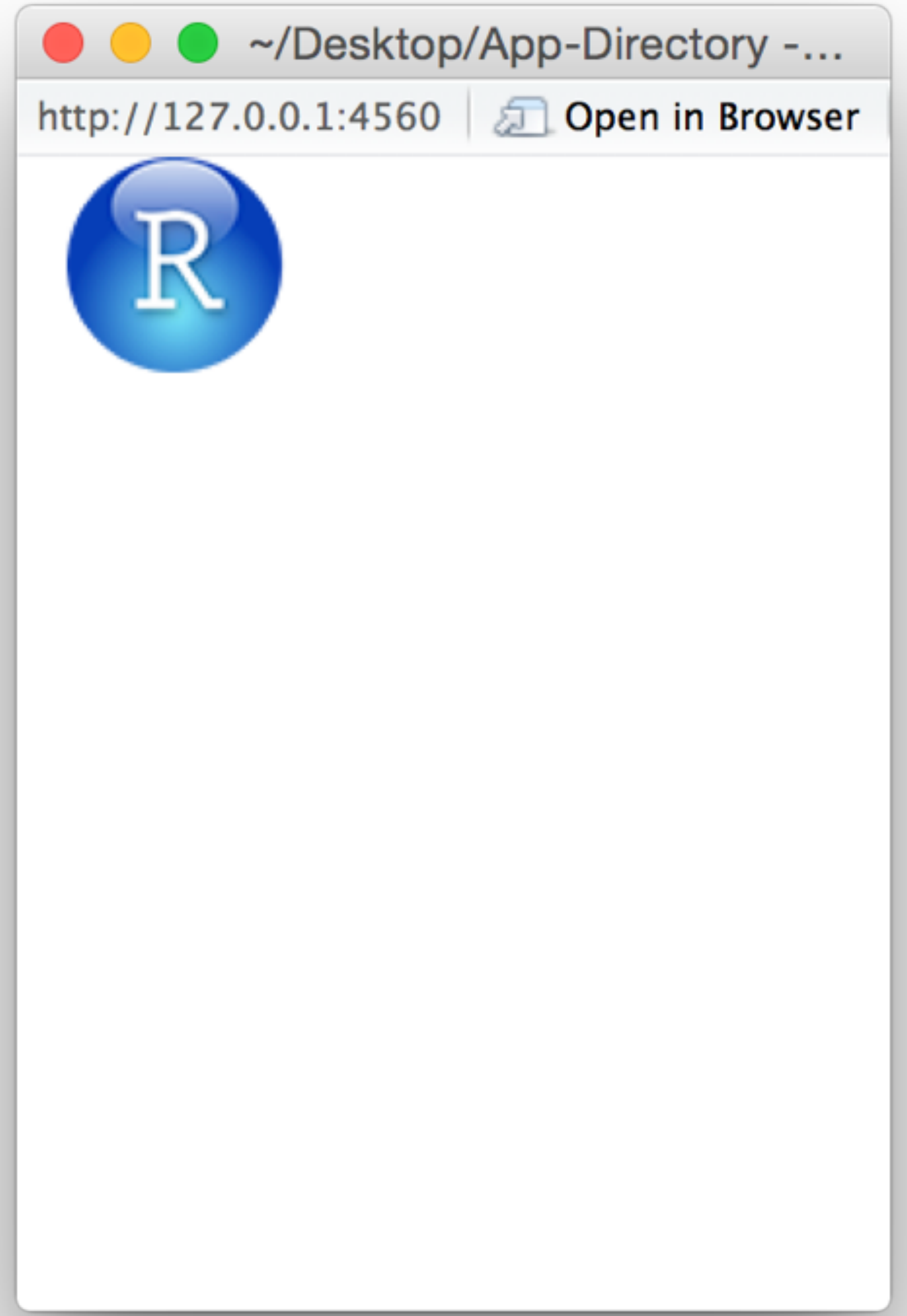
This is a Shiny app.

It is also a web page.

img()

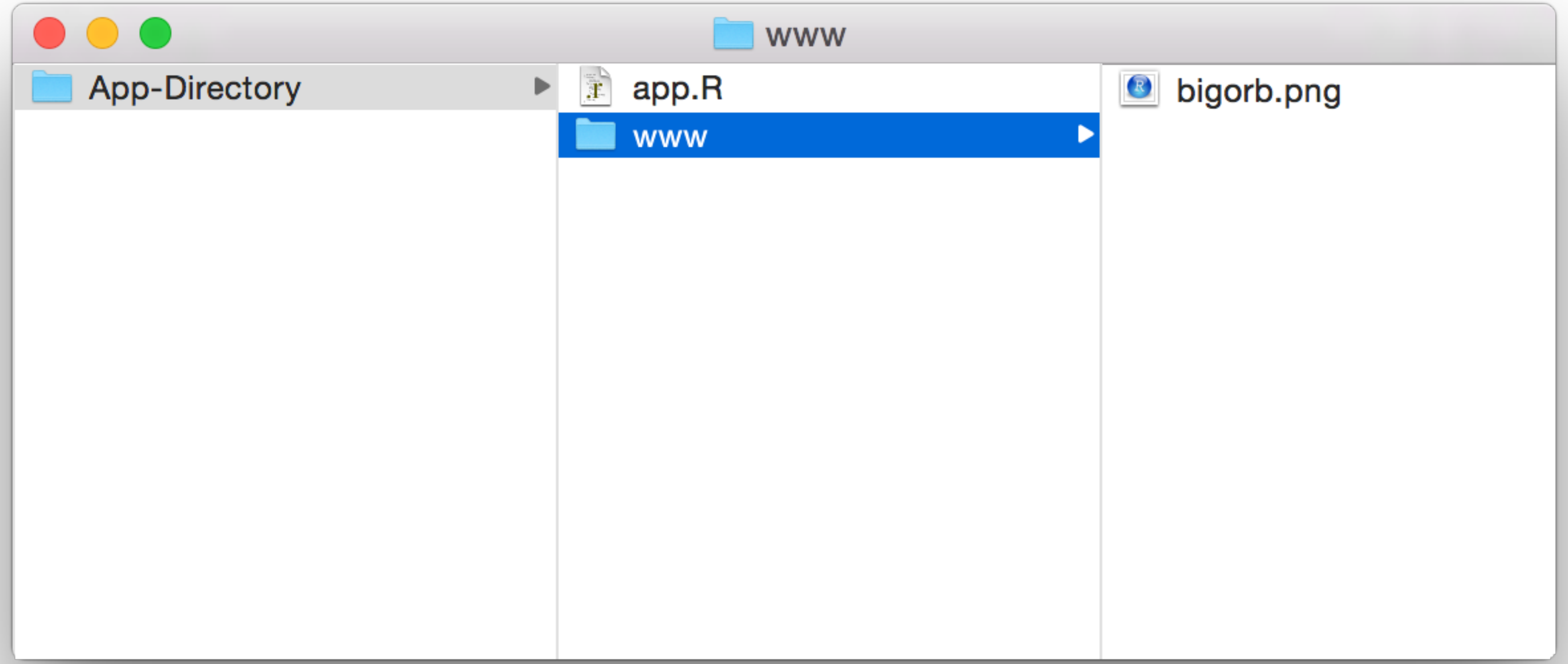
Add an image. Use **src** argument to point to the image URL.

```
fluidPage(
  tags$img(height = 100,
        width = 100,
        src = "http://www.rstudio.com/
               images/RStudio.2x.png")
)
```



Adding Images

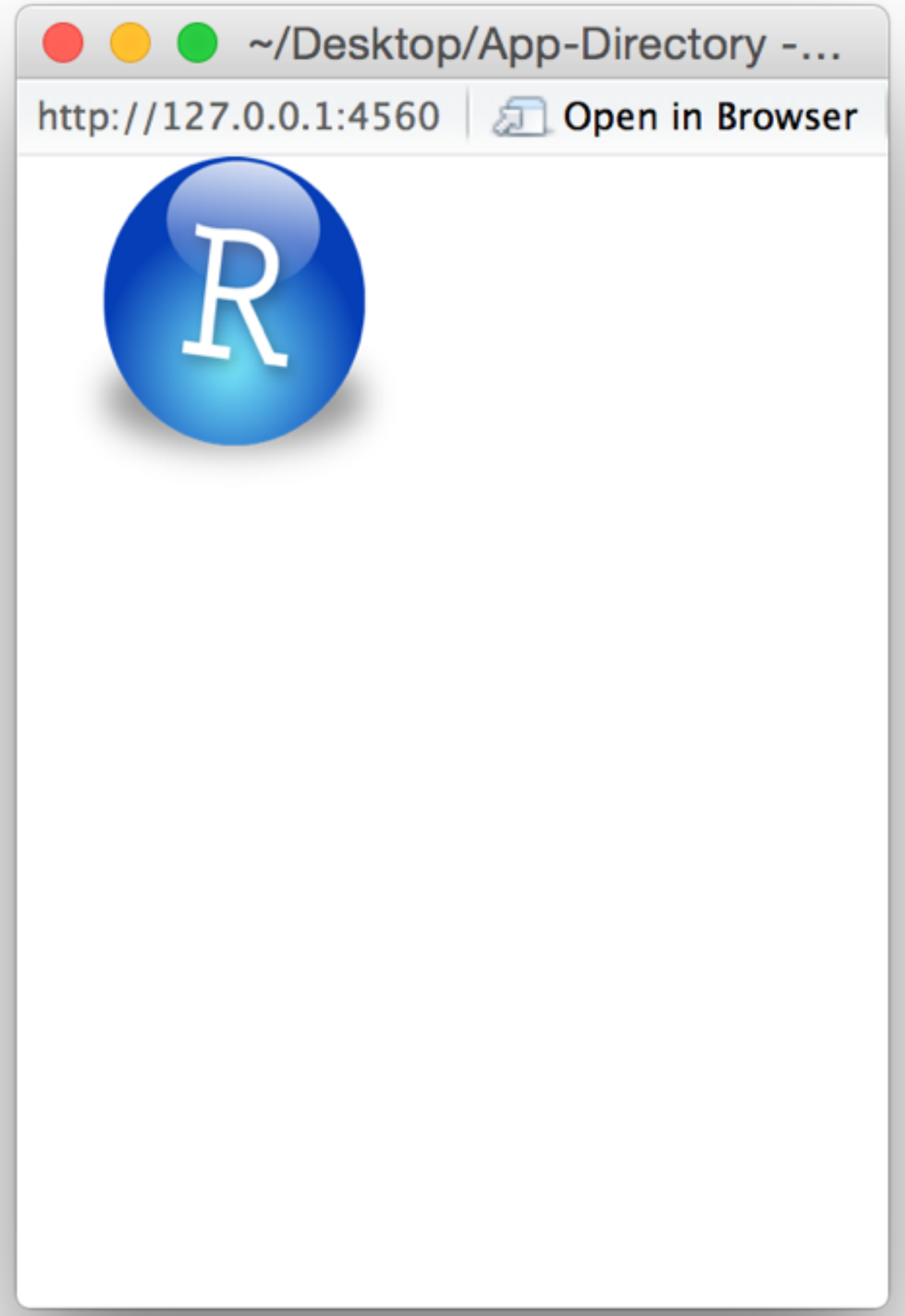
To add an image from a file, save the file in a subdirectory named **www**



img()

Add an img from a file in www

```
fluidPage(
  tags$img(height = 100,
           width = 100,
           src = "bigorb.png")
)
```



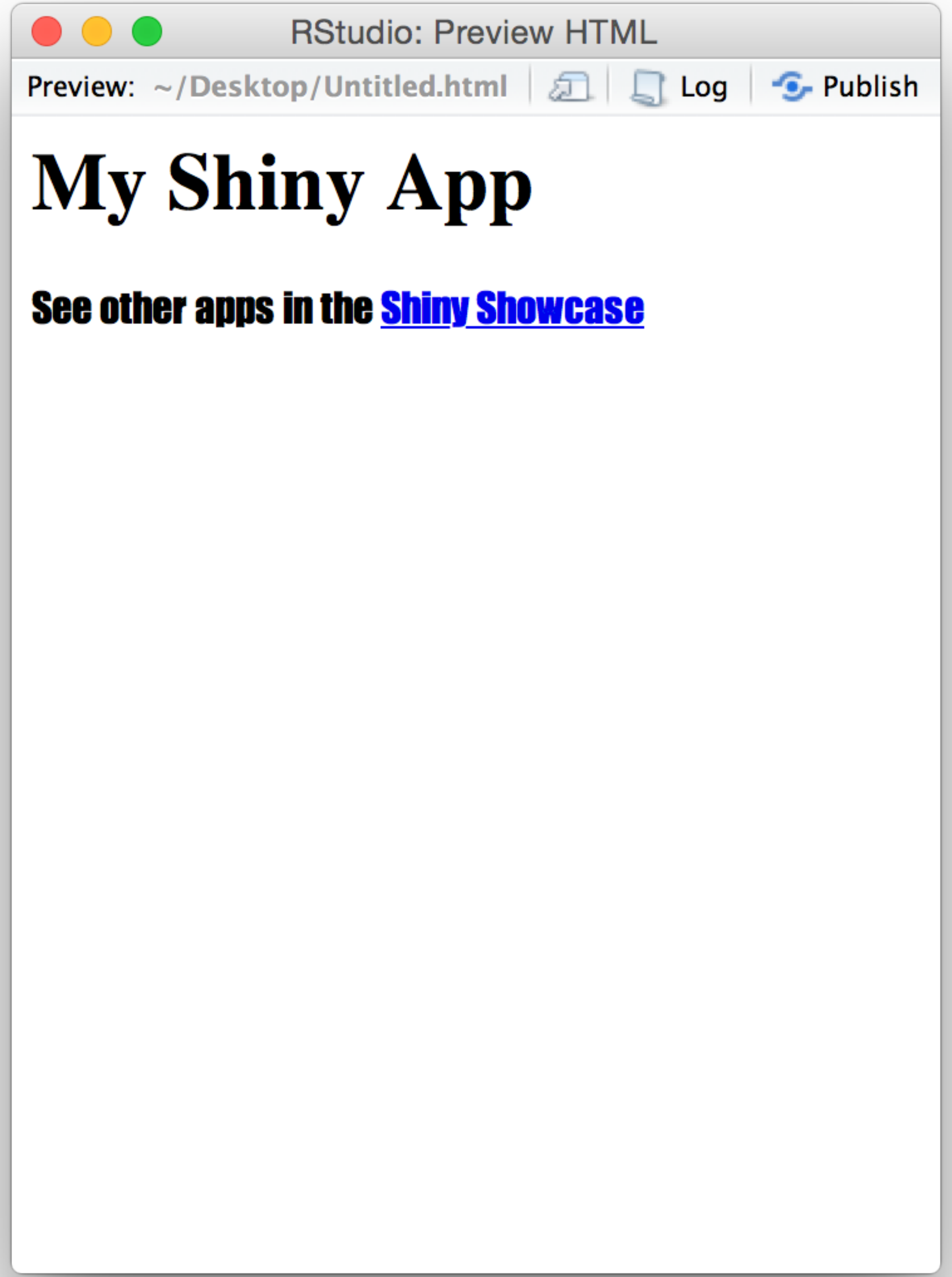
Some tags functions come with a wrapper function, so you do not need to call tags$

```
h1
```

```
function (...)
tags$h1(...)
<environment: namespace:htmltools>
```

Function	Creates
`a()`	A Hyperlink
`br()`	A line break
`code()`	Text formatted like computer code
`em()`	Italicized (emphasized) text
`h1(),h2(),h3(),h4(),h5(),h6()`	Headers (First level to sixth)
`hr()`	A horizontal rule (line)
`img()`	An image
`p()`	A new paragraph
`strong()`	Bold (strong) text

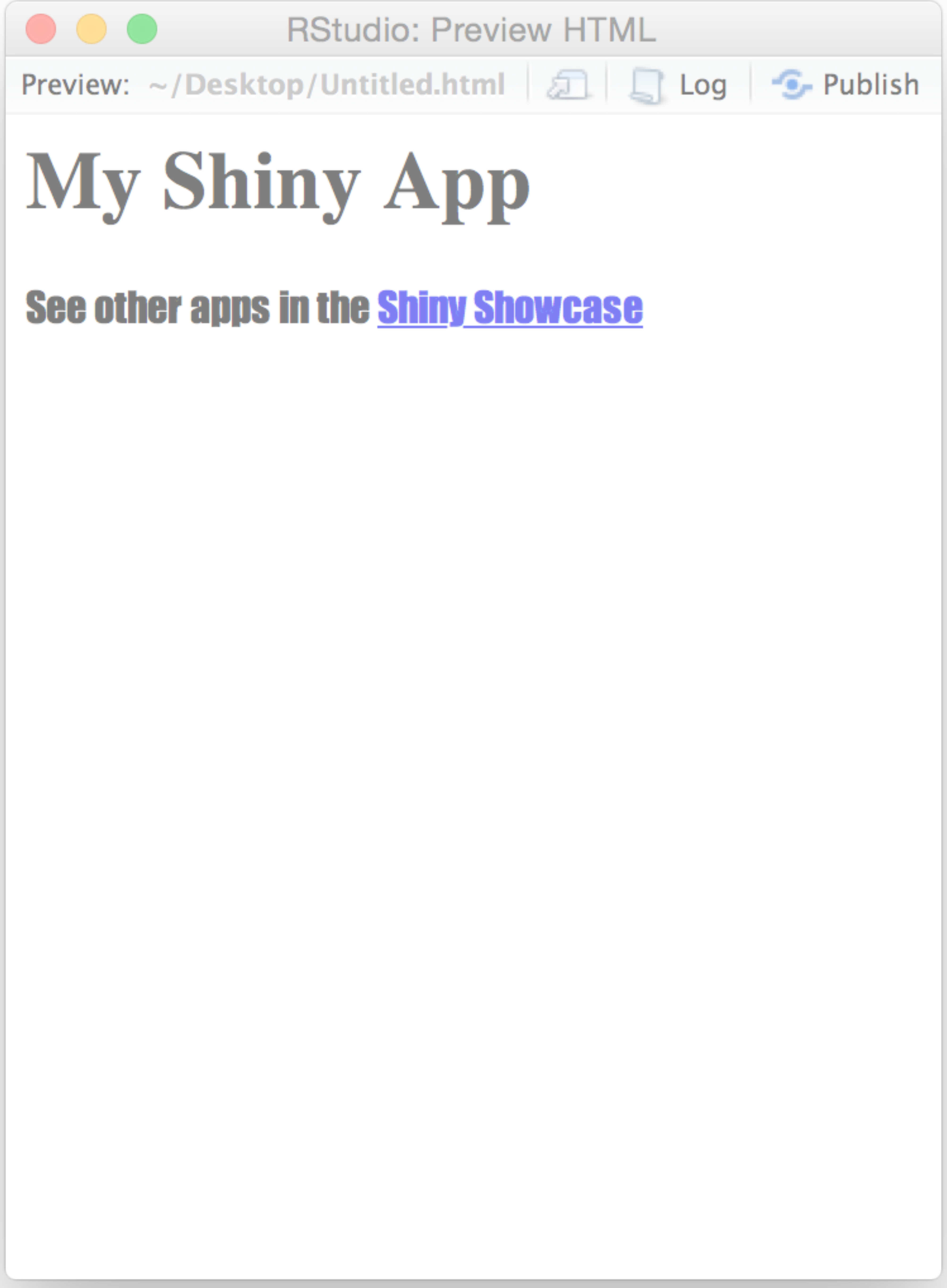
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



My Shiny App

See other apps in the [Shiny Showcase]

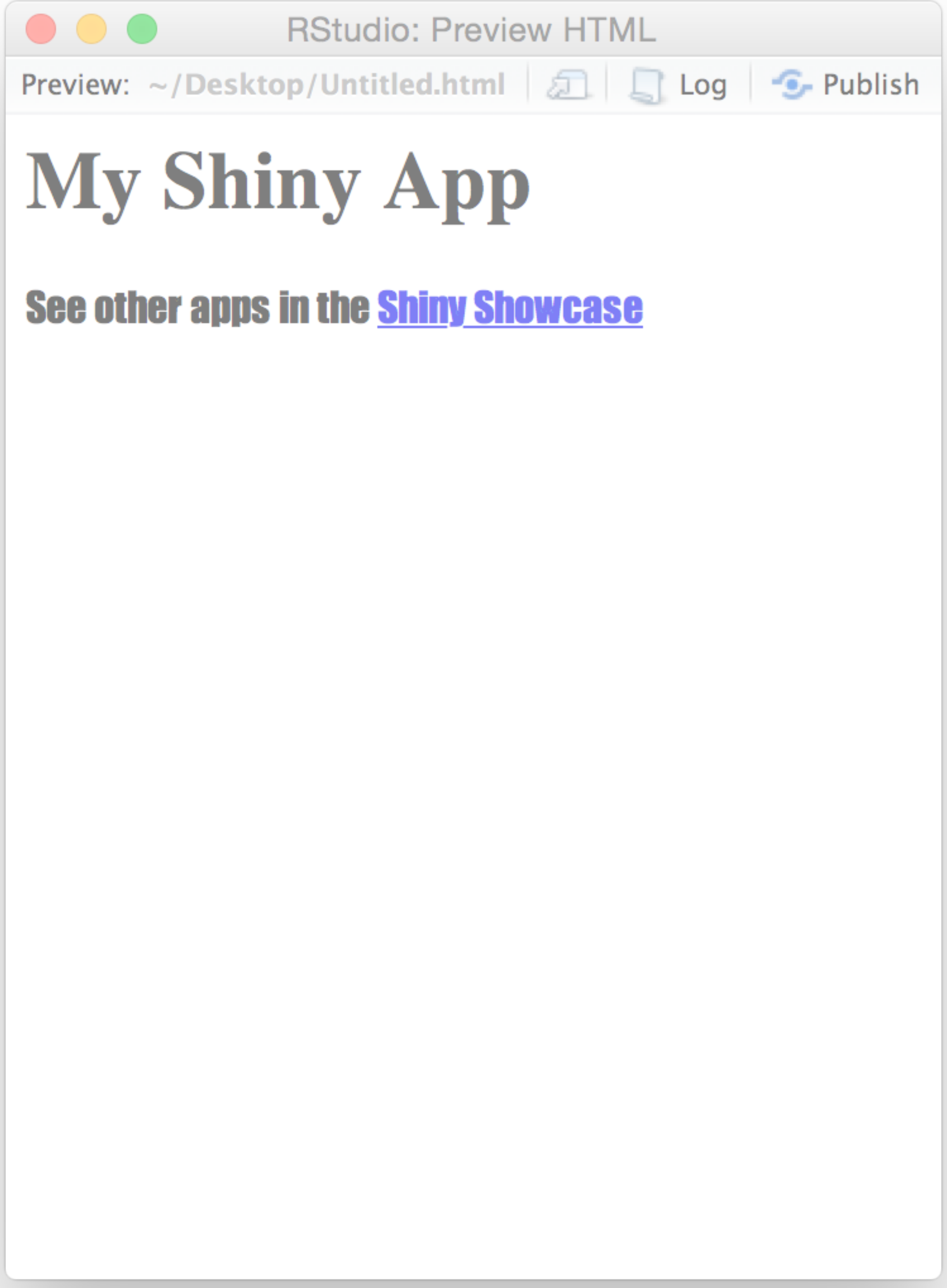
```
fluidPage(
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



My Shiny App

See other apps in the Shiny Showcase

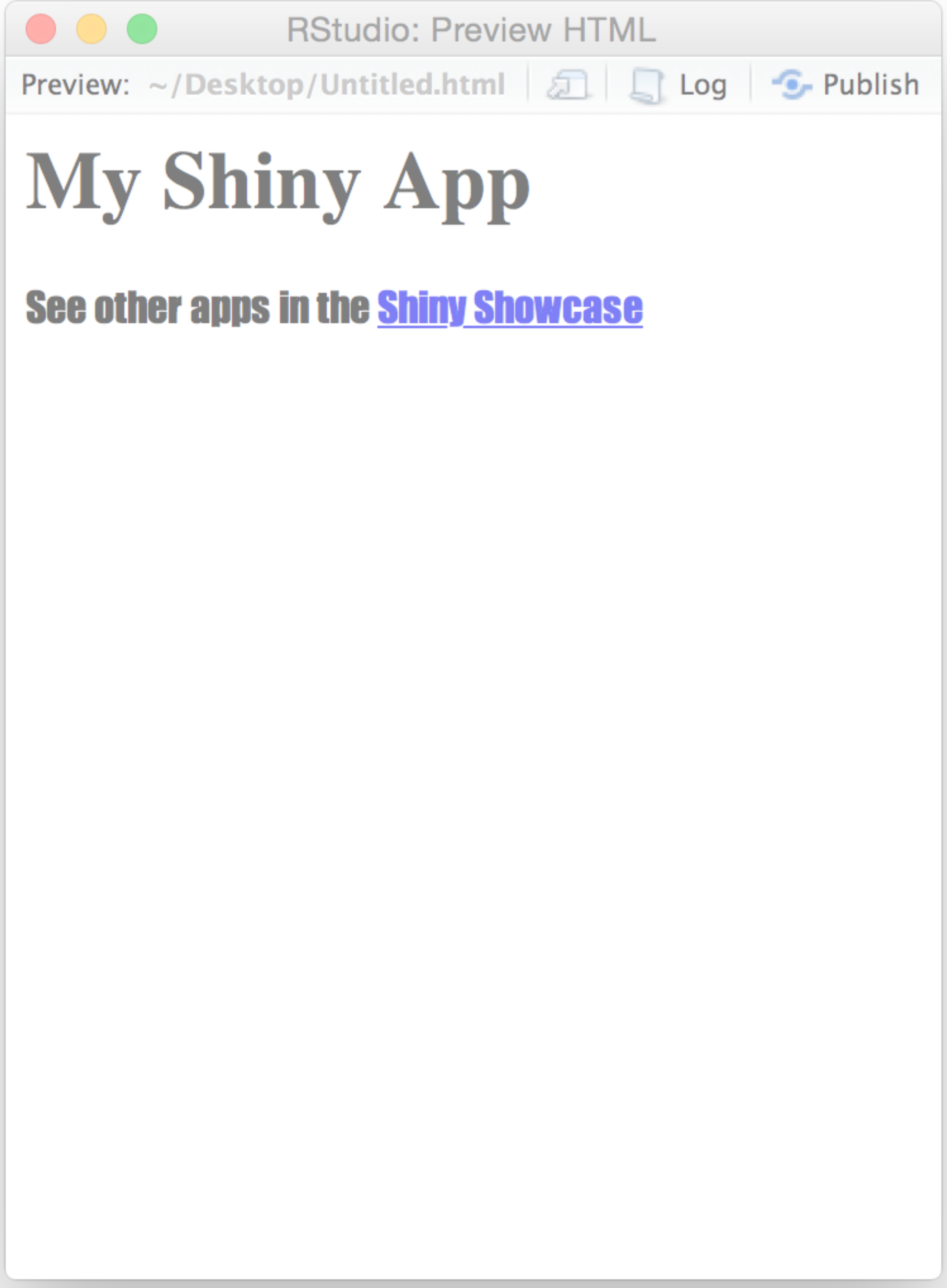
```
fluidPage(
  h1("My Shiny App"),
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



My Shiny App

See other apps in the Shiny Showcase

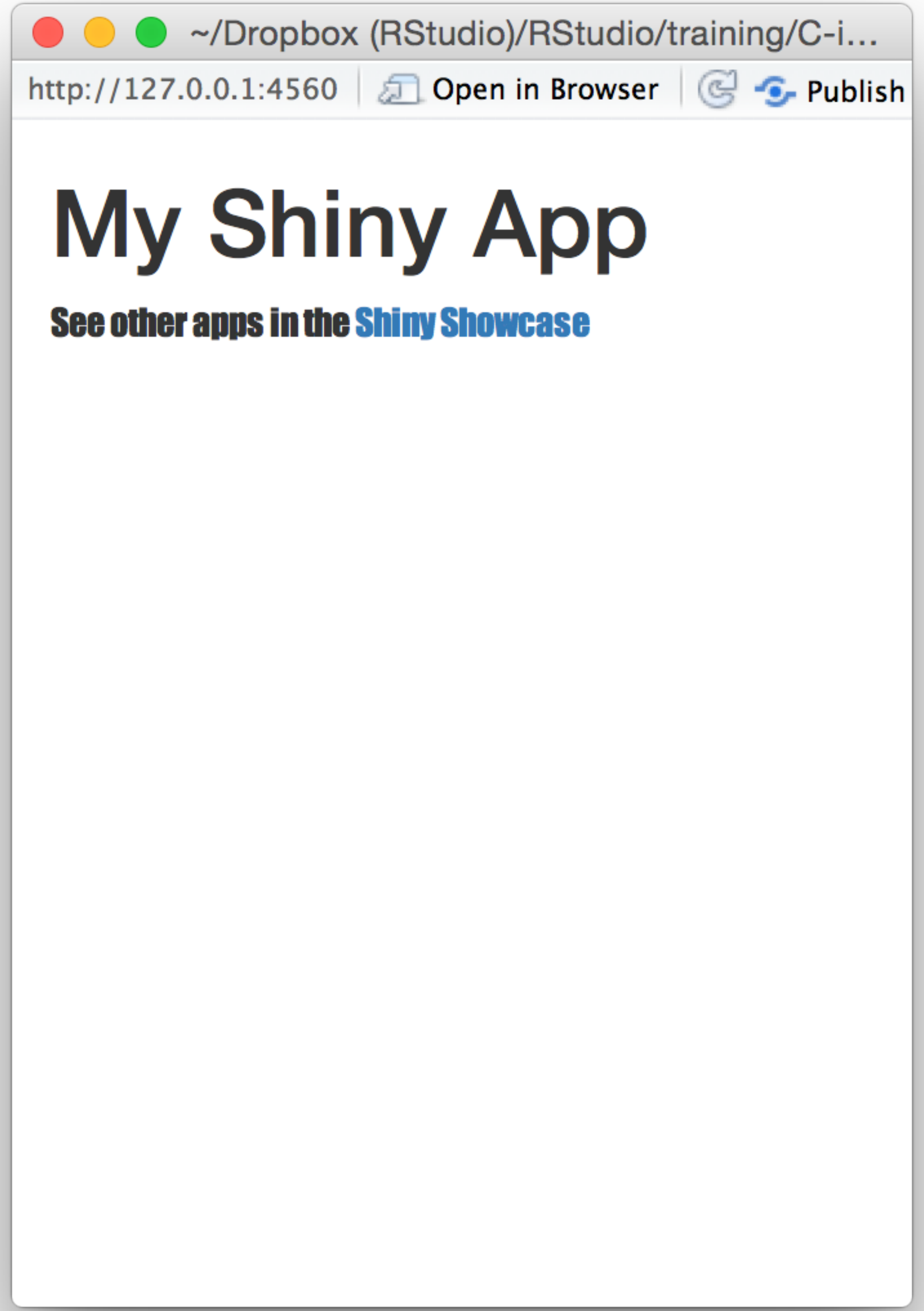
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  )
)
```



My Shiny App

See other apps in the Shiny Showcase

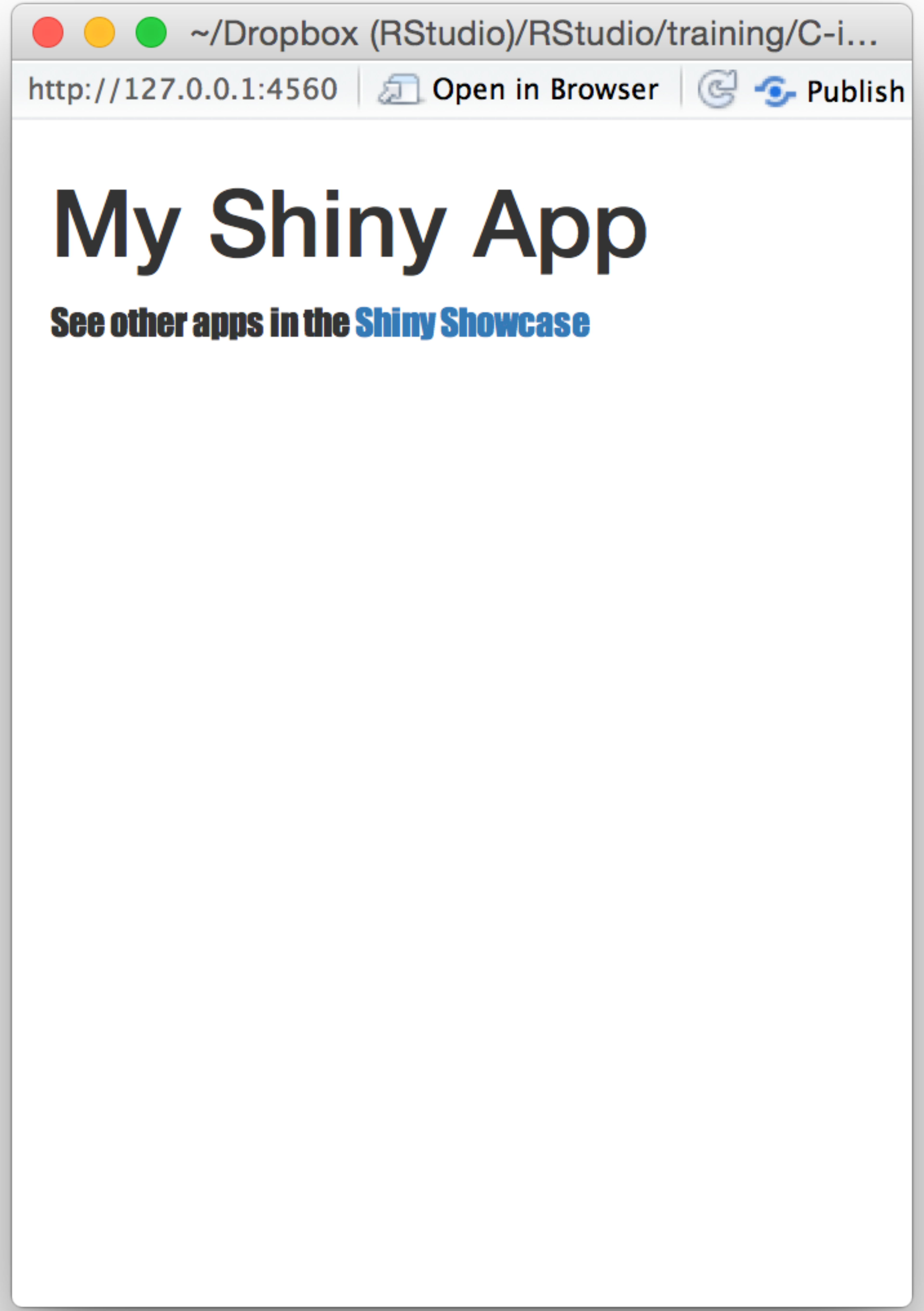
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



My Shiny App

See other apps in the Shiny Showcase

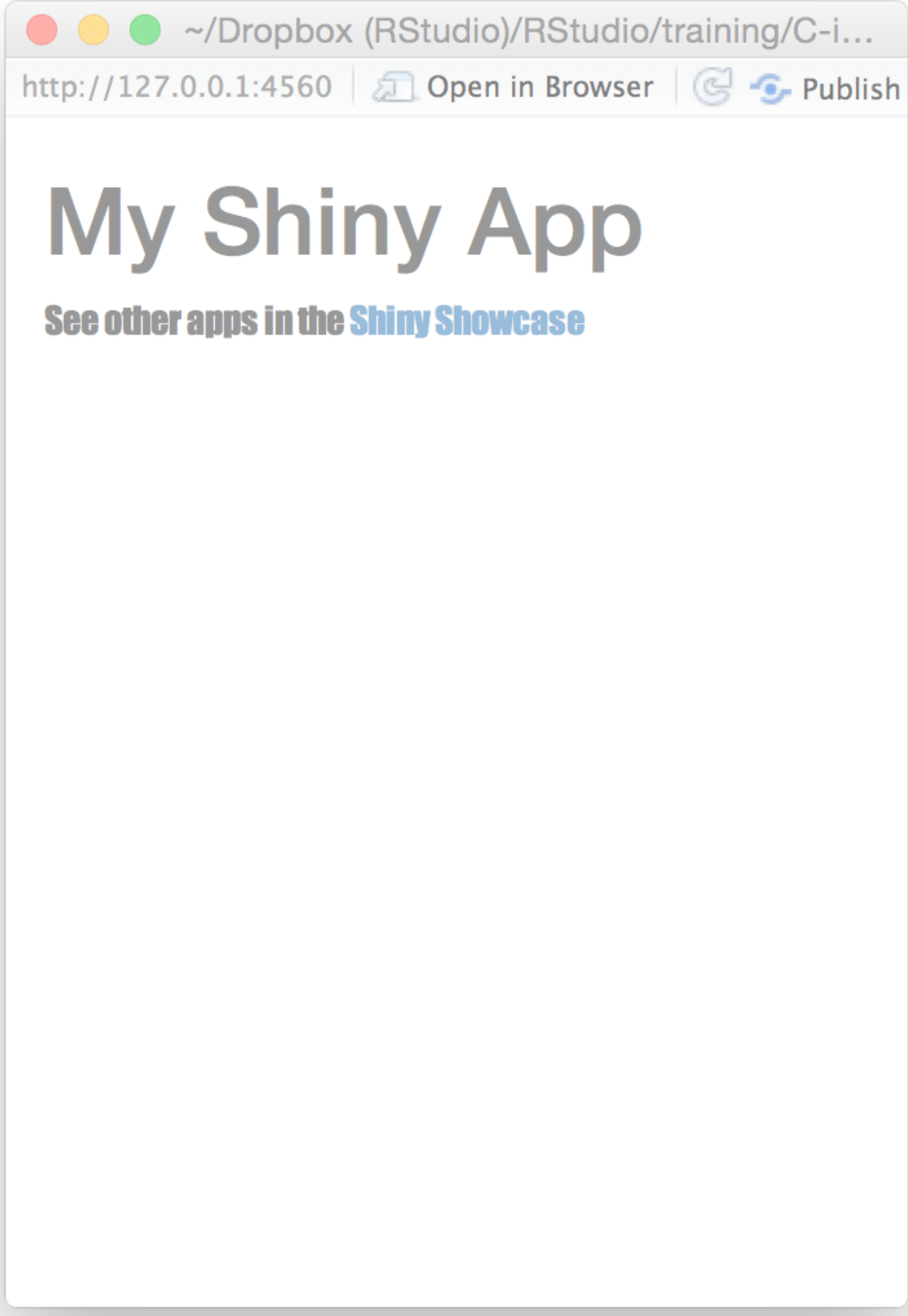
```
fluidPage(
  tags$h1("My Shiny App"),
  tags$p(style = "font-family:Impact",
    "See other apps in the",
    tags$a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



My Shiny App

See other apps in the Shiny Showcase

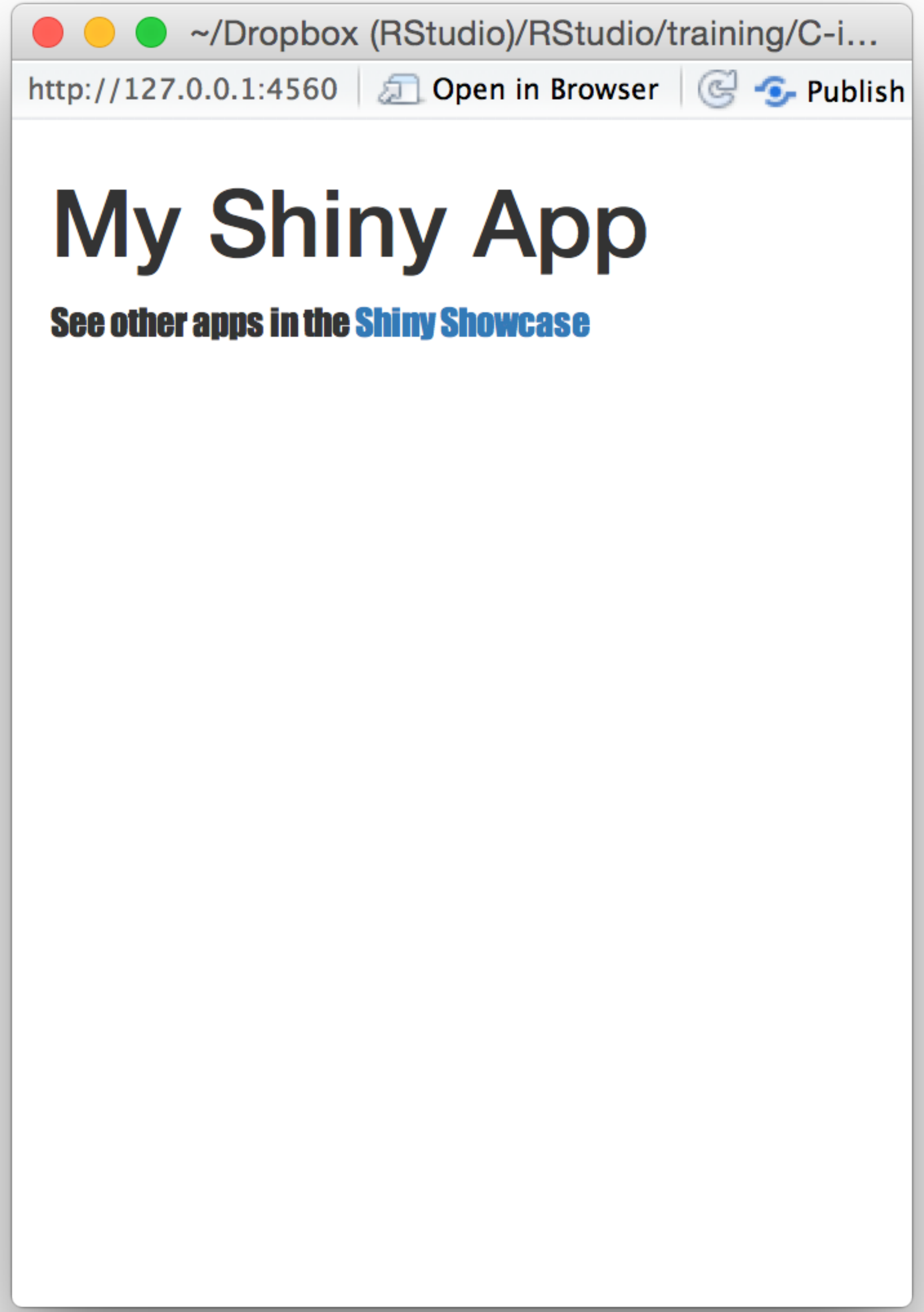
```
fluidPage(

    <div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>

)
```



My Shiny App

See other apps in the Shiny Showcase

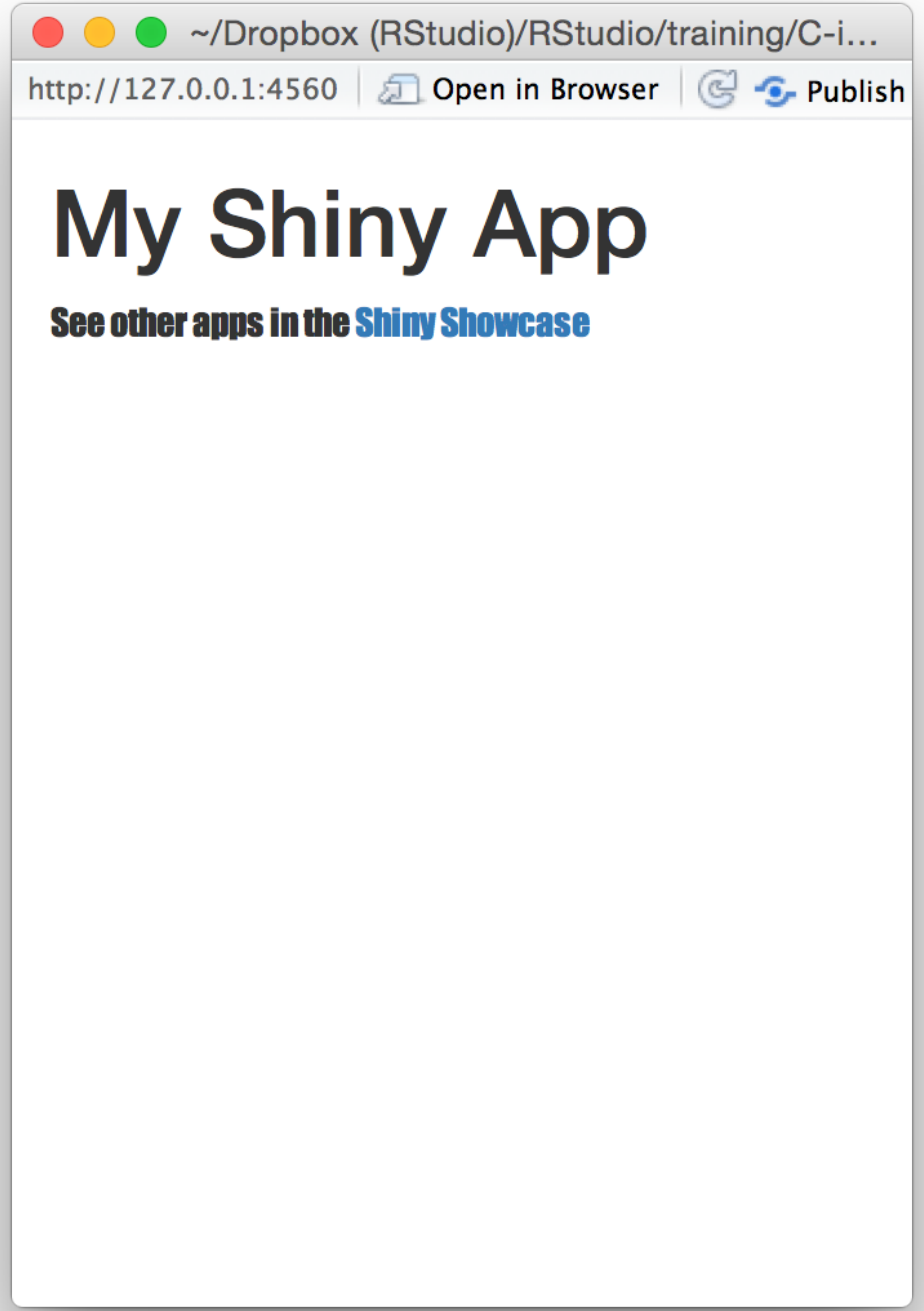
```
fluidPage(

    '<div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>'

)
```



My Shiny App

See other apps in the Shiny Showcase

```
fluidPage(
  HTML(
    '<div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>'
  )
)
```



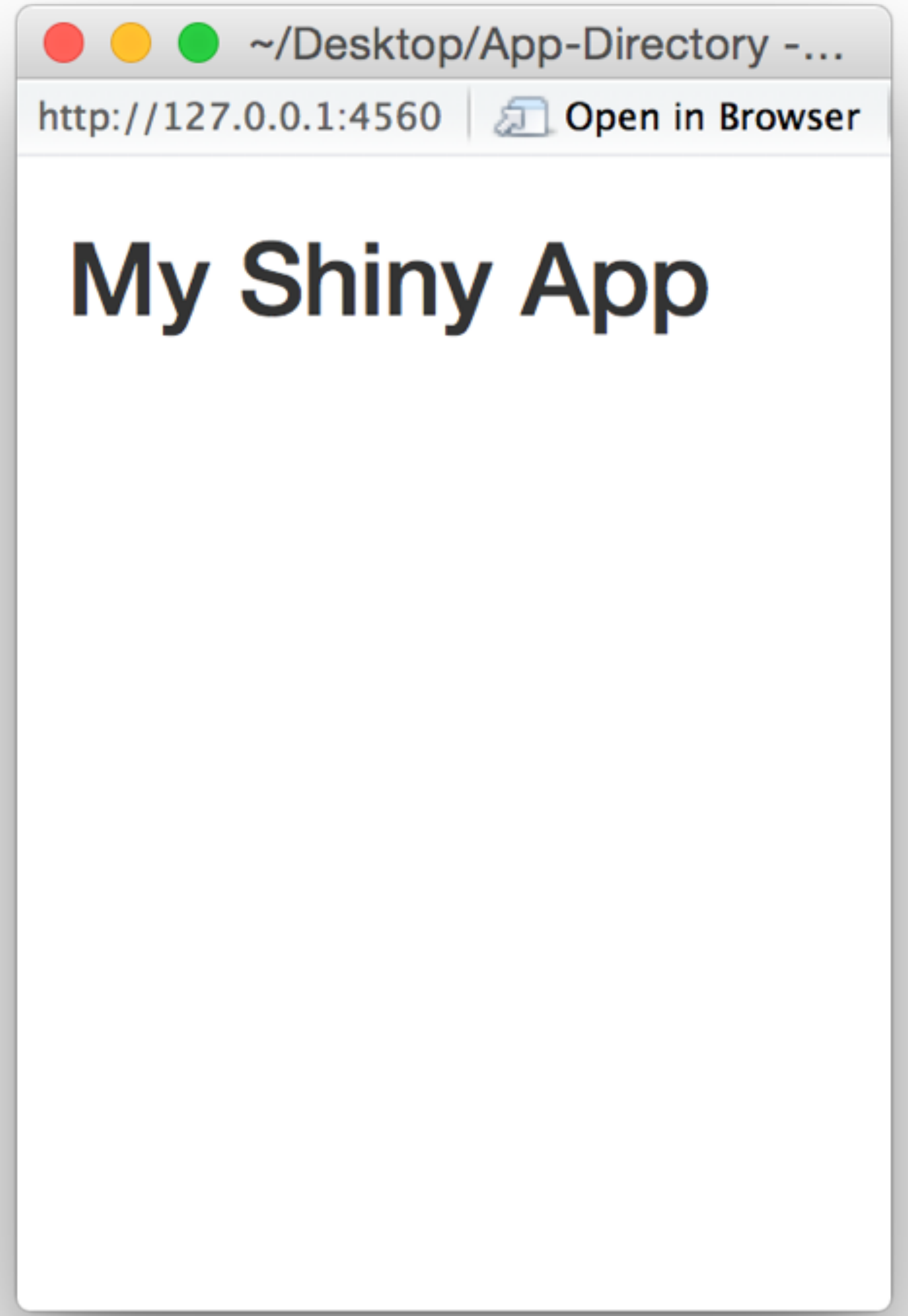
My Shiny App

See other apps in the Shiny Showcase

Raw HTML

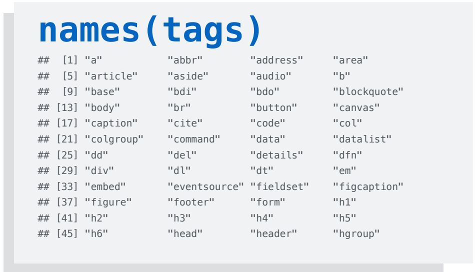
Use HTML() to pass a character string as raw HTML

```
fluidPage(
  HTML("<h1>My Shiny App</h1>")
)
```



My Shiny App

Recap: Reactive values



```
names(tags)
##  [1] "a"          "abbr"        "address"    "area"
##  [5] "article"    "aside"       "audio"      "b"
##  [9] "base"       "bdi"         "bdo"        "blockquote"
## [13] "body"       "br"          "button"     "canvas"
## [17] "caption"    "cite"        "code"       "col"
## [21] "colgroup"   "command"     "data"       "datalist"
## [25] "dd"         "del"         "details"    "dfn"
## [29] "div"        "dl"          "dt"         "em"
## [33] "embed"      "eventsource" "fieldset"   "figcaption"
## [37] "figure"     "footer"      "form"       "h1"
## [41] "h2"         "h3"          "h4"         "h5"
## [45] "h6"         "head"        "header"     "hgroup"
```

Add elements with the **tags$** functions

```
<p id="foo">
  foo
</p>
```

unnamed arguments are passed into HTML tags

```
<p id="foo">
  foo
</p>
```

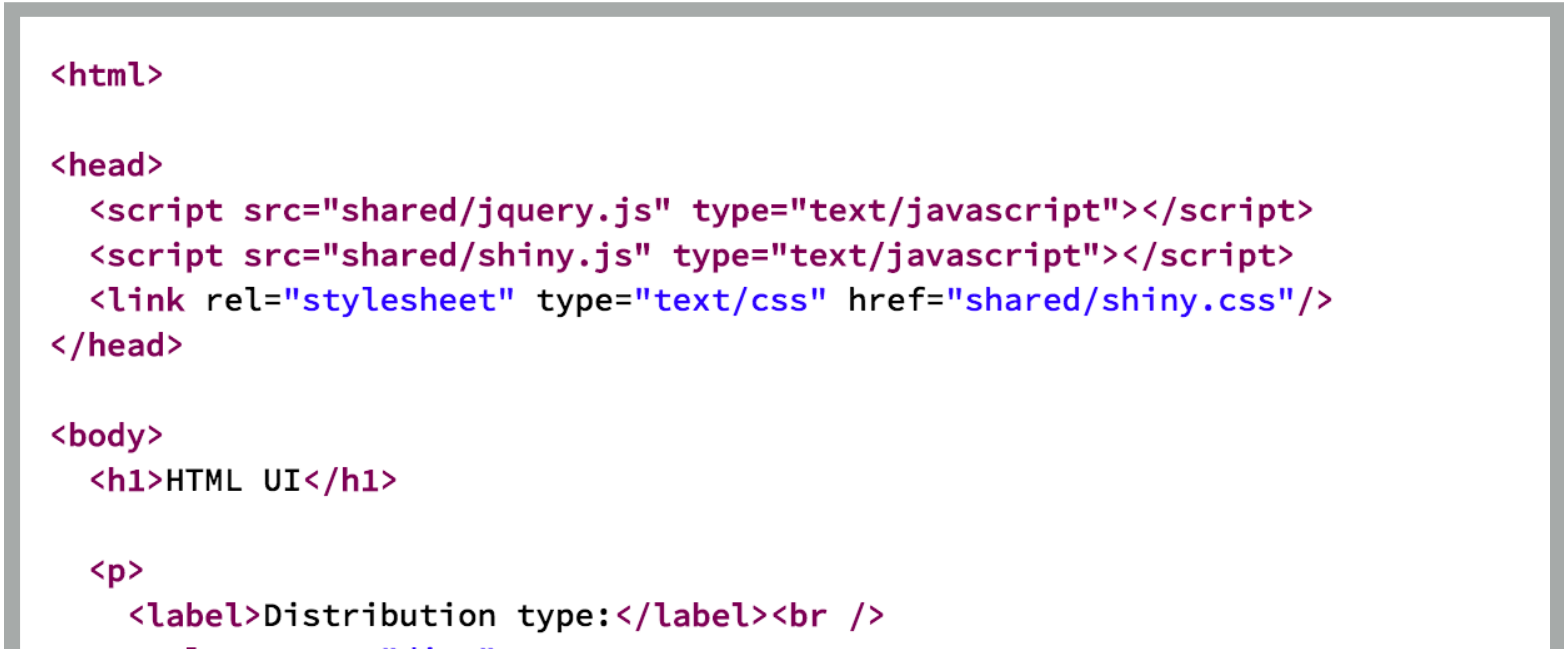
named arguments are passed as HTML tag attributes

Add raw html with **HTML()**

HTML UI

shiny.rstudio.com/articles/html-ui.html

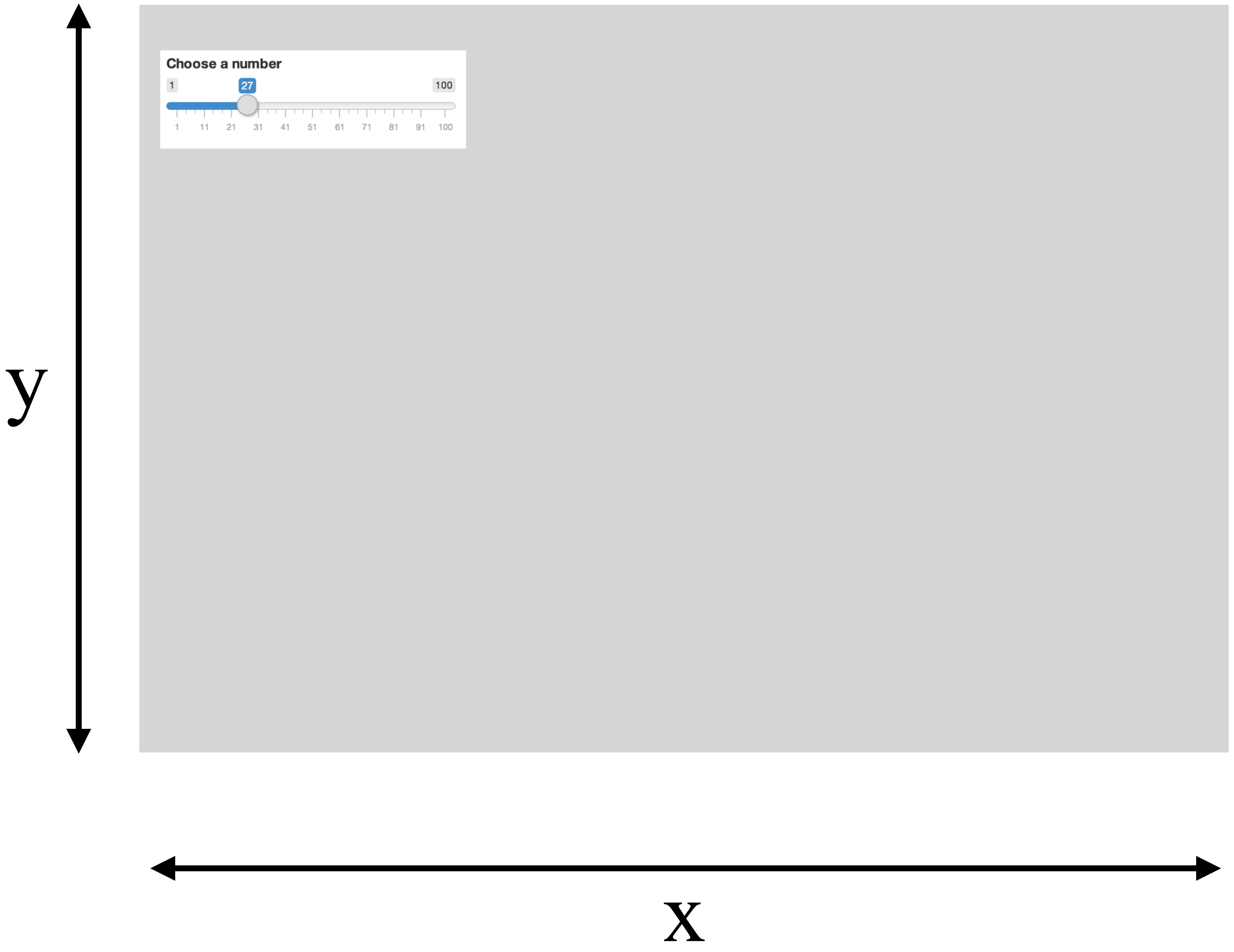
Build your entire UI from HTML



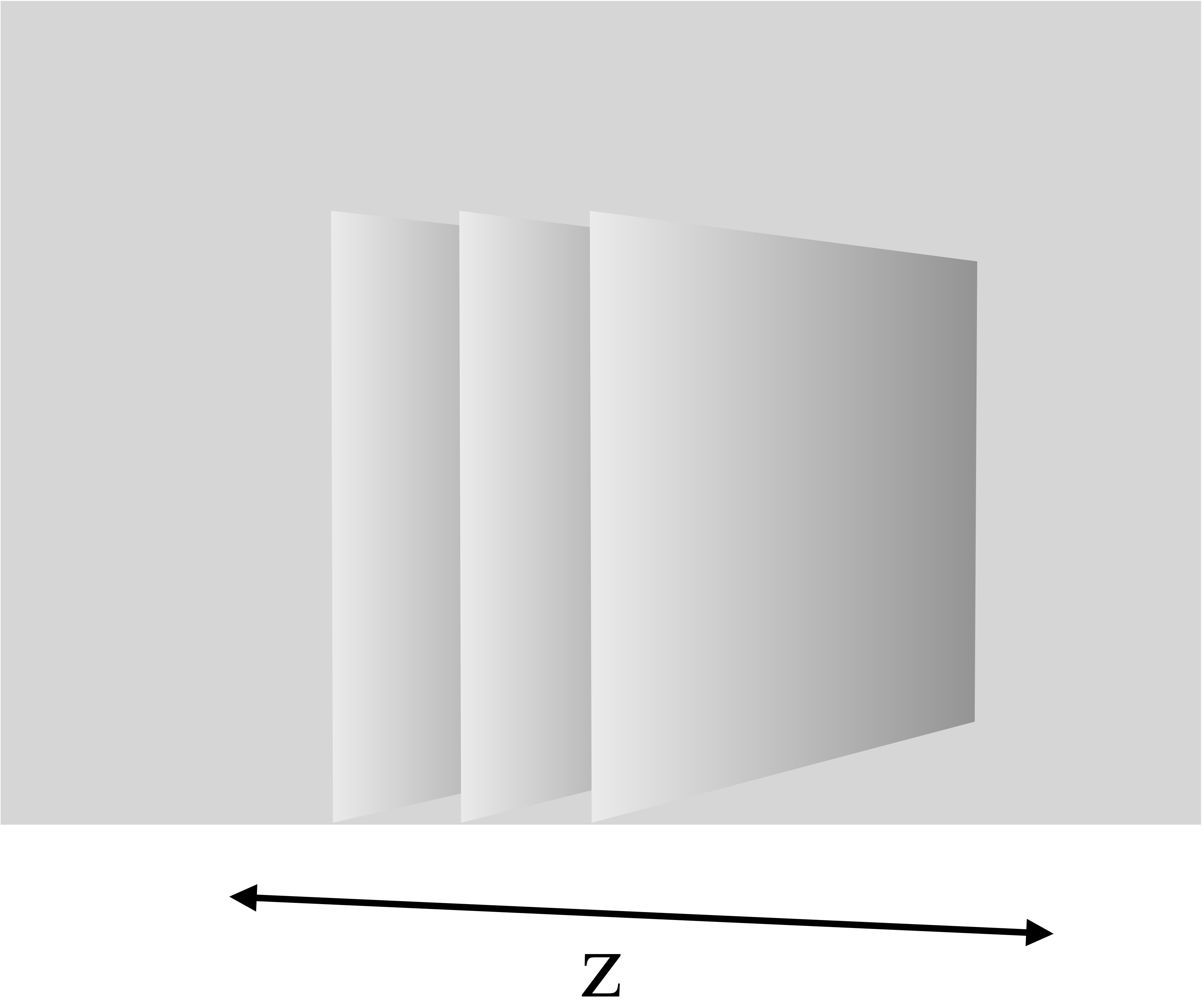
```
<html>

<head>
  <script src="shared/jquery.js" type="text/javascript"></script>
  <script src="shared/shiny.js" type="text/javascript"></script>
  <link rel="stylesheet" type="text/css" href="shared/shiny.css"/>
</head>

<body>
  <h1>HTML UI</h1>

  <p>
    <label>Distribution type:</label><br />
```

Create a layout

Use layout functions to position elements within your app



Use layout functions to position elements within your app



Layout functions

Add HTML that divides the UI into a grid

```
fluidRow()
```

```
<div class="row"></div>
```

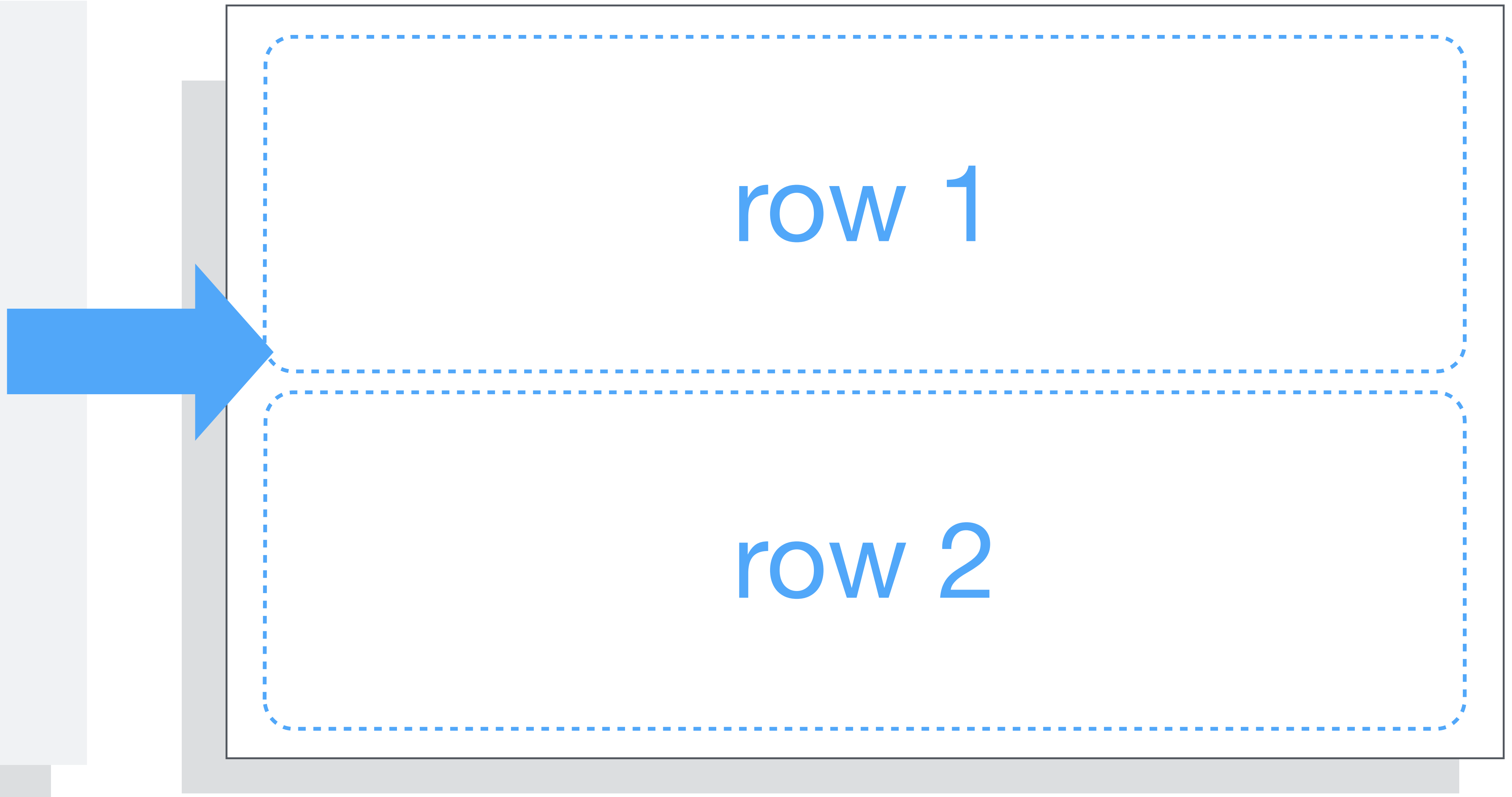
```
column(width = 2)
```

```
<div class="col-sm-2"></div>
```

fluidRow()

`fluidRow()` adds rows to the grid. Each new row goes below the previous rows.

```
ui <- fluidPage(
  fluidRow(),
  fluidRow()
)
```



row 1

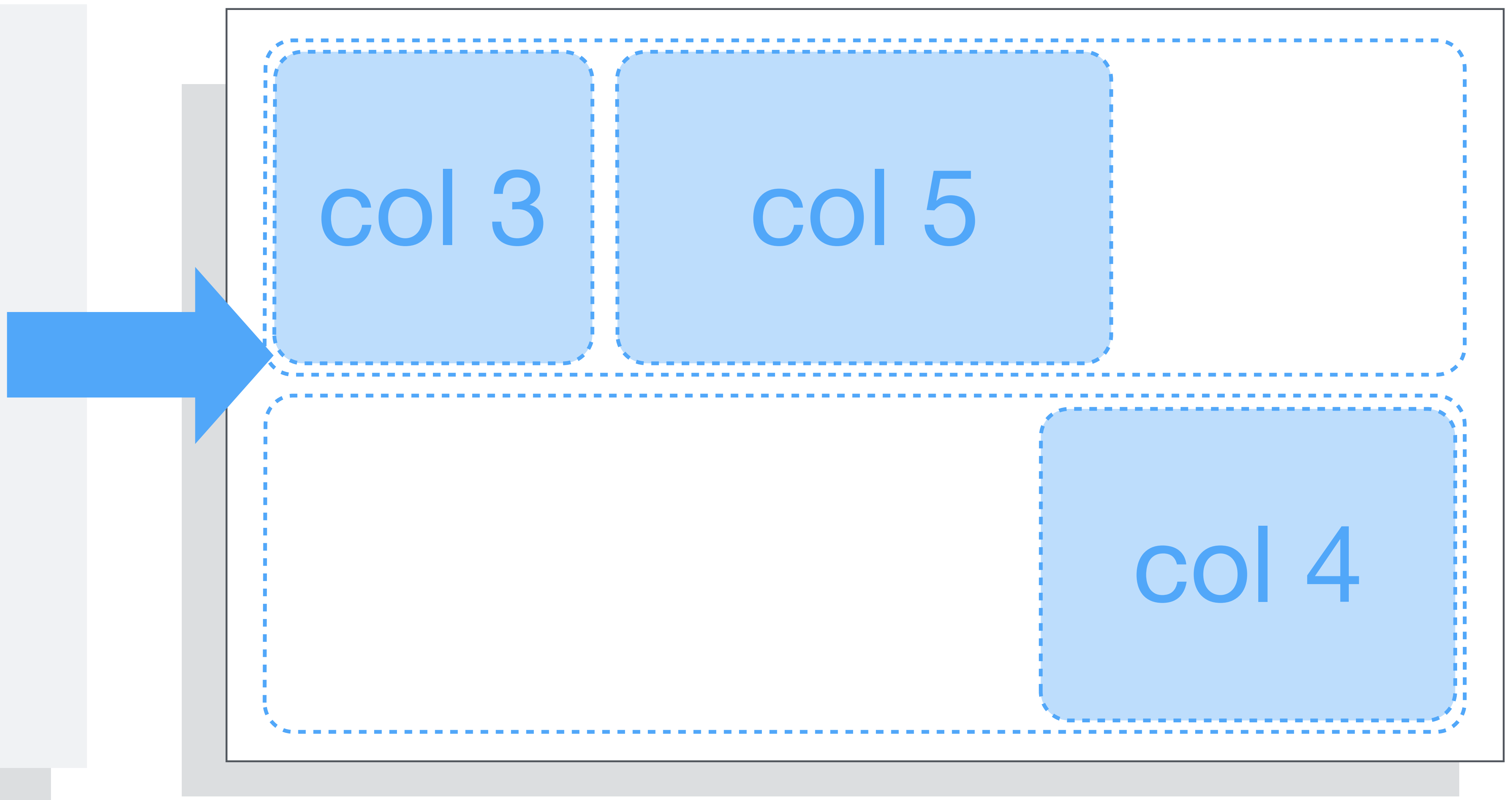
row 2

column()

column() adds columns within a row. Each new column goes to the left of the previous column.

Specify the **width** and **offset** of each column out of 12

```
ui <- fluidPage(
  fluidRow(
   column(3),
   column(5)),
  fluidRow(
   column(4, offset = 8)
)
```



To place an element in the grid, call it as an argument of a layout function

```
fluidRow()
```

```
<div class="row"></div>
```

```
column(2)
```

```
<div class="col-sm-2">
</div>
```

To place an element in the grid, call it as an argument of a layout function

```
fluidRow("In the row")
```

```
<div class="row"></div>
```

```
column(2)
```

```
<div class="col-sm-2">
</div>
```

To place an element in the grid, call it as an argument of a layout function

```
fluidRow("In the row")
```

```
<div class="row">In the row</div>
```

```
column(2)
```

```
<div class="col-sm-2">
</div>
```

To place an element in the grid, call it as an argument of a layout function

```
fluidRow("In the row")
```

```
<div class="row">In the row</div>
```

```
column(2, plotOutput("hist"))
```

```
<div class="col-sm-2">
</div>
```

To place an element in the grid, call it as an argument of a layout function

```
fluidRow("In the row")
```

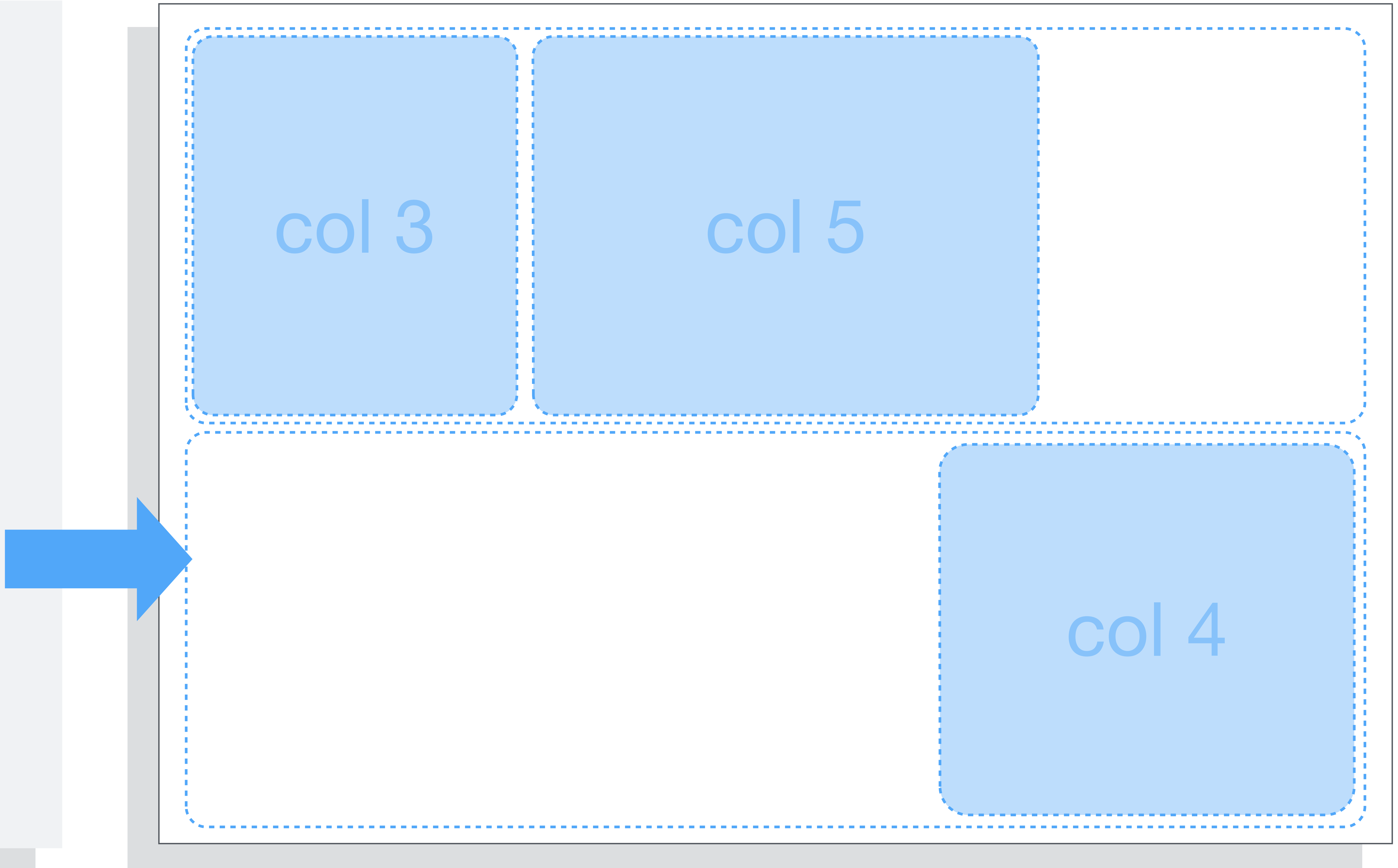
```
<div class="row">In the row</div>
```

```
column(2, plotOutput("hist"))
```

```
<div class="col-sm-2">
  <div id="hist" class="shiny-plot-output"
  style="width: 100% ; height: 400px"></div>
</div>
```

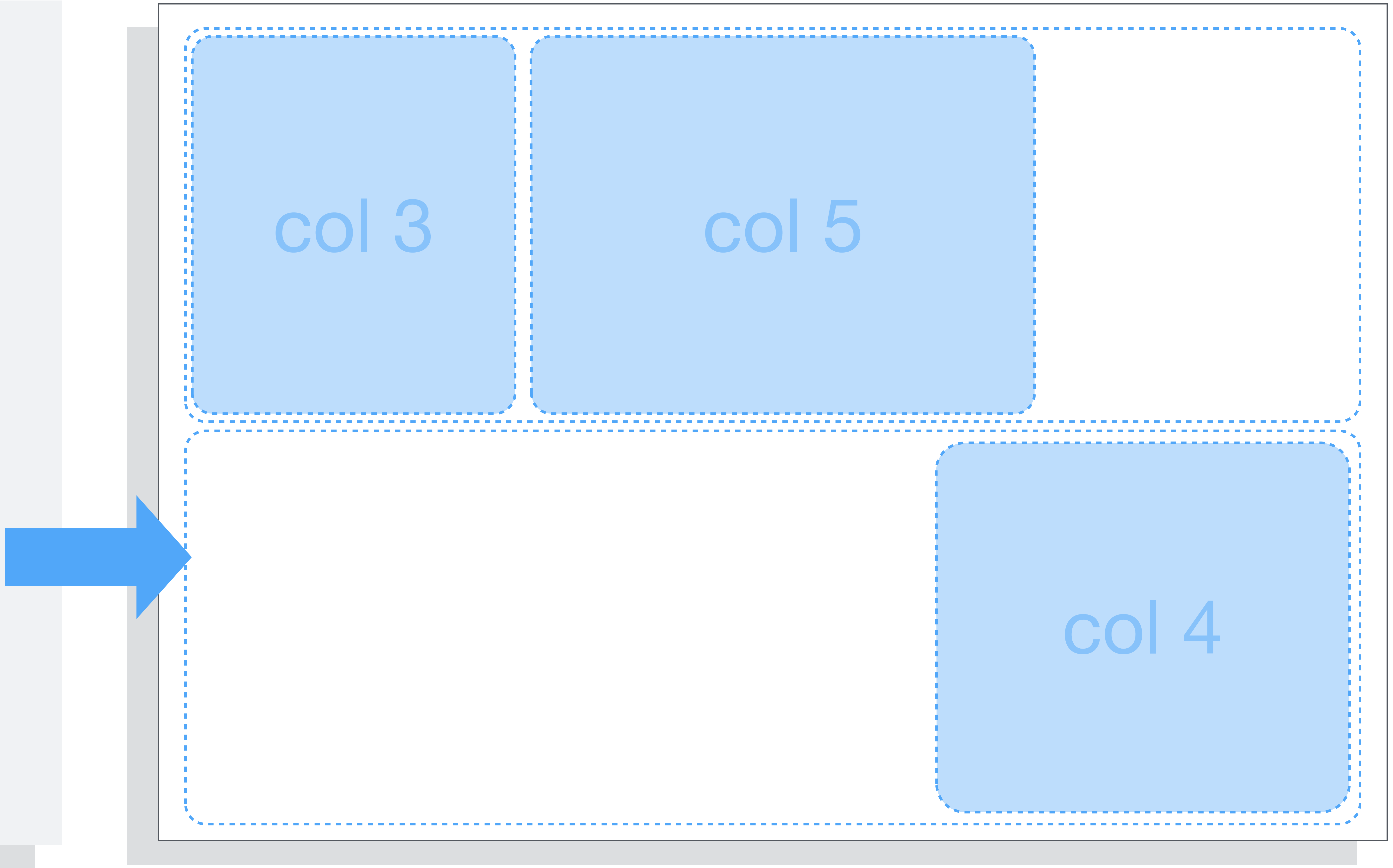
To place an element in the grid, call it as an argument of a layout function

```
fluidPage(
  fluidRow(
   column(3),
   column(5)
  ),
  fluidRow(
    column(4, offset = 8,)
  )
)
```



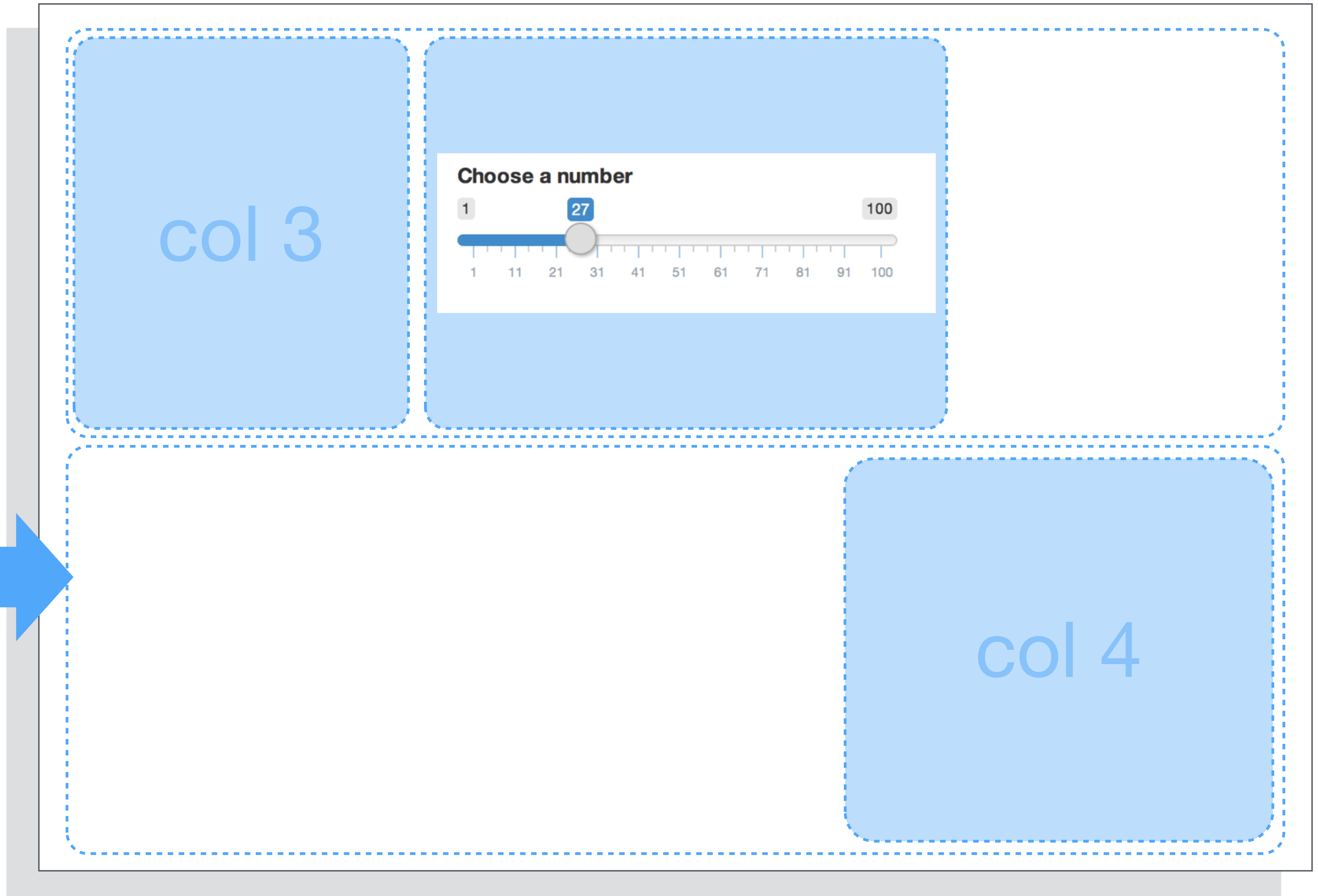
To place an element in the grid, call it as an argument of a layout function

```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8)
  )
)
```



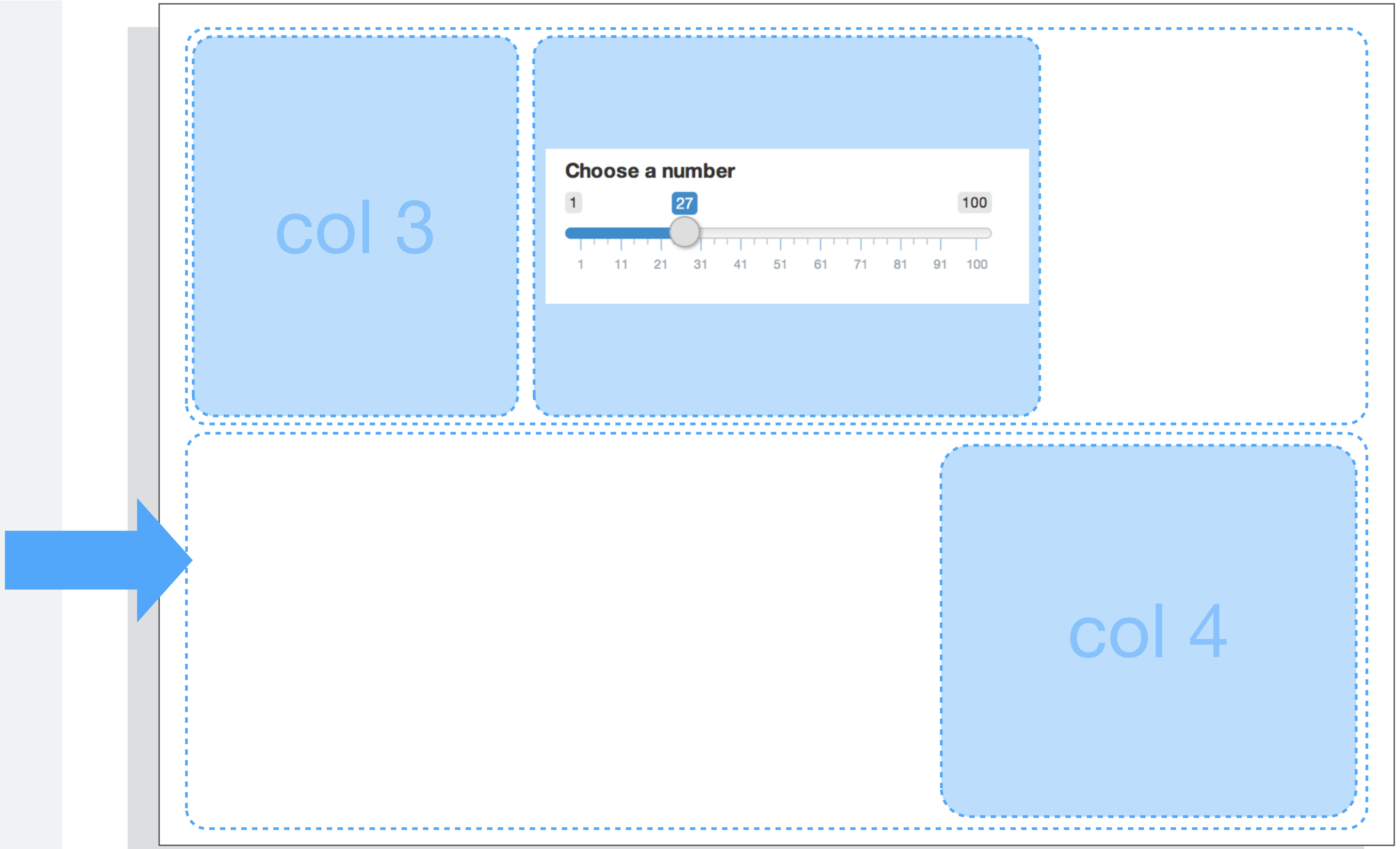
To place an element in the grid, call it as an argument of a layout function

```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8)
  )
)
```



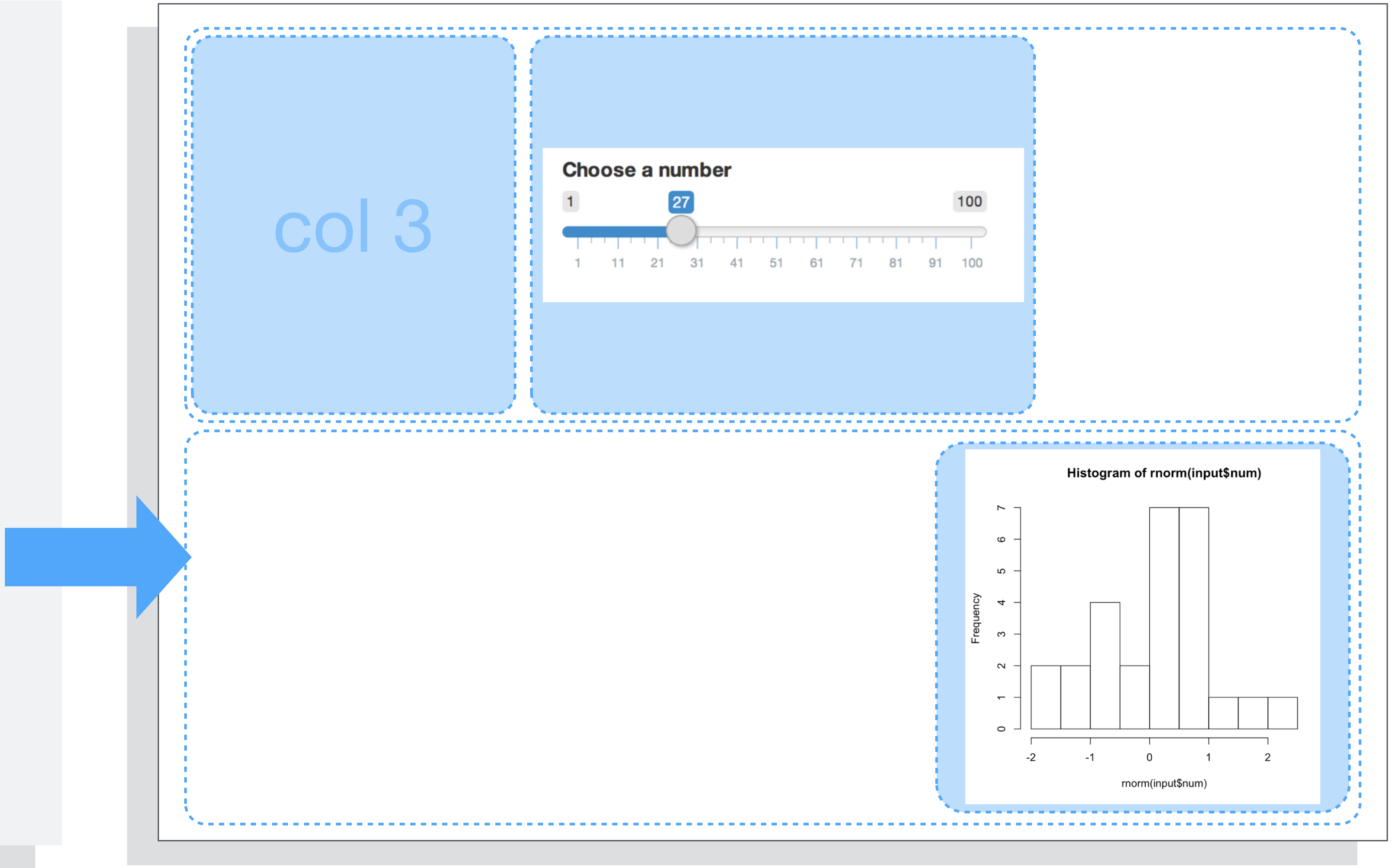
To place an element in the grid, call it as an argument of a layout function

```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
```



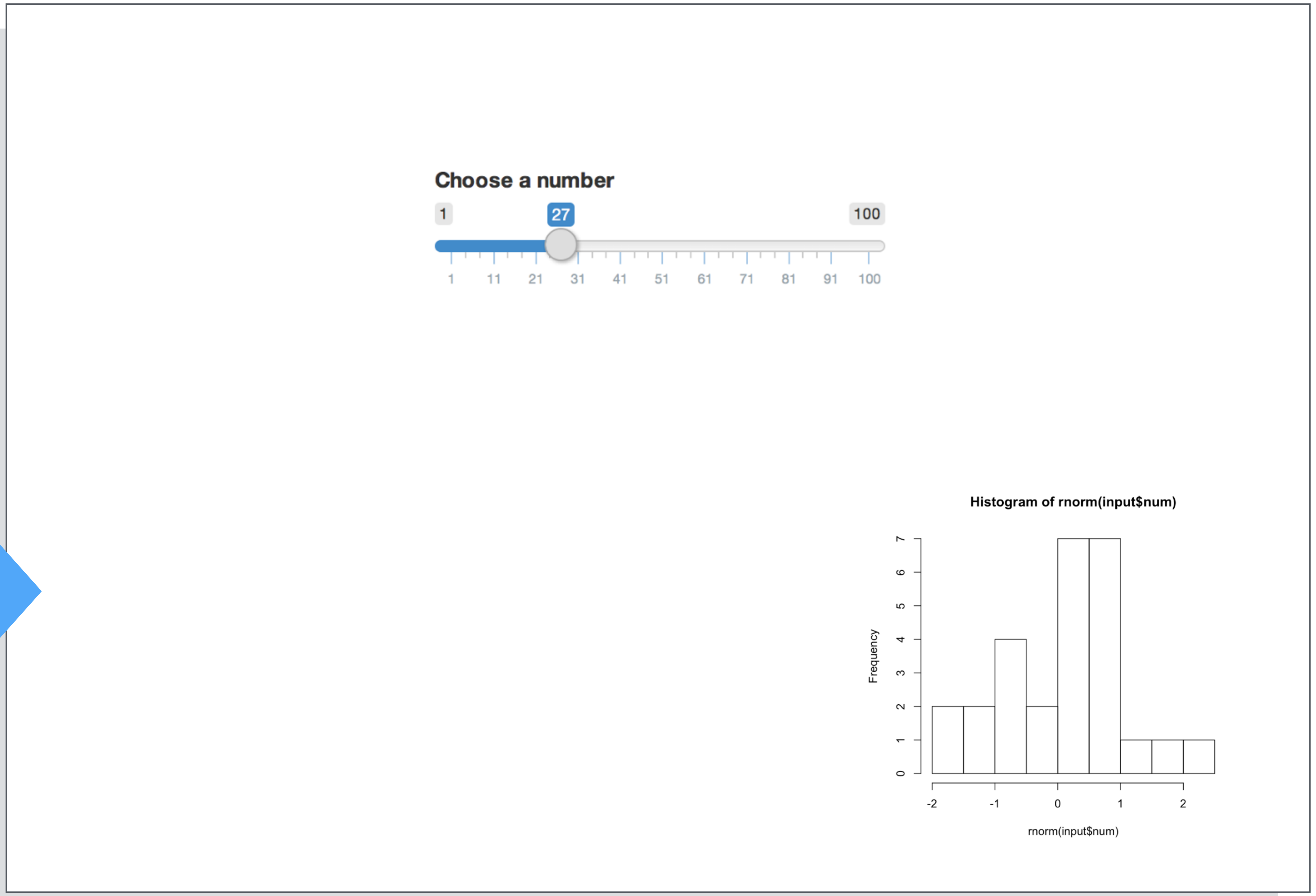
To place an element in the grid, call it as an argument of a layout function

```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
```

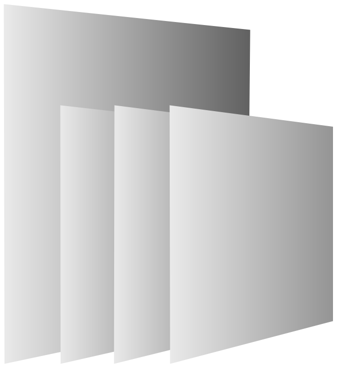


To place an element in the grid, call it as an argument of a layout function

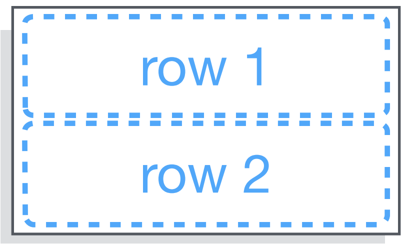
```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
```



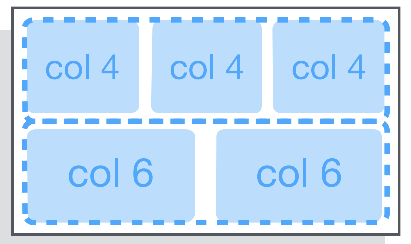
Recap: Layout functions



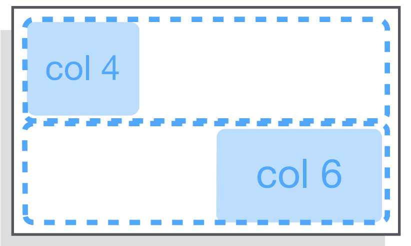
Position elements in a grid, or stack them in layers.



Use **fluidRow()** to arrange elements in rows



Use **column()** to arrange elements in columns

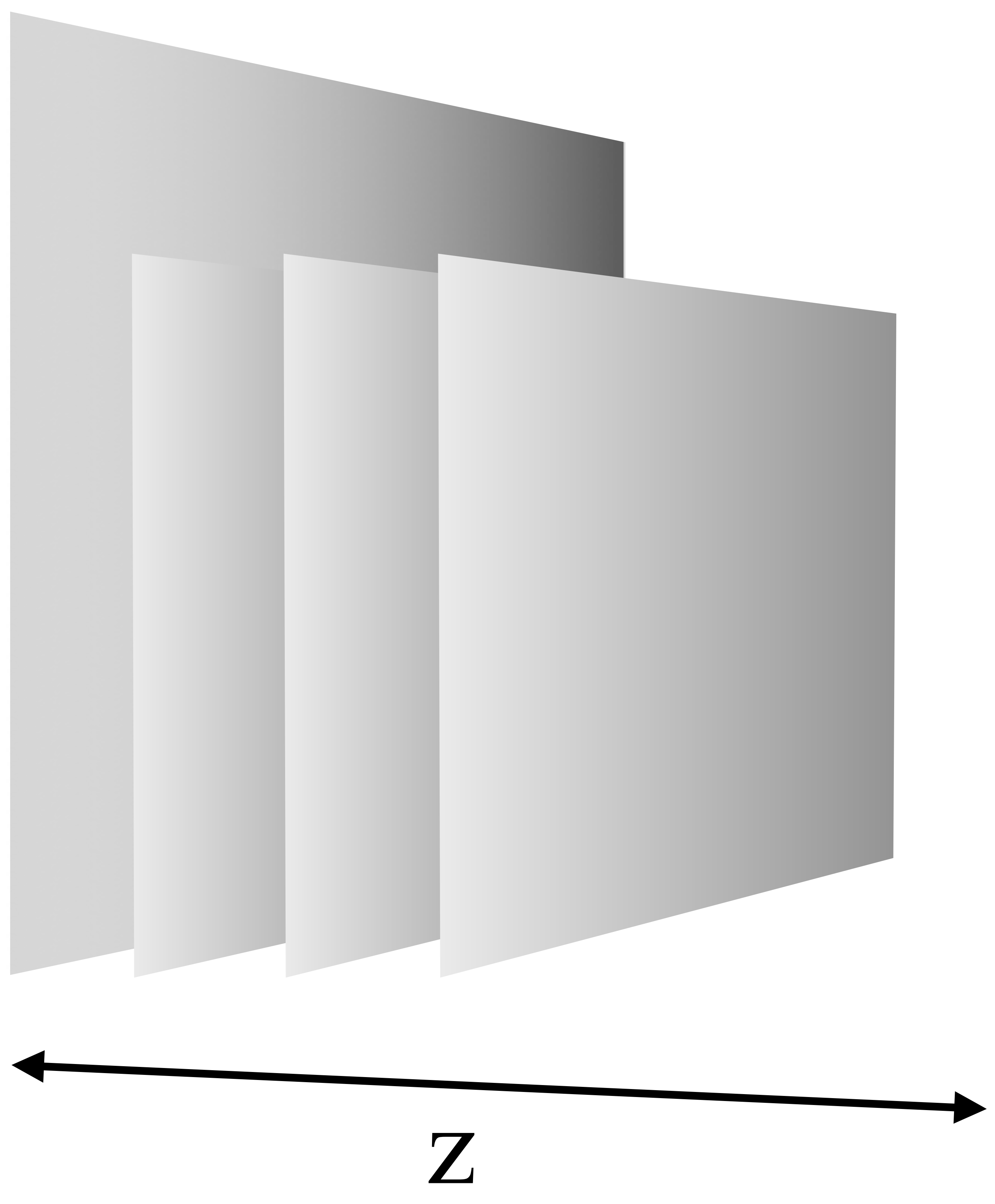


Column takes **width** and **offset** arguments

Assemble

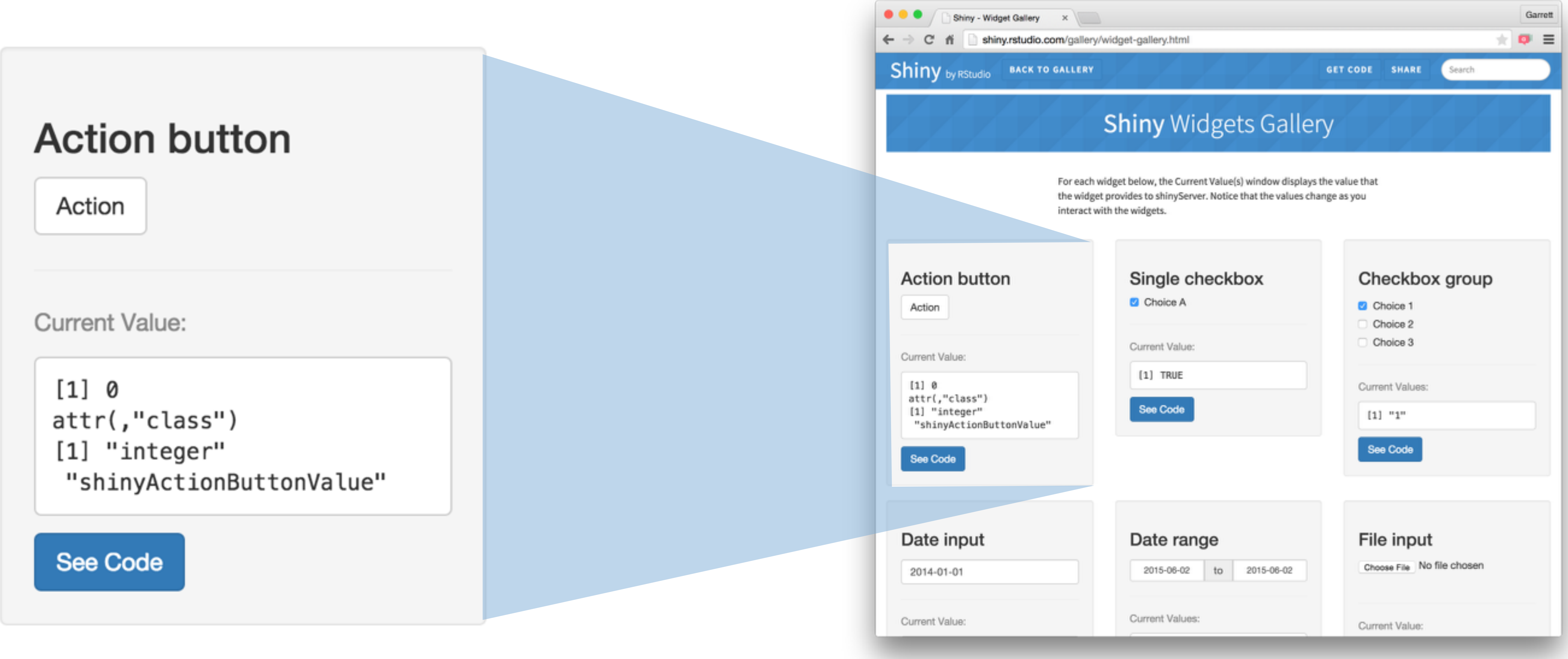
layers of panels

Use layout functions to position elements within your app



Panels

Panels to group multiple elements into a single unit with its own properties.



```
[1] 0
attr(,"class")
[1] "integer"
 "shinyActionButtonValue"
```

For each widget below, the Current Value(s) window displays the value that the widget provides to shinyServer. Notice that the values change as you interact with the widgets.

```
[1] 0
attr(,"class")
[1] "integer"
 "shinyActionButtonValue"
```

```
[1] TRUE
```

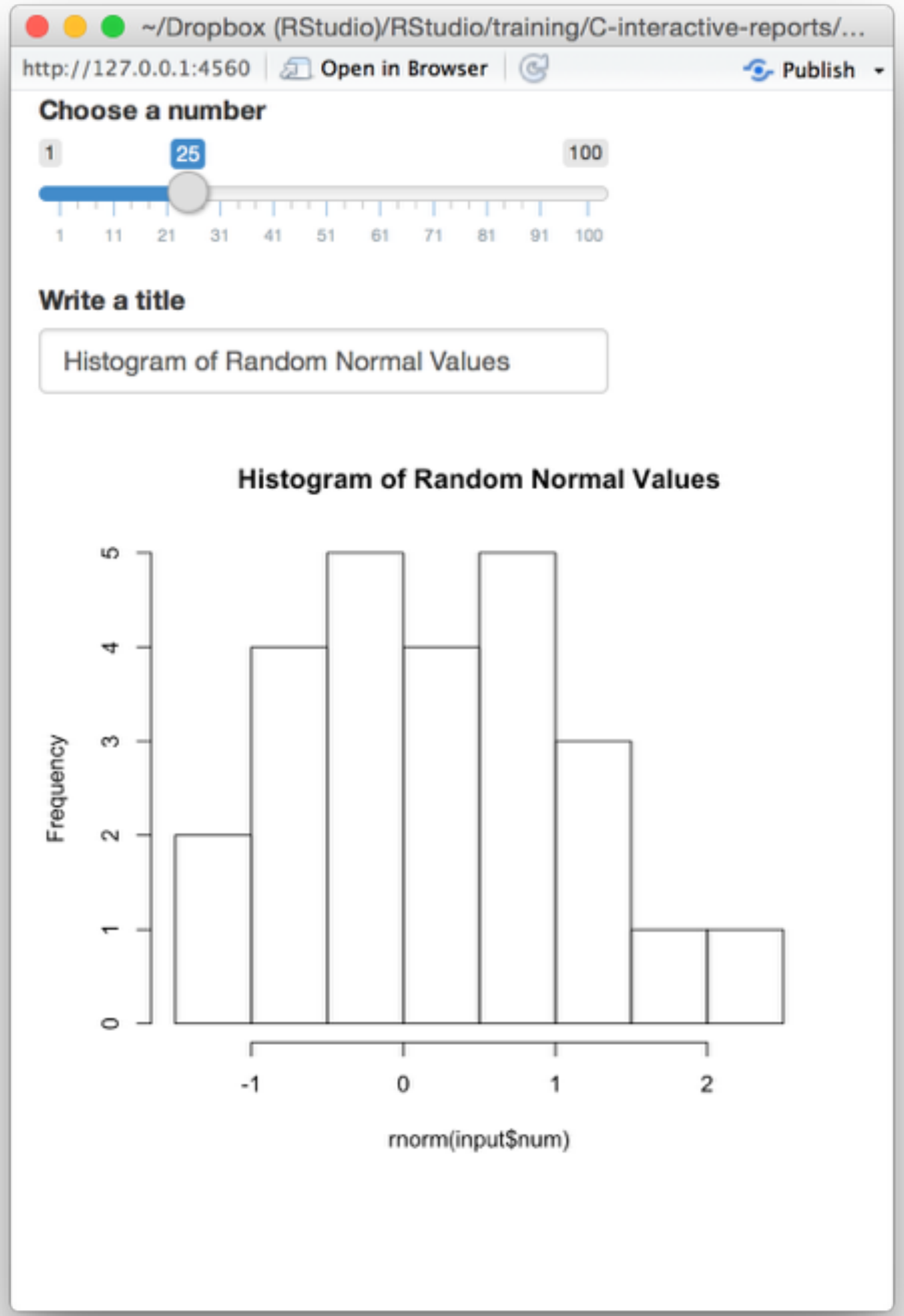
```
[1] "1"
```

http://shiny.rstudio.com/gallery/widget-gallery.html

wellPanel()

Groups elements into a grey "well"

```
# 04-well.R

ui <- fluidPage(

  sliderInput("num", "Choose a number",
    value = 25, min = 1, max = 100),
  textInput("title", value = "Histogram",
    label = "Write a title"),

 plotOutput("hist")
)
```



~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/...

http://127.0.0.1:4560 Open in Browser Publish

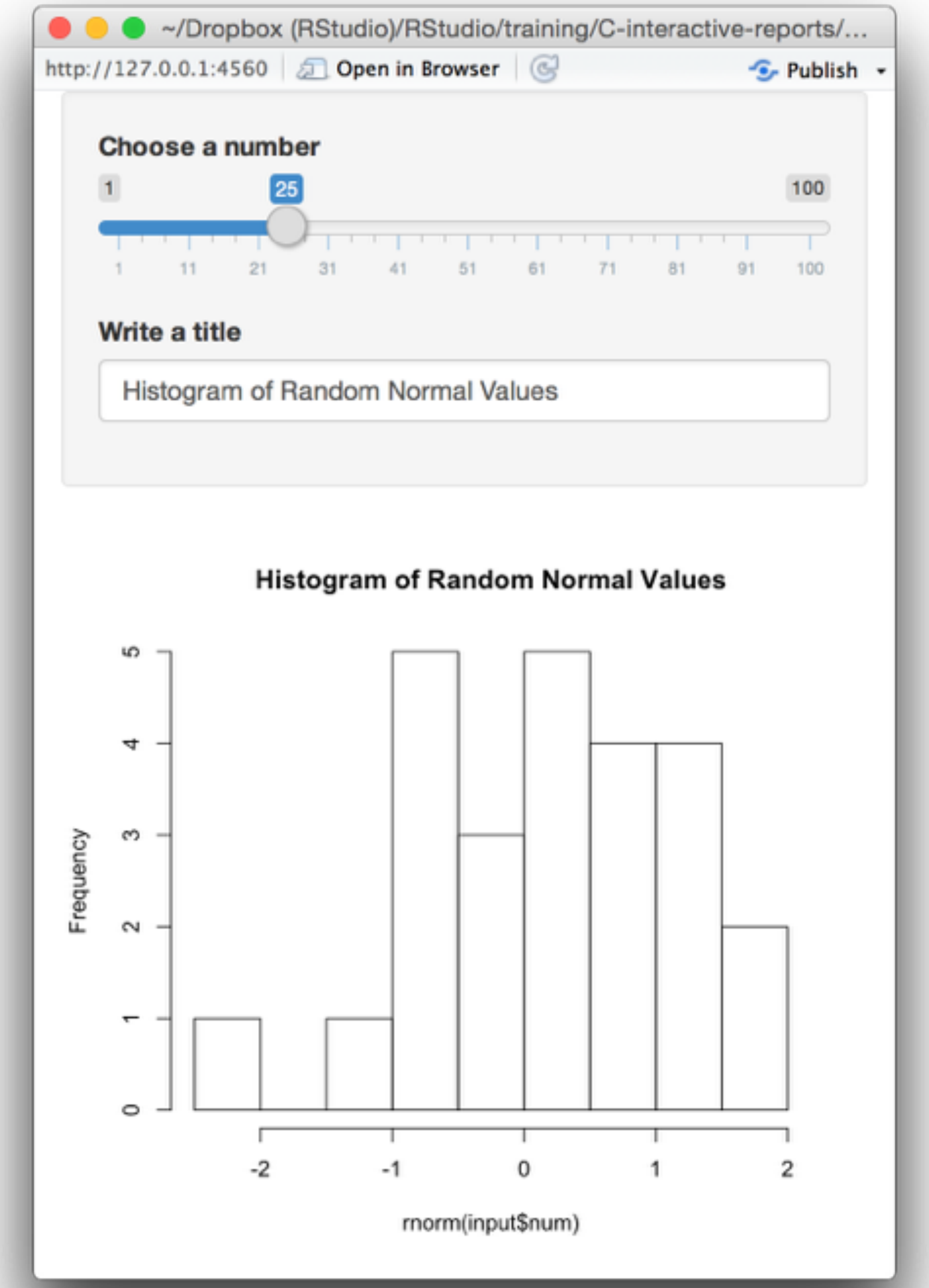
Choose a number: 25 (1 to 100)

Write a title: Histogram of Random Normal Values

wellPanel()

Groups elements into a grey "well"

```
# 04-well.R

ui <- fluidPage(
 wellPanel(
  sliderInput("num", "Choose a number",
    value = 25, min = 1, max = 100),
  textInput("title", value = "Histogram",
    label = "Write a title"),
 ),
 plotOutput("hist")
)
```



Choose a number

25

Write a title

Histogram of Random Normal Values

Histogram of Random Normal Values

`absolutePanel()`

Panel position set rigidly (absolutely), not fluidly

`conditionalPanel()`

A JavaScript expression determines whether panel is visible or not.

`fixedPanel()`

Panel is fixed to browser window and does not scroll with the page

`headerPanel()`

Panel for the app's title, used with pageWithSidebar()

`inputPanel()`

Panel with grey background, suitable for grouping inputs

`mainPanel()`

Panel for displaying output, used with pageWithSidebar()

`navlistPanel()`

Panel for displaying multiple stacked tabPanels(). Uses sidebar navigation

`sidebarPanel()`

Panel for displaying a sidebar of inputs, used with pageWithSidebar()

`tabPanel()`

Stackable panel. Used with navlistPanel() and tabsetPanel()

`tabsetPanel()`

Panel for displaying multiple stacked tabPanels(). Uses tab navigation

`titlePanel()`

Panel for the app's title, used with pageWithSidebar()

`wellPanel()`

Panel with grey background.

tabPanel()

`tabPanel()` creates a stackable layer of elements. Each tab is like a small UI of its own.

```
tabPanel("Tab 1", …)
```

A title
(for navigation)

elements to
appear in the tab

Combine tabPanel()'s with one of:

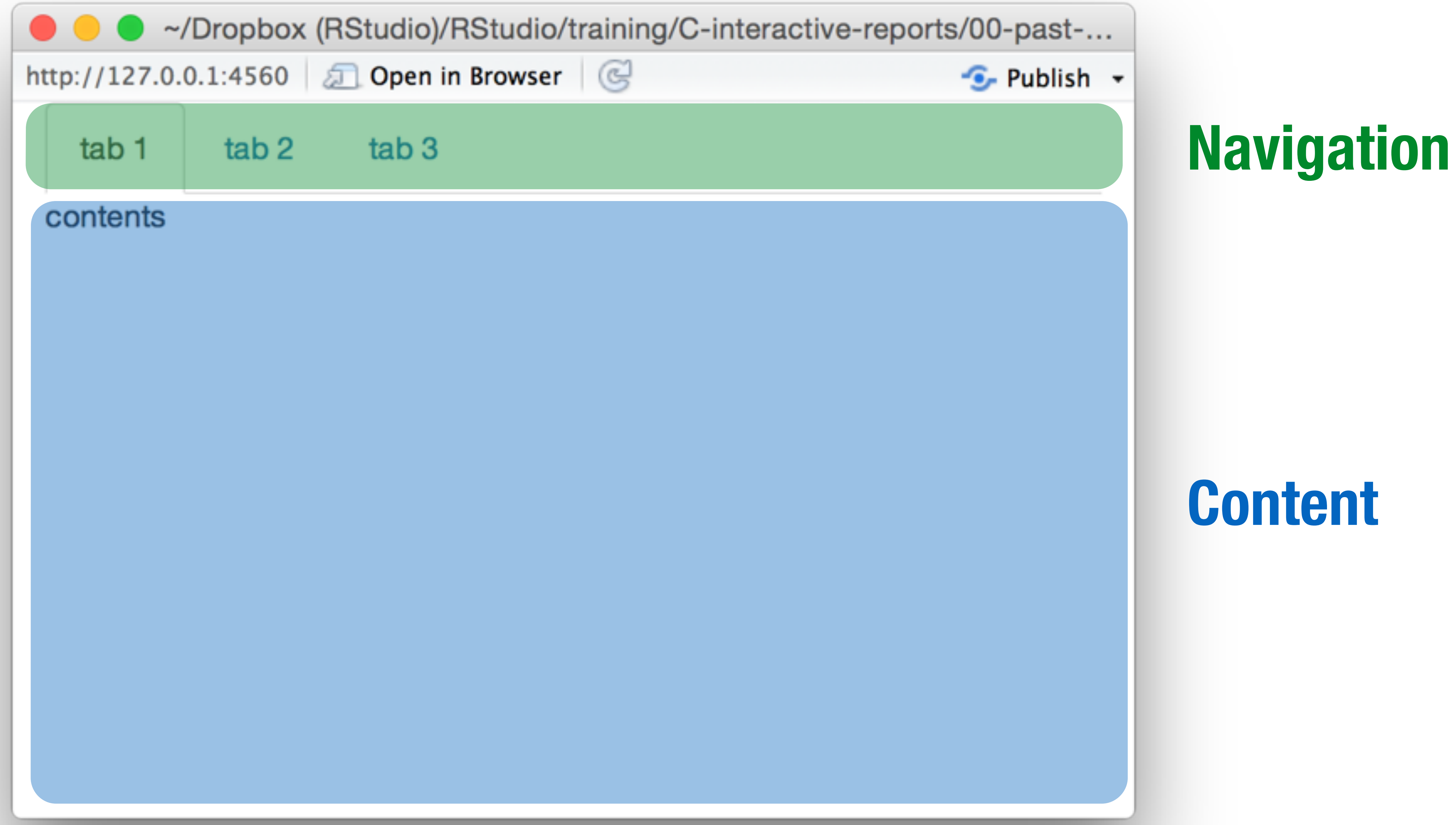
- tabsetPanel()
- navlistPanel()
- navbarPage()

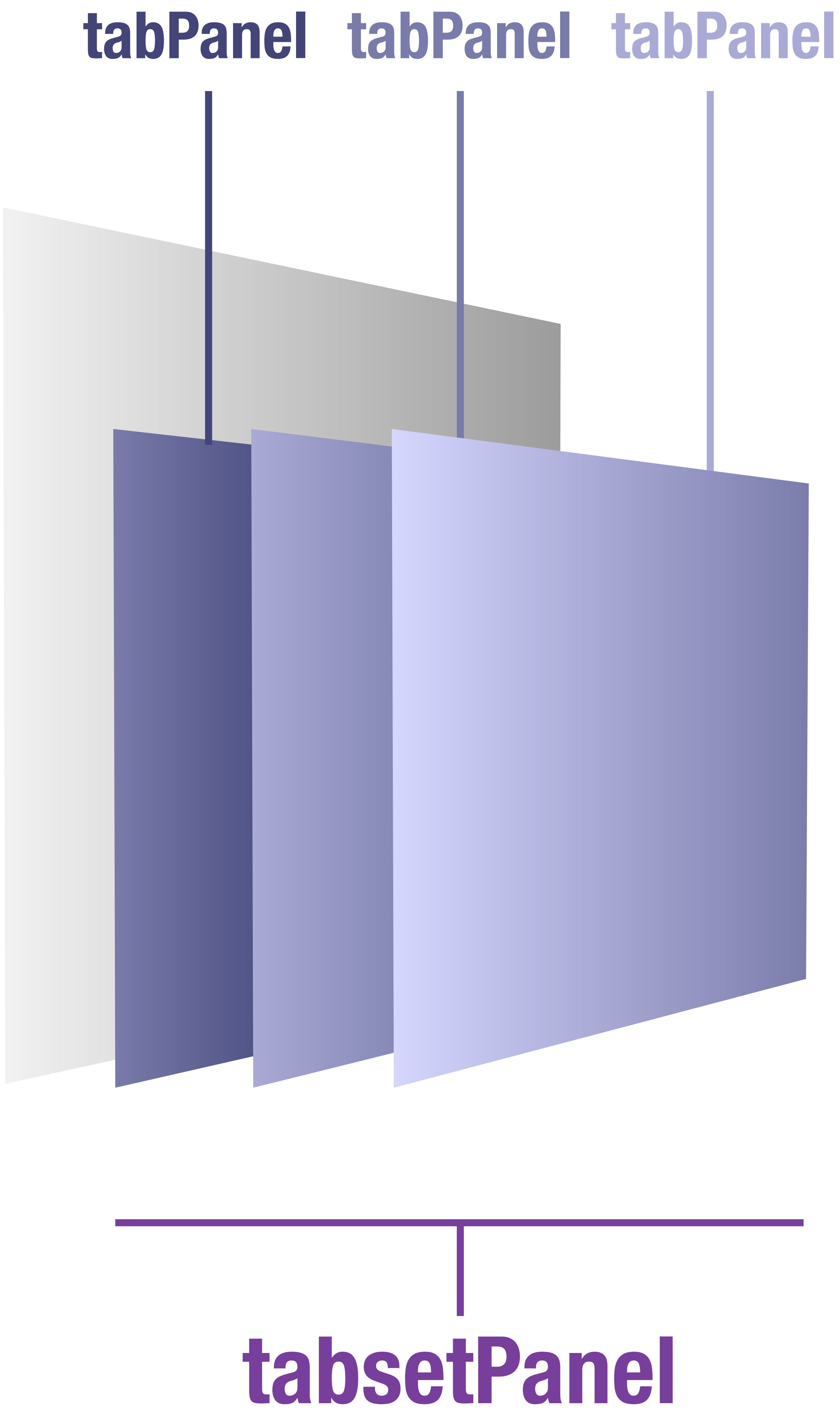
tabsetPanel()

`tabsetPanel()` combines tabs into a single *panel*.

Use *tabs* to navigate between tabs.

```
fluidPage(
  tabsetPanel(
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   tabPanel("tab 3", "contents")
  )
)
```



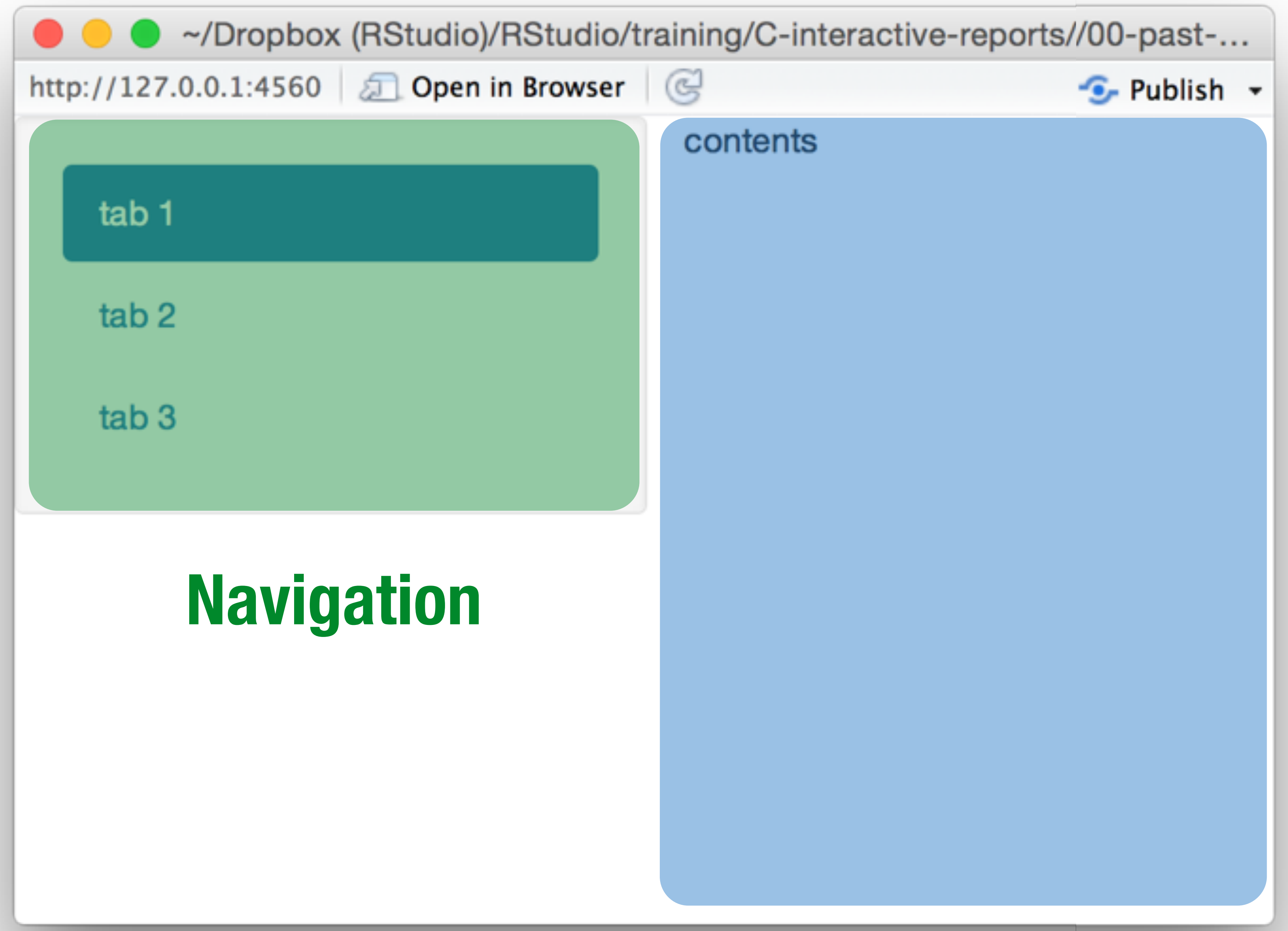

tabPanel
tabPanel
tabPanel
tabsetPanel

navlistPanel()

navlistPanel() combines tabs into a single *panel*.

Use *links* to navigate between tabs.

```
fluidPage(
  navlistPanel(
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   tabPanel("tab 3", "contents")
  )
)
```


~/Dropbox (RStudio)/RStudio/training/C-interactive-reports//00-past-...
http://127.0.0.1:4560
Open in Browser
Publish
tab 1
tab 2
tab 3
contents

Navigation

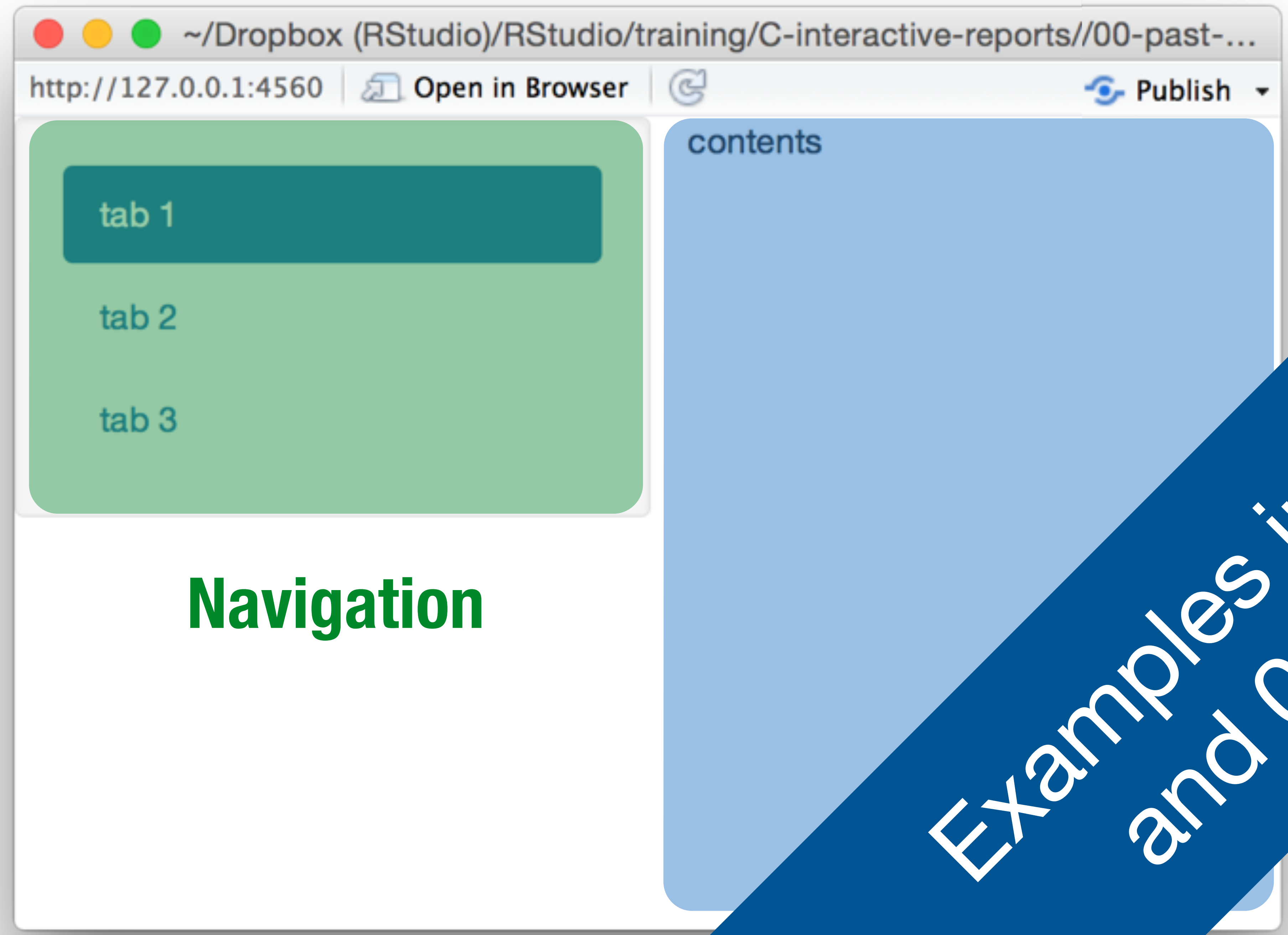
Content

navlistPanel()

`navlistPanel()` combines tabs into a single *panel*.

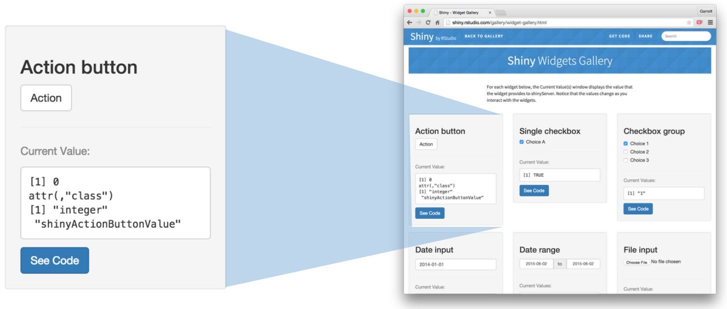
Use *links* to navigate between tabs.

```
fluidPage(
  navlistPanel(
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   tabPanel("tab 3", "contents")
  )
)
```



Examples in 05-tabs.R and 06-navlist.R

Recap: Panels

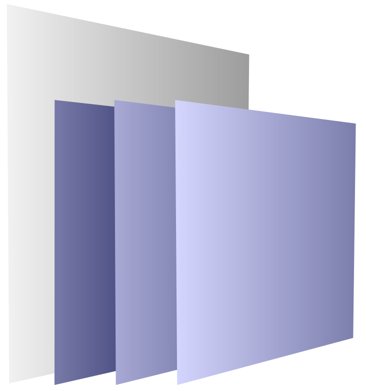


```
[1] 0
attr(,"class")
[1] "integer"
 "shinyActionButtonValue"
```

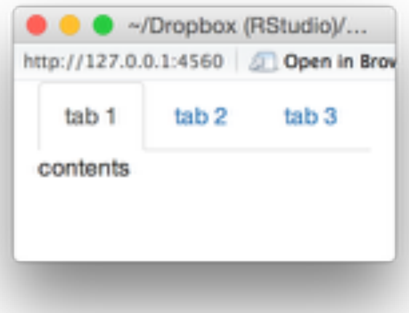
For each widget below, the Current Value(s) window displays the value that the widget provides to shinyServer. Notice that the values change as you interact with the widgets.

```
[1] TRUE
[1] "1"
2014-01-01
2015-06-02 to 2015-06-02
```

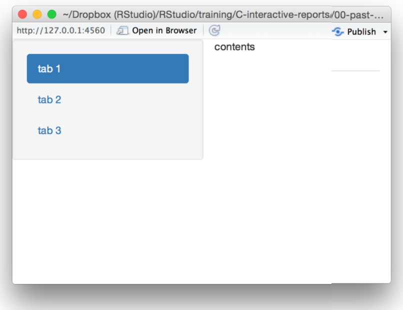
Panels group elements into a single unit for aesthetic or functional reasons



Use **tabPanel()** to create a stackable panel



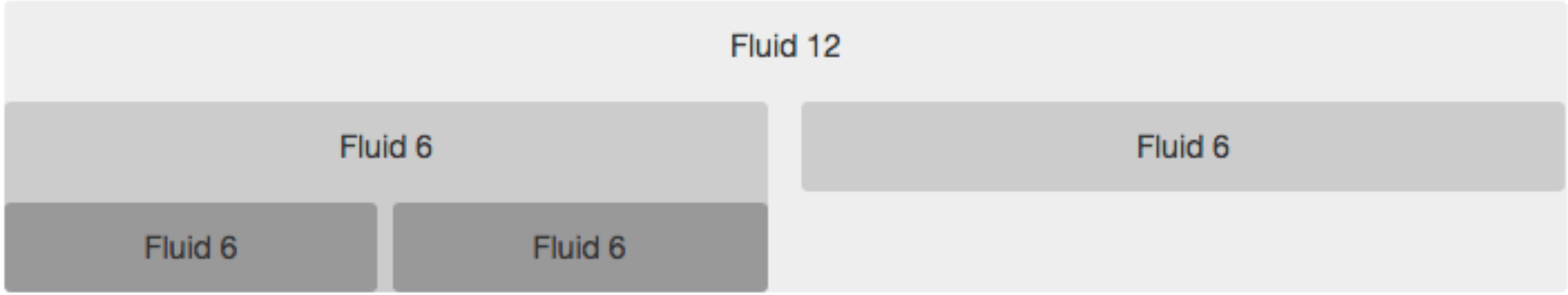
Use **tabsetPanel()** to arrange tab panels into a stack with tab navigation


~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/00-past-...
http://127.0.0.1:4560
Open in Browser
Publish
contents
tab 1
tab 2
tab 3

Use **navlistPanel()** to arrange tab panels into a stack with sidebar navigation

The Shiny Layout Guide

http://shiny.rstudio.com/articles/layout-guide.html



Fluid 12

Fluid 6

Fluid 6

Fluid 6

Fluid 6

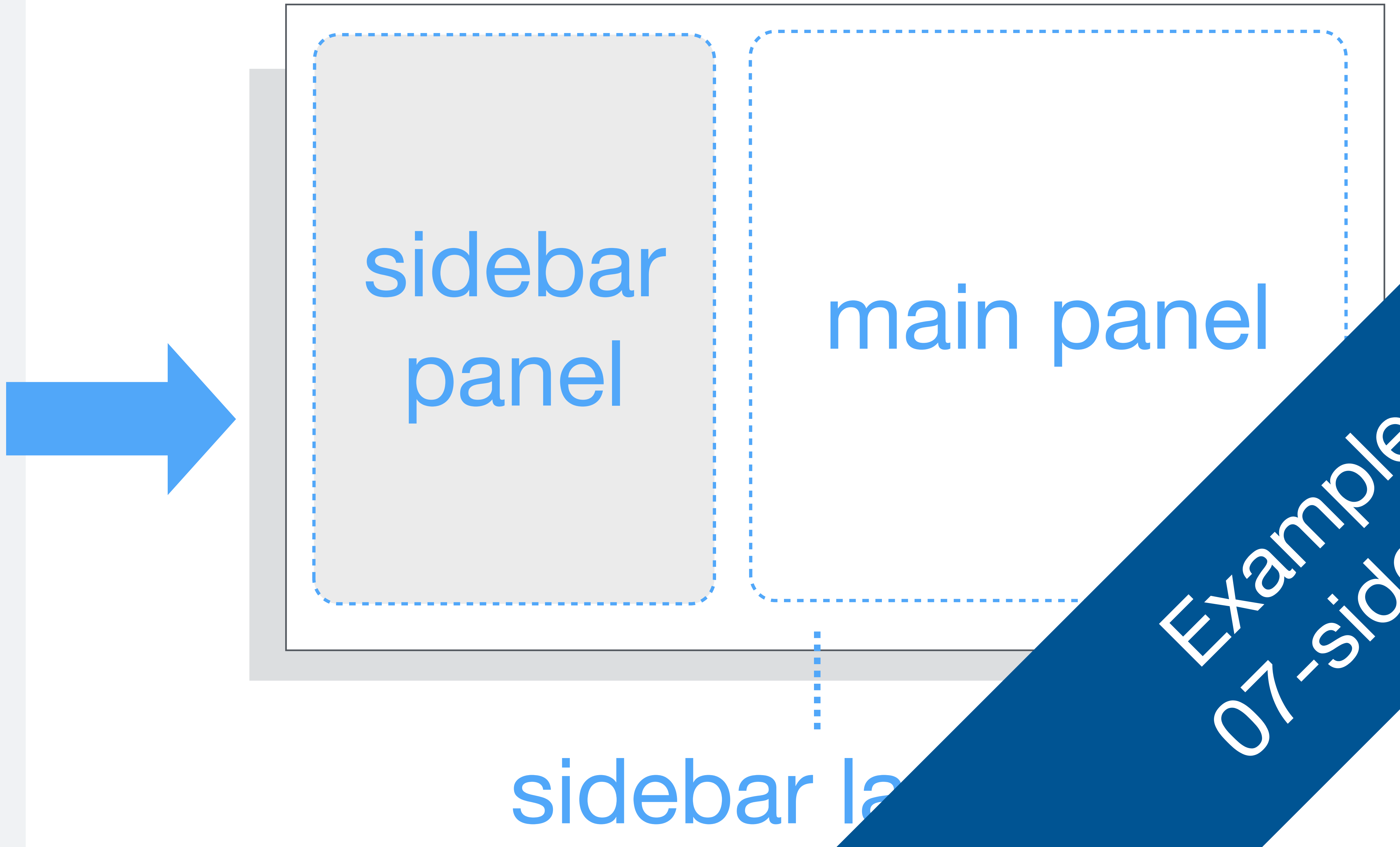
You can build sophisticated, customized layouts with Shiny's grid system.

Use a prepackaged layout

sidebarLayout()

Use with sidebarPanel() and mainPanel() to divide app into two sections.

```
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```



Example in 07-sidebar.R

# 🔒 fixedPage()

Creates a page that defaults to a width of 724, 940, or 1170 pixels (depending on browser window)

```
ui <- fixedPage(
  fixedRow(
    column(5, # etc.)
  )
)
```

Use with **fixedRow()**

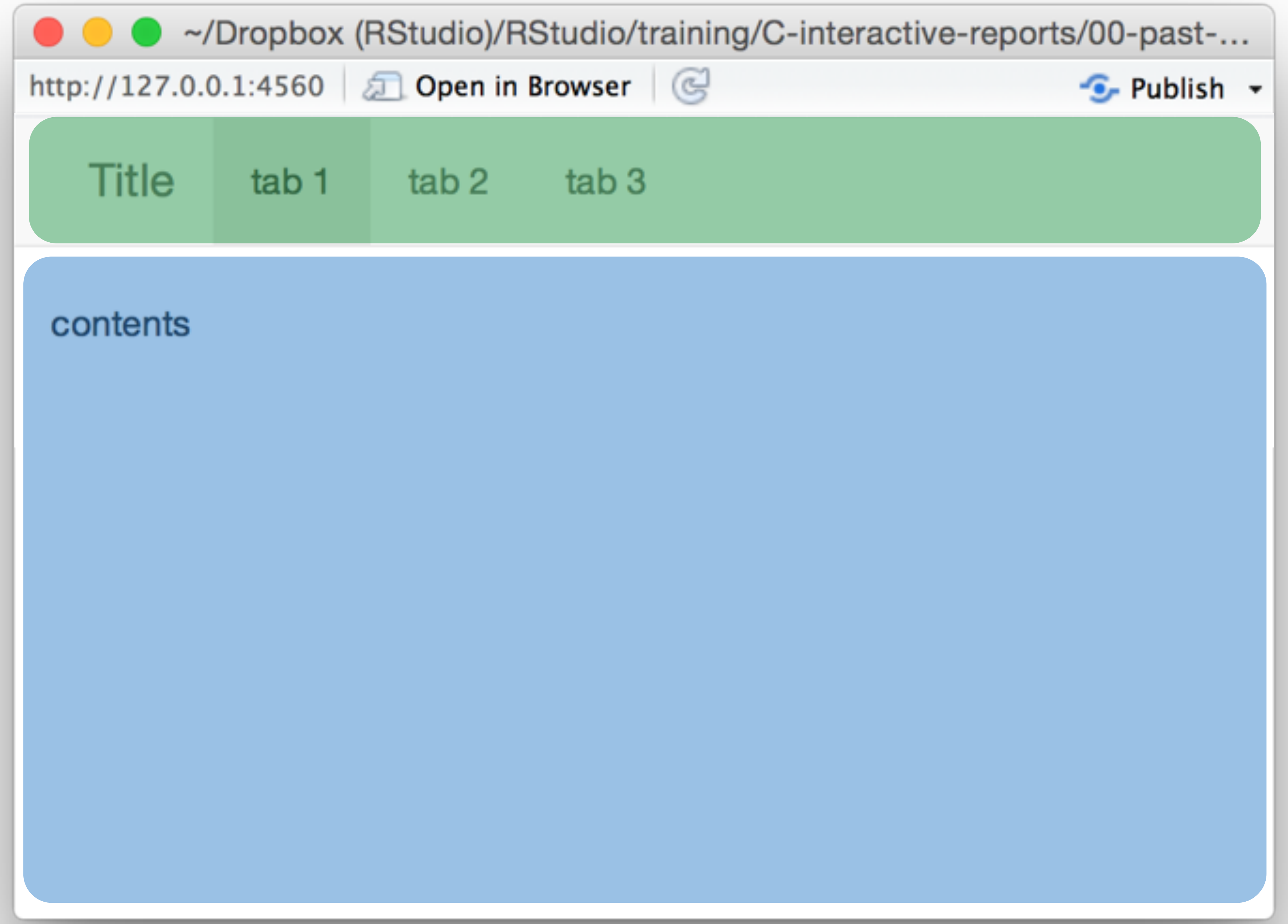
Compare to fluidPage() and fluidRow() which adjust to browser window

navbarPage()

navbarPage() combines tabs into a single *page*.

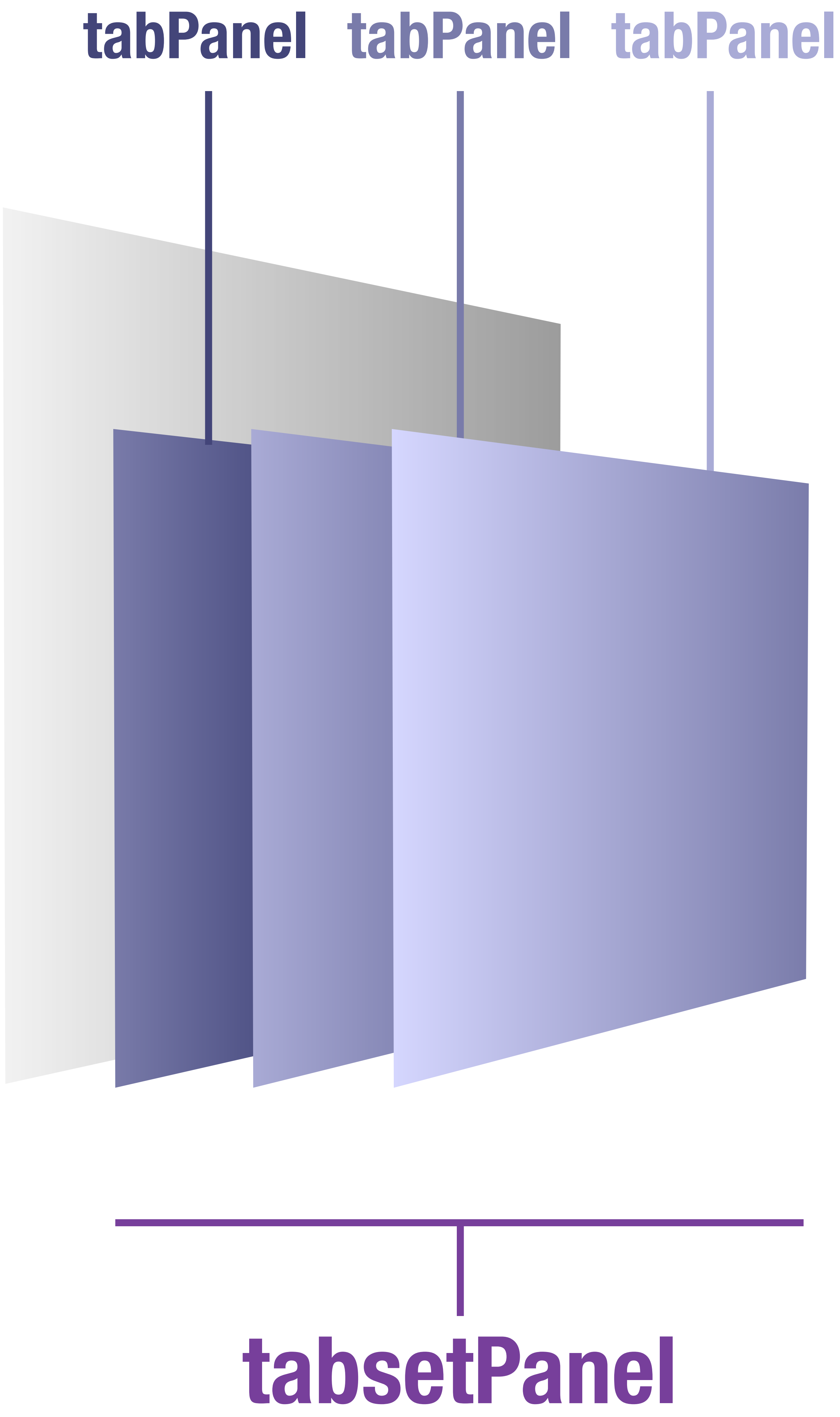
navbarPage() replaces fluidPage(). Requires title.

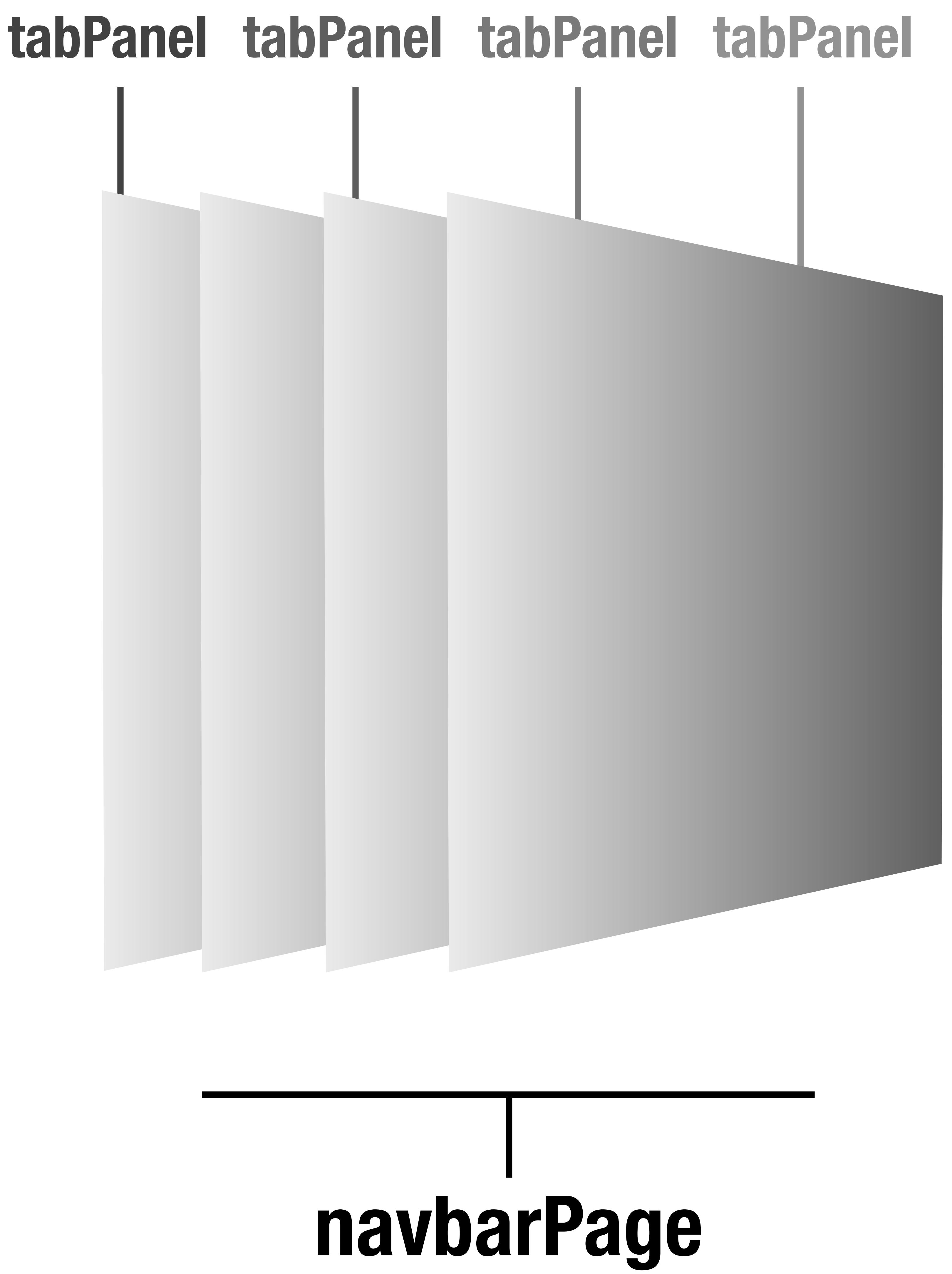
```
navbarPage(title = "Title",
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   tabPanel("tab 3", "contents")
)
```



Navigation

Content

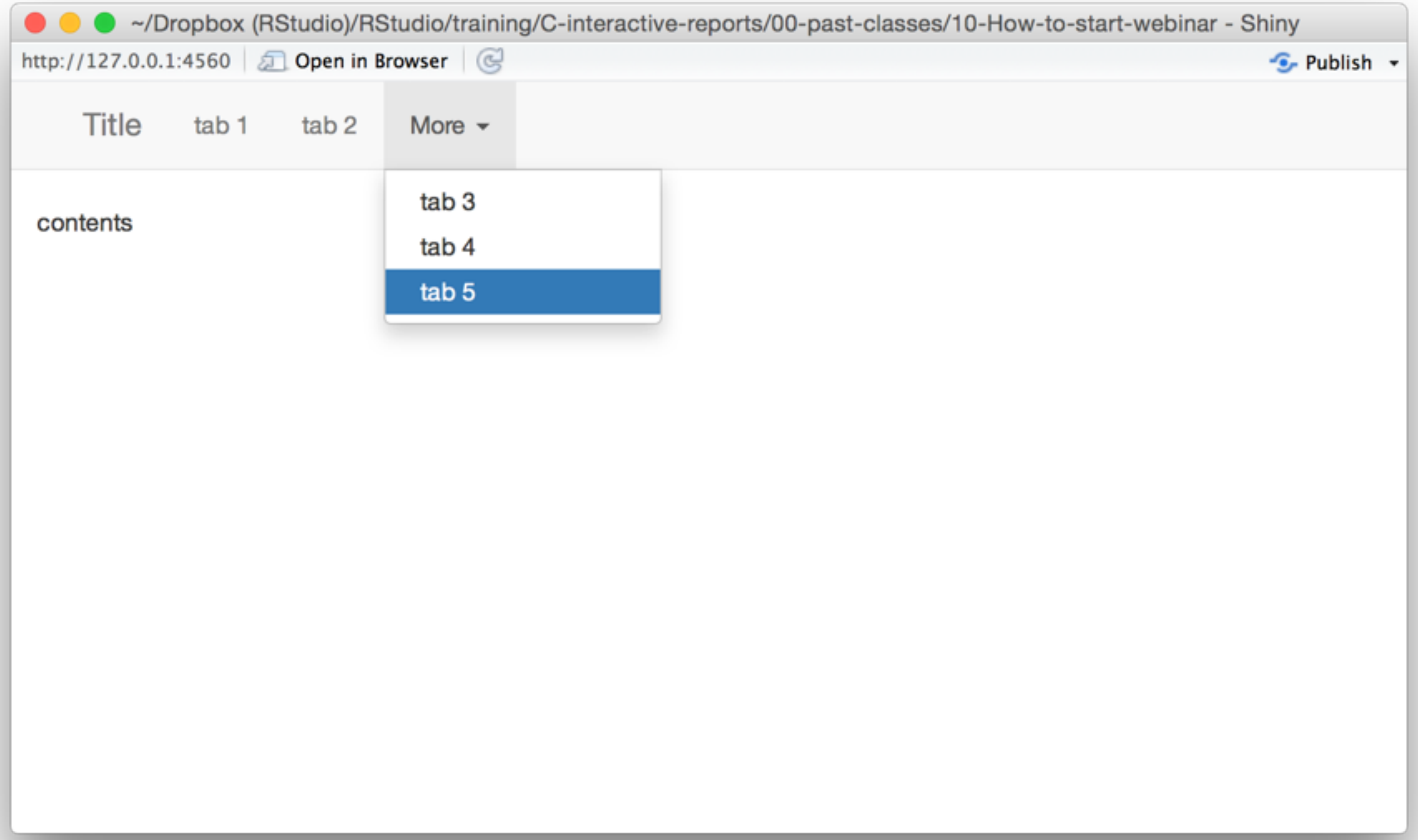

tabPanel
tabPanel
tabPanel
tabsetPanel


tabPanel
tabPanel
tabPanel
tabPanel
navbarPage

navbarMenu()

navbarMenu() combines tab links into a dropdown menu for navbarPage()

```
navbarPage(title = "Title",
   tabPanel("tab 1", "contents"),
   tabPanel("tab 2", "contents"),
   navbarMenu(title = "More",
     tabPanel("tab 3", "contents"),
     tabPanel("tab 4", "contents"),
     tabPanel("tab 5", "contents")
  )
)
```

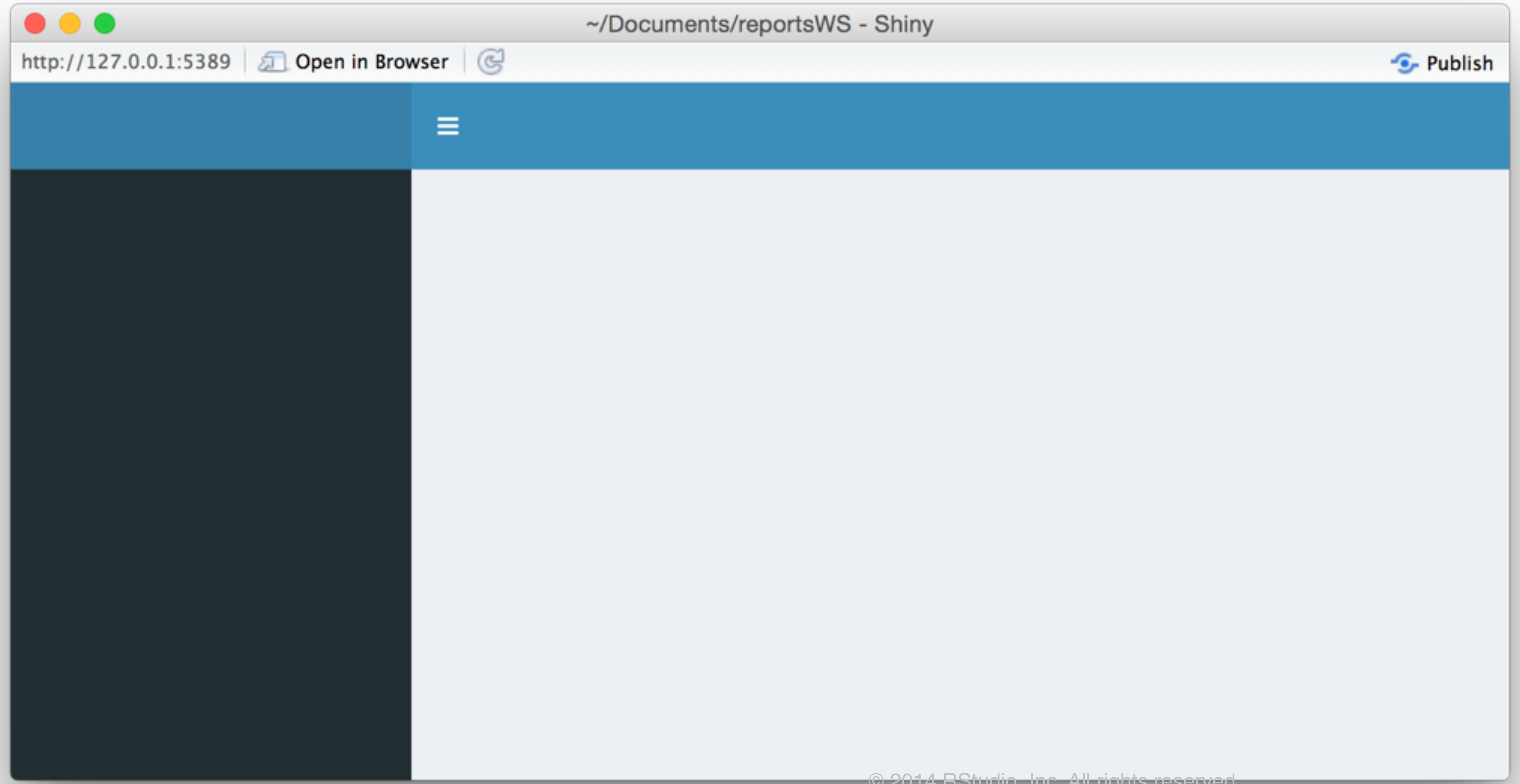

~/Dropbox (RStudio)/RStudio/training/C-interactive-reports/00-past-classes/10-How-to-start-webinar - Shiny
http://127.0.0.1:4560
Open in Browser
Publish
Title
tab 1
tab 2
More
tab 3
tab 4
tab 5

contents

dashboardPage()

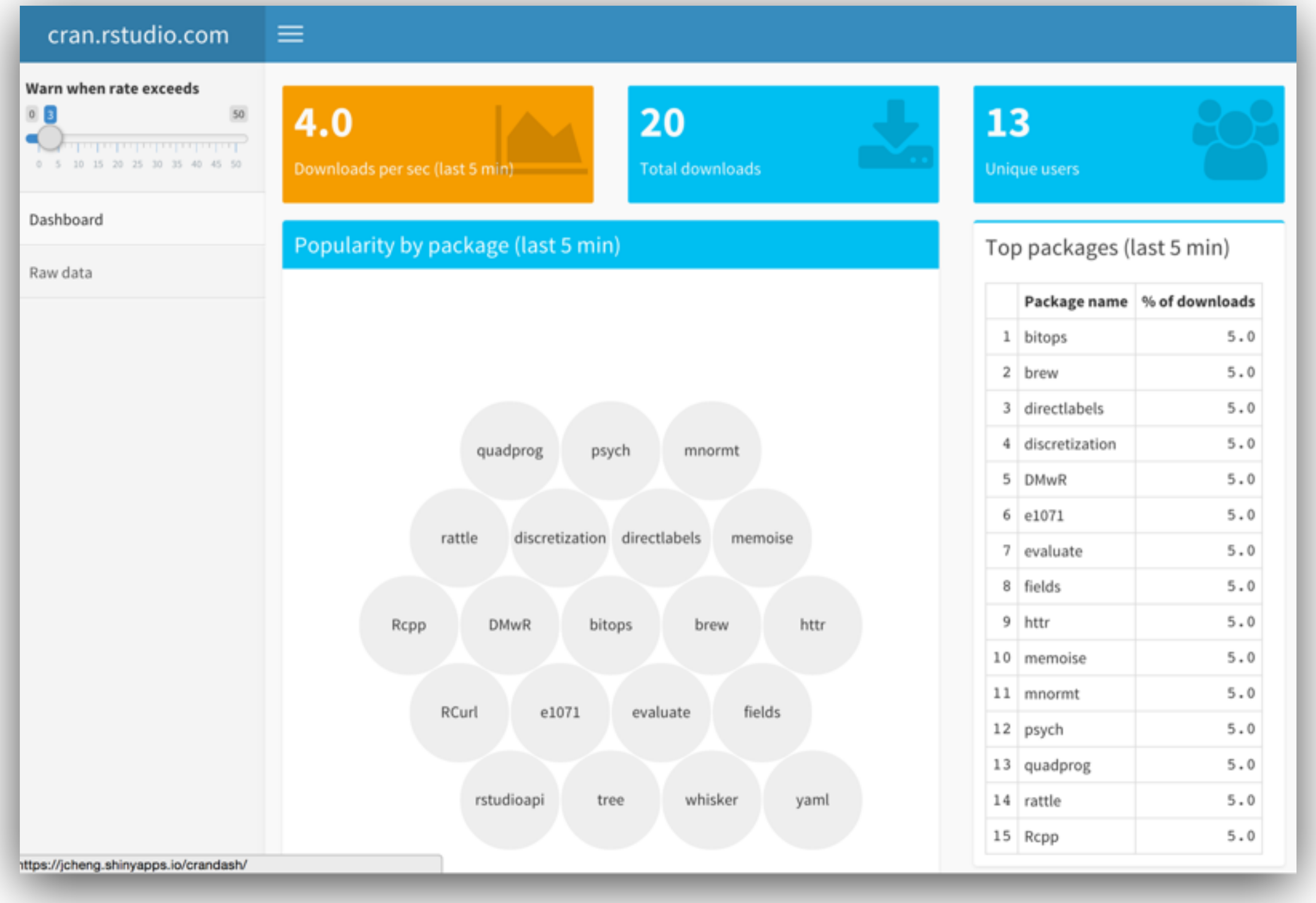
`dashboardPage()` comes in the shinydashboard package

```
library(shinydashboard)
ui <- dashboardPage(
  dashboardHeader(),
  dashboardSidebar(),
  dashboardBody()
)
```



shinydashboard

http://rstudio.github.io/shinydashboard/



4.0
Downloads per sec (last 5 min)

20
Total downloads

13
Unique users

Popularity by package (last 5 min)

Top packages (last 5 min)

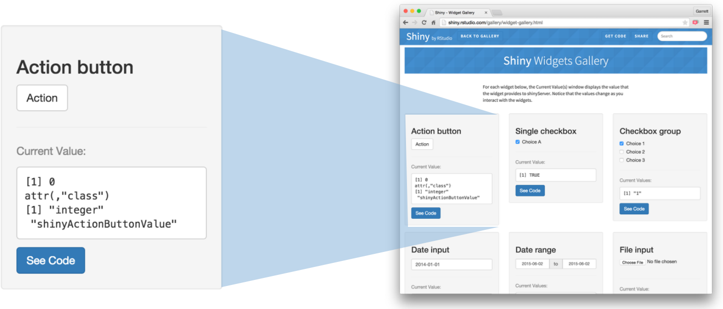
	Package name	% of downloads
1	bitops	5.0
2	brew	5.0
3	directlabels	5.0
4	discretization	5.0
5	DMwR	5.0
6	e1071	5.0
7	evaluate	5.0
8	fields	5.0
9	httr	5.0
10	memoise	5.0
11	mnormt	5.0
12	psych	5.0
13	quadprog	5.0
14	rattle	5.0
15	Rcpp	5.0

A package of layout functions for building administrative dashboards with Shiny

Dynamic Dashboards with Shiny Webinar:

www.rstudio.com/resources/webinars/

Recap: Prepackaged Layouts

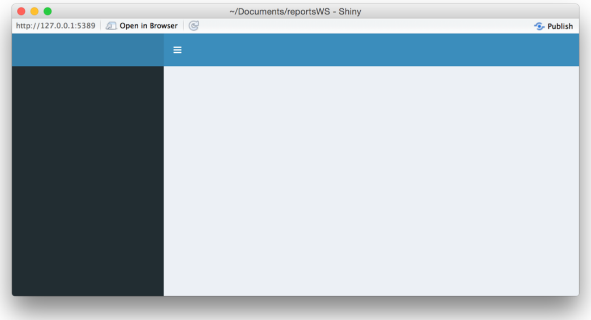


```
[1] 0
attr(,"class")
[1] "integer"
"shinyActionButtonValue"
```

Use **sidebarLayout()** with **sidebarPanel()** and **mainPanel()** to quickly create a sidebar design.

Use **fixedPanel()** with **fixedrow()** to create a fixed (non-fluid) design

Use **navbarPage()** with **navbarMenu()** to create "multipage" app



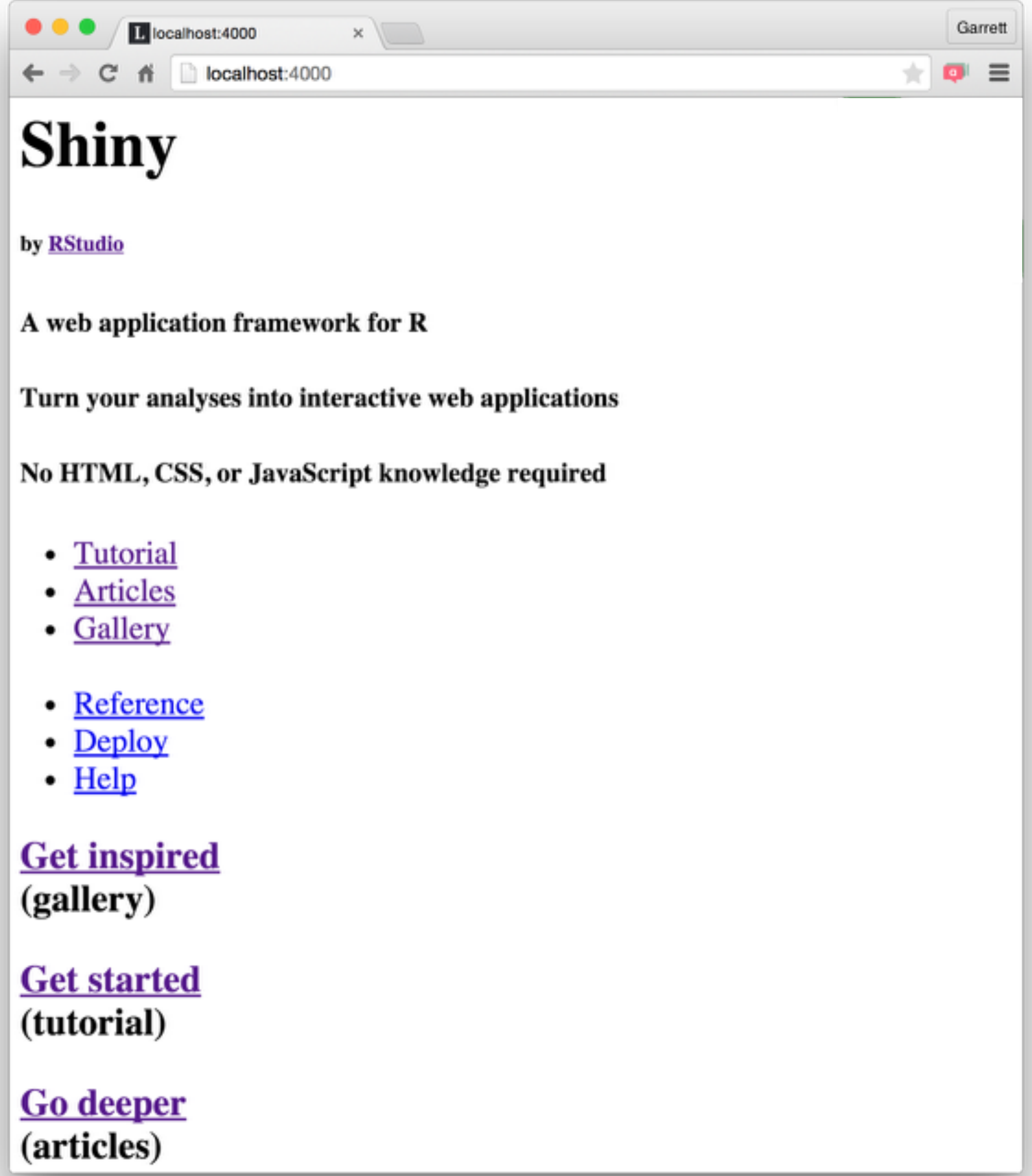
Use the **shinyDashboard** package to create dashboard layouts

Style with

CSS

What is CSS?

Cascading Style Sheets (CSS) are a framework for customizing the appearance of elements in a web page.



Shiny

by RStudio

A web application framework for R

Turn your analyses into interactive web applications

No HTML, CSS, or JavaScript knowledge required

- Tutorial
- Articles
- Gallery

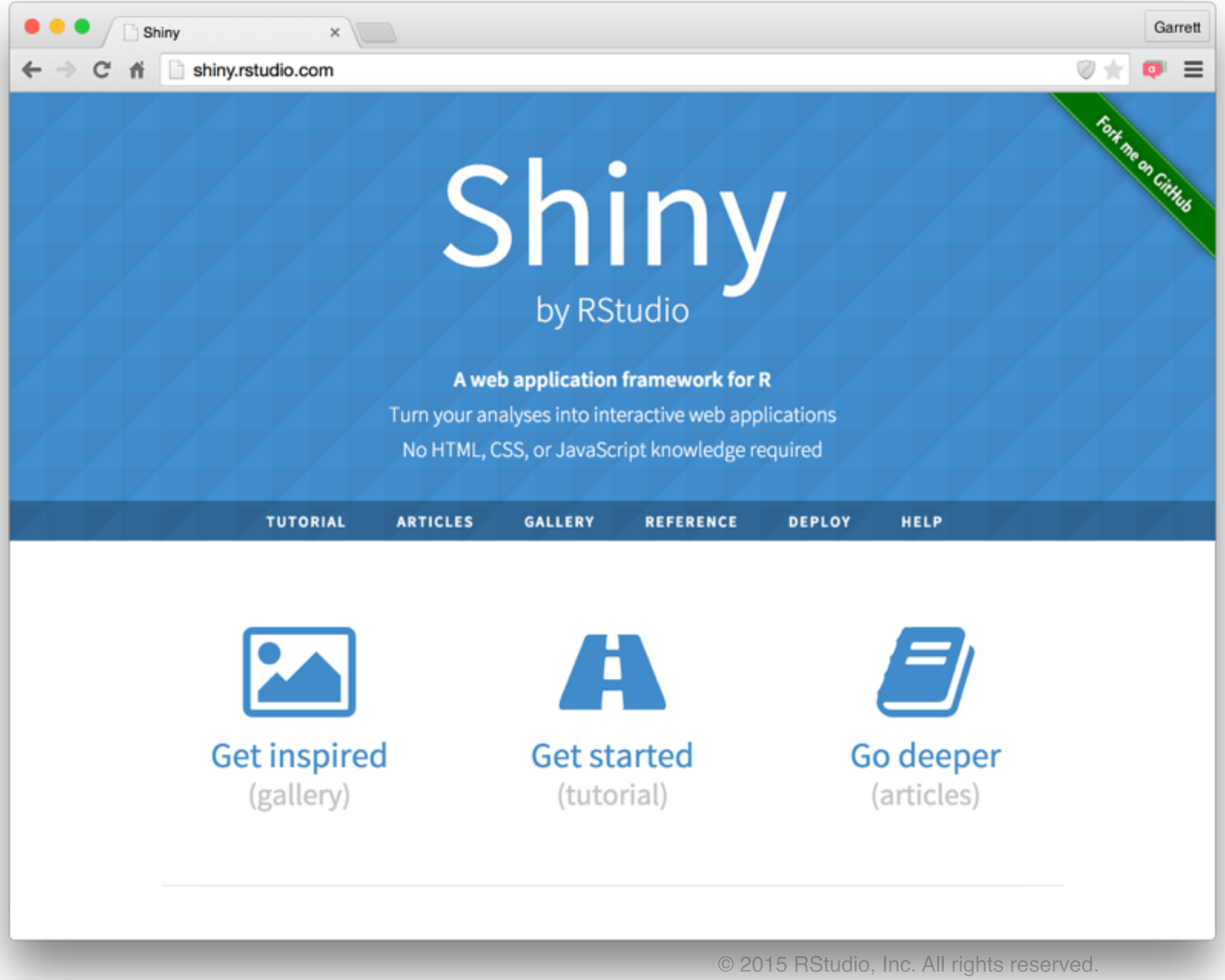
- Reference
- Deploy
- Help

Get inspired
(gallery)

Get started
(tutorial)

Go deeper
(articles)

+ CSS



Shiny

by RStudio

A web application framework for R

Turn your analyses into interactive web applications

No HTML, CSS, or JavaScript knowledge required

Get inspired
(gallery)

Get started
(tutorial)

Go deeper
(articles)

Style a web page in three ways:

1 Link to an external CSS file

2 Write global CSS in header

3 Write individual CSS in a tag’s style attribute

Overrides

Match styling to:

1 Tag

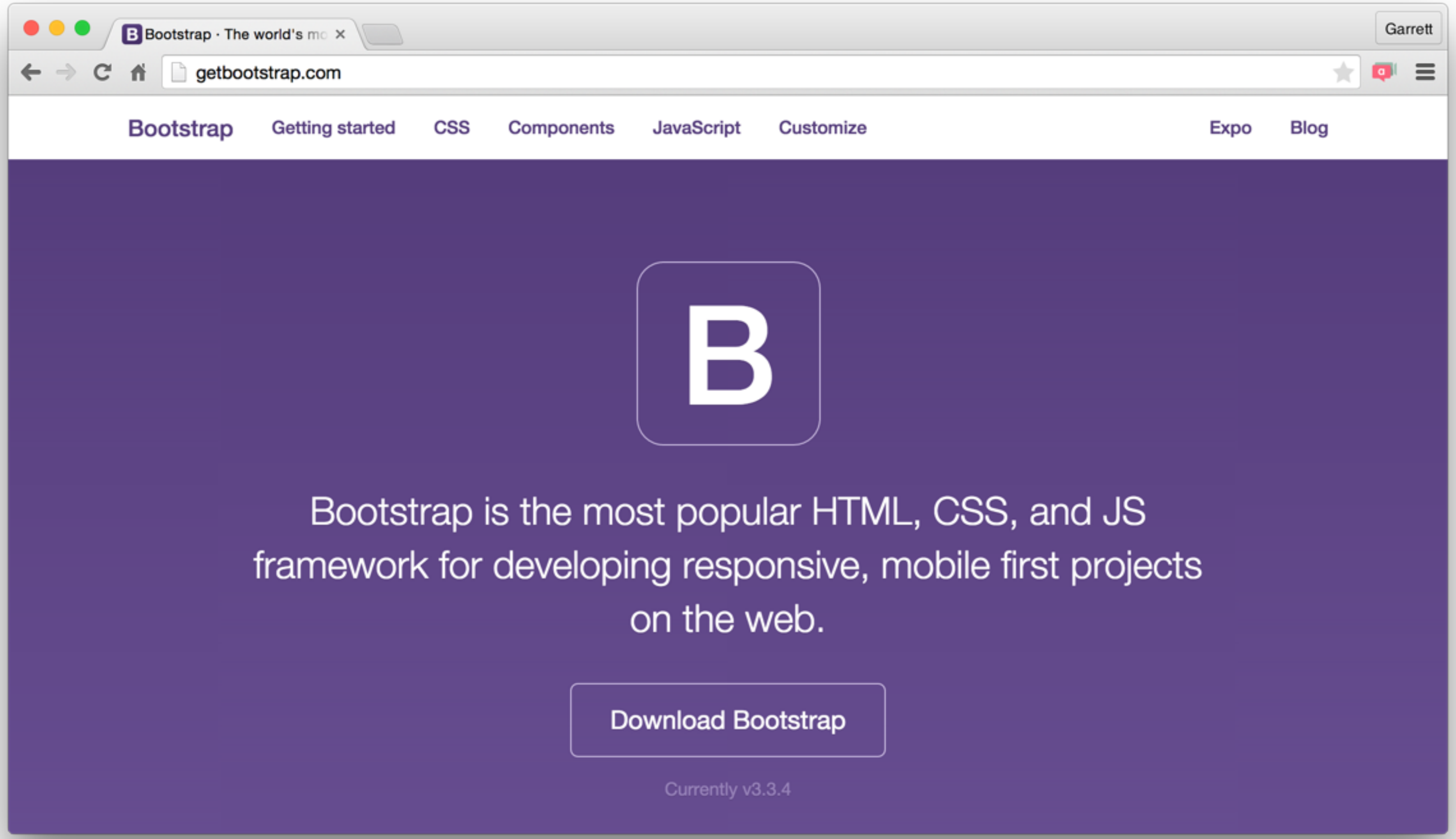
2 Class

3 id

Overrides

Bootstrap

Shiny uses the Bootstrap 3 CSS framework, getbootstrap.com



Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Bootstrap

Shiny uses the Bootstrap 3 CSS framework, getbootstrap.com

```
fluidPage()
```

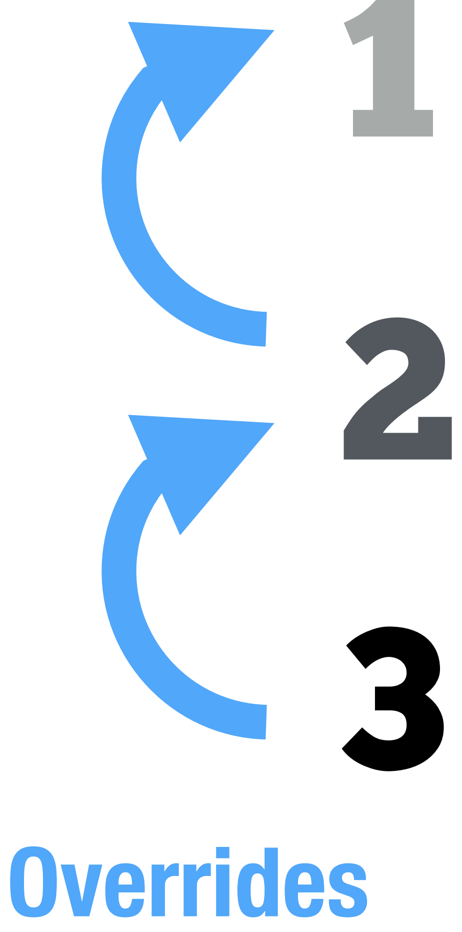
```
<div class="container-fluid"></div>
```

Style a Shiny app in three ways:

1 Link to an external CSS file

2 Write global CSS in header

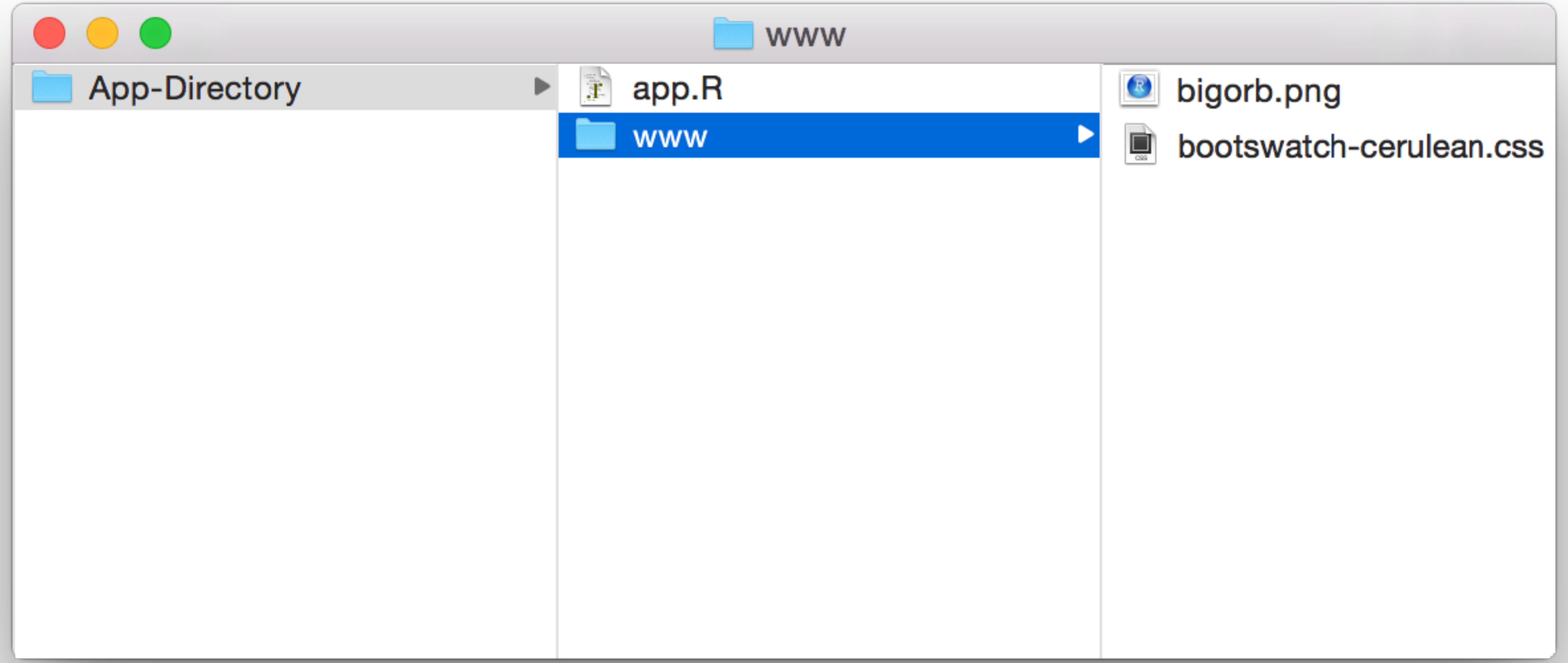
3 Write individual CSS in a tag’s style attribute



*CSS designed to work with Bootstrap 3 will work best with Shiny.

1 Link to an external CSS file

Place .css files in the **www** folder of your app directory



Shiny will share a file with your user's browser if the file appears in www. Shiny will not share files that you do not place in www.

1 Link to an external CSS file

Set the theme argument of fluidPage() to the .css filename, or...

```
ui <- fluidPage(
  theme = "bootswatch-cerulean.css",
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```

1 Link to an external CSS file

Or place a link in the app's header with to the file with **tags$head()** and **tags$link()**

```
ui <- fluidPage(
  tags$head(



  )
)
```

```
<head>

</head>
<body>
  <div class="container-fluid">
  </div>
</body>
```

1 Link to an external CSS file

Or place a link in the app's header with to the file with **tags$head()** and **tags$link()**

```
ui <- fluidPage(
  tags$head(
   tags$link(
    rel = "stylesheet",
    type = "text/css",
    href = "file.css"
   )
  )
)
```

```
<head>
  <link type="text/css" rel="stylesheet" href="file.css"/>
</head>
<body>
  <div class="container-fluid">
  </div>
</body>
```

2 Write global CSS in header

Write global CSS with **tags$head()** and **tags$style()** and **HTML()**

```
ui <- fluidPage(
  tags$head(
   tags$style(HTML("
     p {
       color:red;
     }
   "))
  )
)
```

```
<head>

  <style>
    p {
      color:red;
    }
  </style>

</head>

<body>

  <div class="container-fluid">

  </div>

</body>
```

2 Write global CSS in header

Or save the CSS as a file in your app directory and include it with **includeCSS()**

```
ui <- fluidPage(
  includeCSS("file.css")
)
```

```
<head>
  <style>
    p {
      color:red;
    }
  </style>
</head>
<body>
  <div class="container-fluid">
  </div>
</body>
```

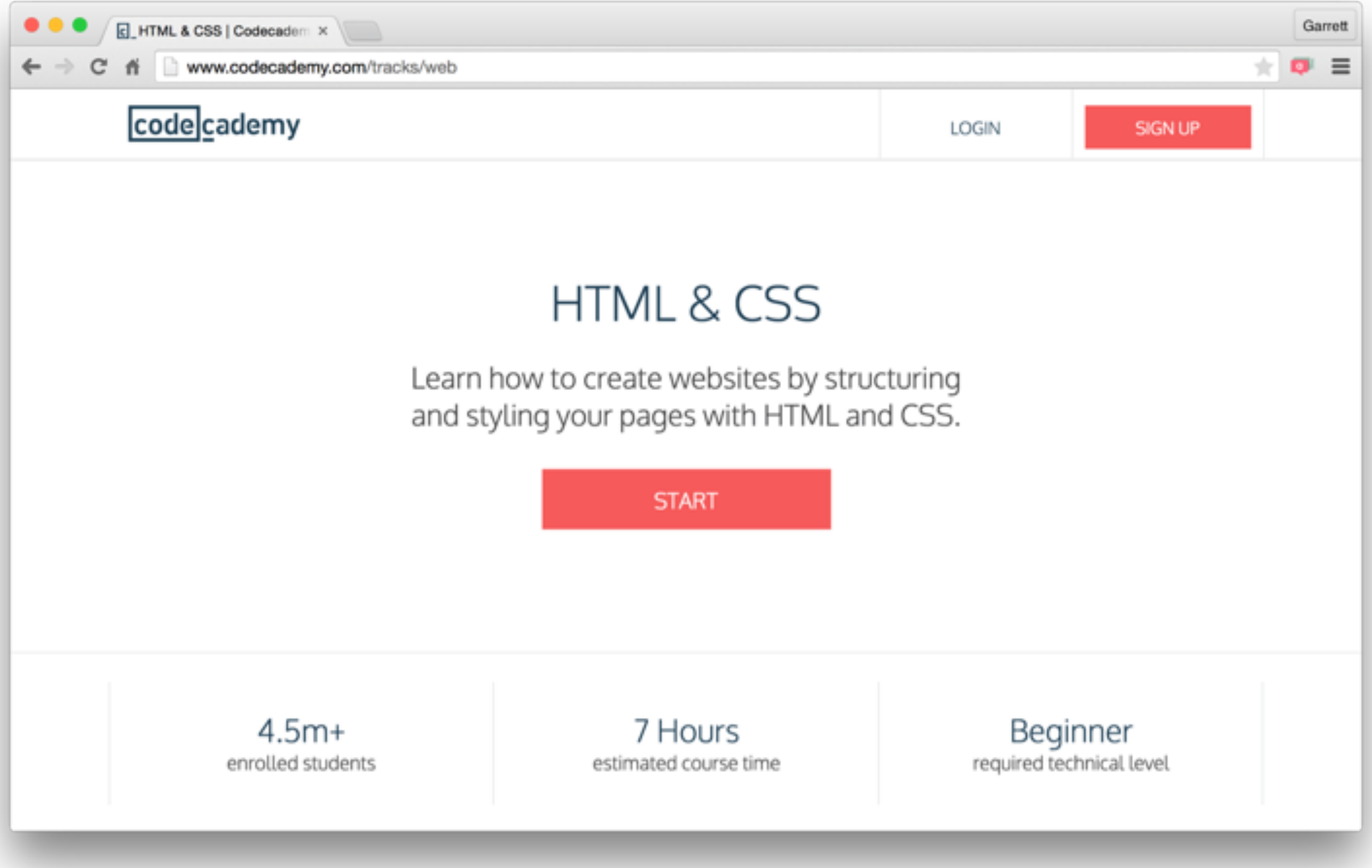
3 Write individual CSS in a tag’s style attribute

Set the style argument in Shiny’s tag functions

```
ui <- fluidPage(
  tags$h1("Title", style = "color:red;")
)
```

To learn more about CSS & HTML

http://www.codecademy.com/tracks/web



HTML & CSS

Learn how to create websites by structuring and styling your pages with HTML and CSS.

4.5m+ enrolled students

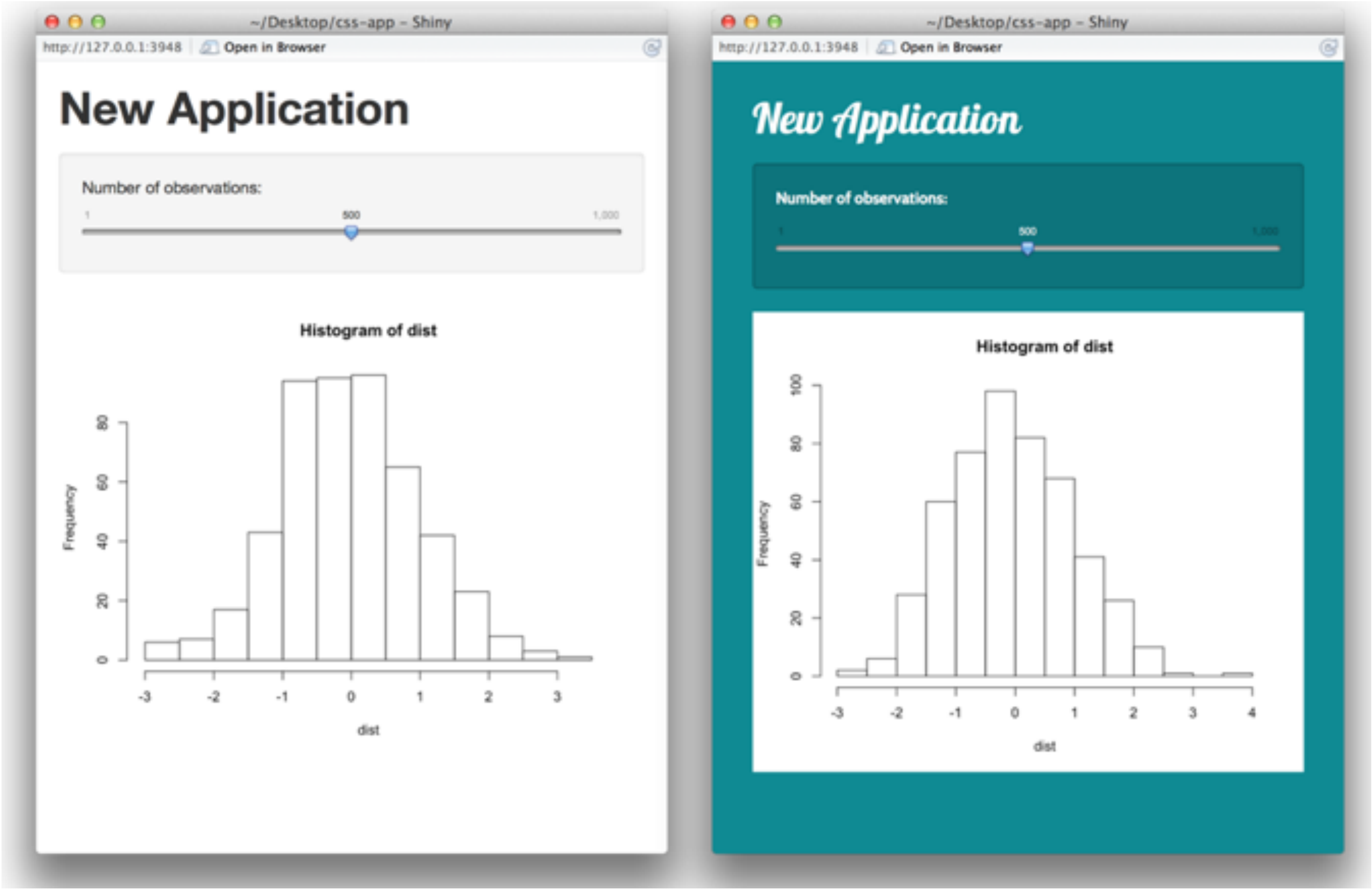
7 Hours estimated course time

Beginner required technical level

I recommend the free codecademy tutorial

Customize your apps with HTML, CSS, and Javascript

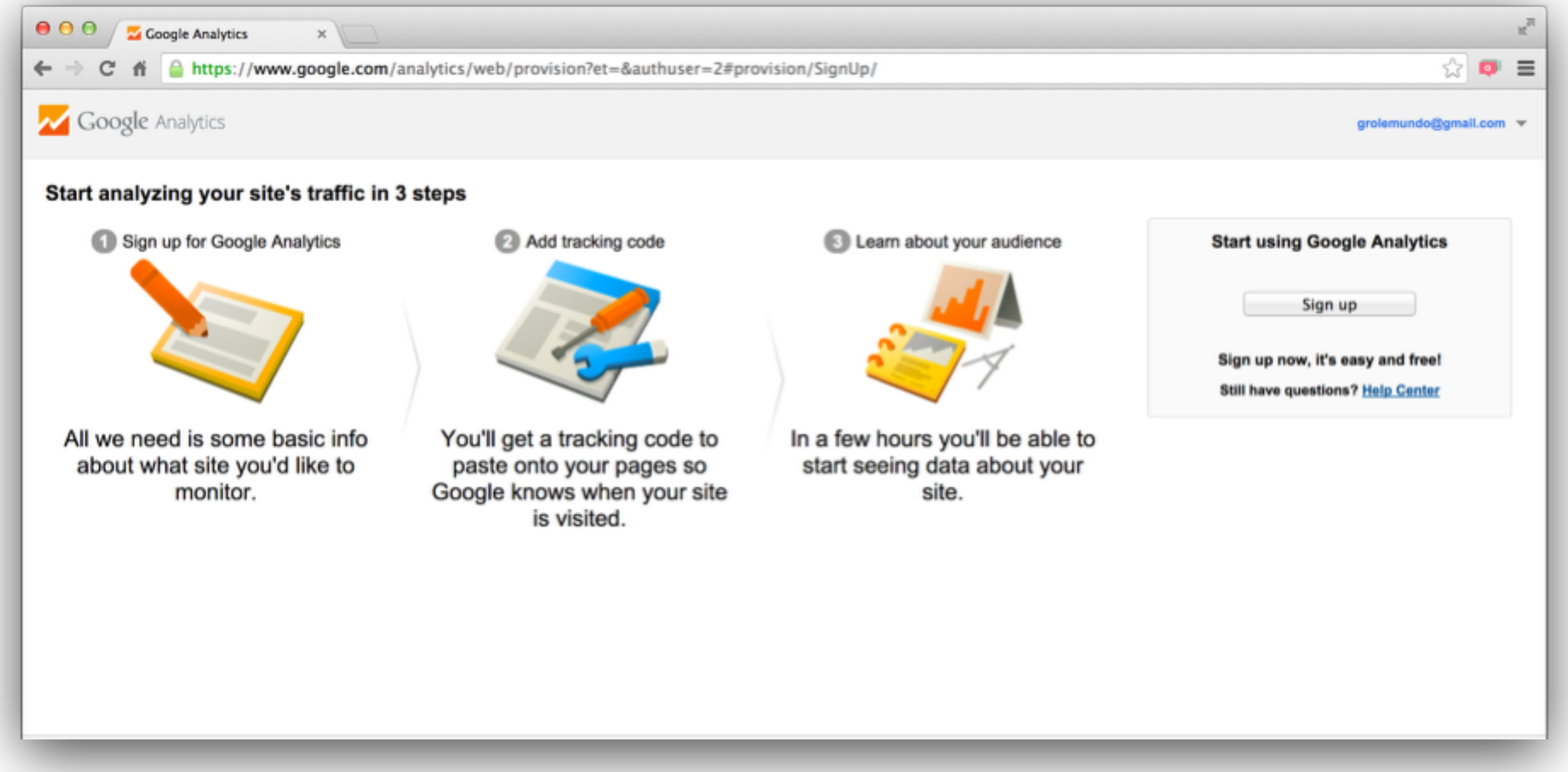
http://shiny.rstudio.com/articles/css.html



You can pair any app with whatever web technologies you wish. The above guide explains how to style your app with CSS.

Add Google Analytics to a Shiny app

http://shiny.rstudio.com/articles/google-analytics.html



Google Analytics
https://www.google.com/analytics/web/provision?et=&authuser=2#provision/SignUp/
Google Analytics
grolemundo@gmail.com

Start analyzing your site's traffic in 3 steps

1 Sign up for Google Analytics

All we need is some basic info about what site you'd like to monitor.

2 Add tracking code

You'll get a tracking code to paste onto your pages so Google knows when your site is visited.

3 Learn about your audience

In a few hours you'll be able to start seeing data about your site.

Start using Google Analytics

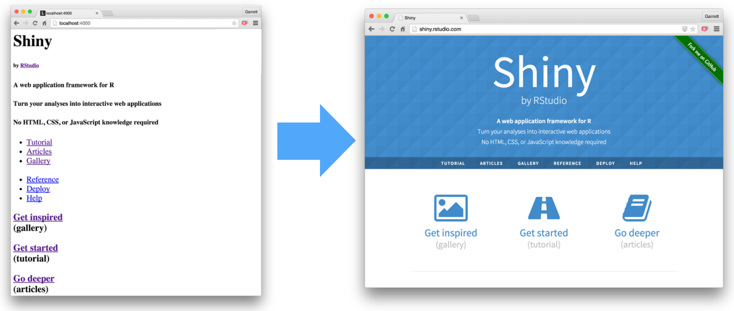
Sign up

Sign up now, it's easy and free!

Still have questions? Help Center

A case study in using jQuery to track visitor actions with Google Analytics

Recap: Style with CSS

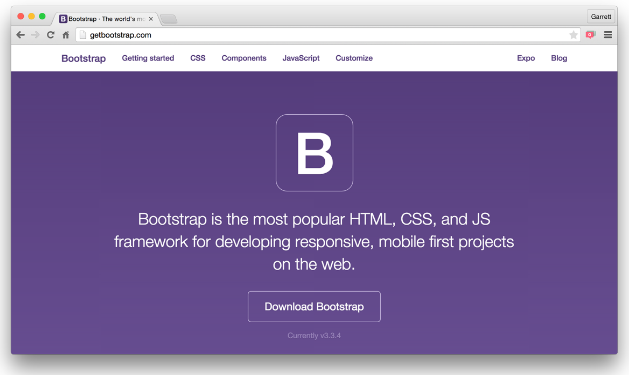


A web application framework for R

Turn your analyses into interactive web applications

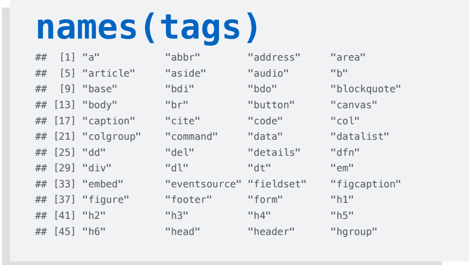
No HTML, CSS, or JavaScript knowledge required

Style Shiny apps like web pages: with CSS.



Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web.

Shiny's general CSS classes come from the **bootstrap 3** framework



```
names(tags)
##  [1] "a"           "abbr"        "address"     "area"
##  [5] "article"     "aside"       "audio"       "b"
##  [9] "base"        "bdi"         "bdo"         "blockquote"
## [13] "body"        "br"          "button"      "canvas"
## [17] "caption"     "cite"        "code"        "col"
## [21] "colgroup"    "command"     "data"        "datalist"
## [25] "dd"          "del"         "details"     "dfn"
## [29] "div"         "dl"          "dt"          "em"
## [33] "embed"       "eventsource" "fieldset"    "figcaption"
## [37] "figure"      "footer"      "form"        "h1"
## [41] "h2"          "h3"          "h4"          "h5"
## [45] "h6"          "head"        "header"      "hgroup"
```

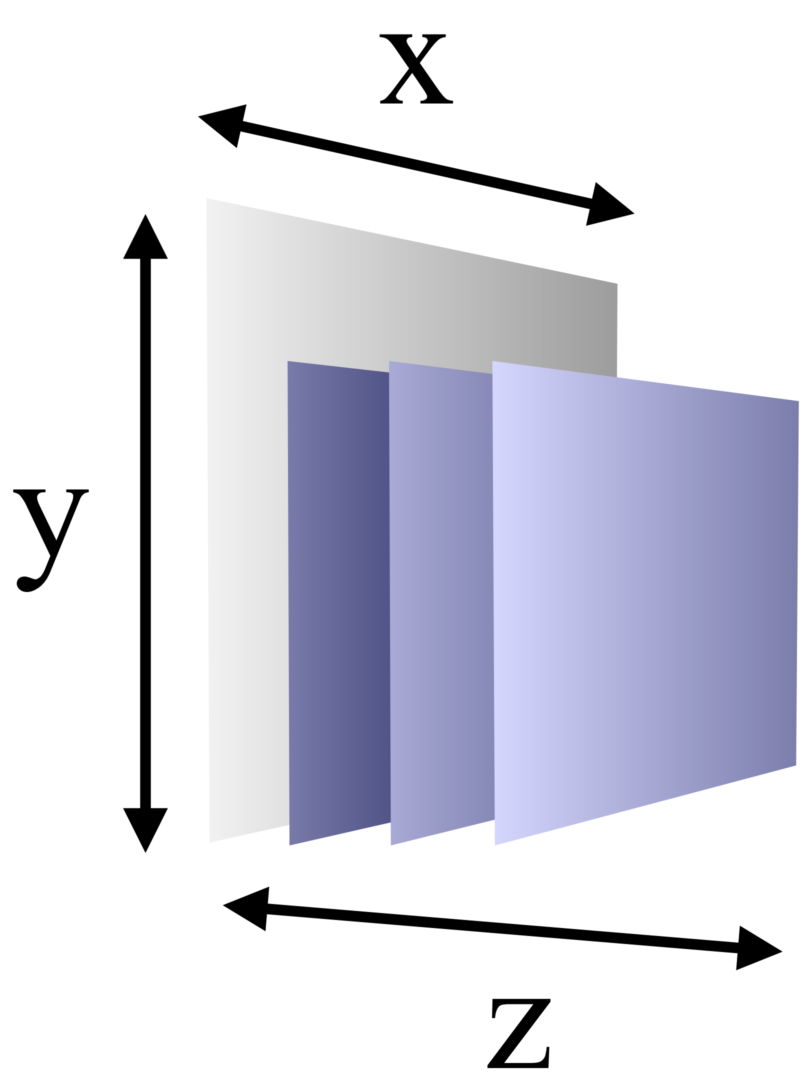
You can recreate HTML methods for including CSS with **tags$head()**, **tags$link()** and **tags$style()**

You now how to

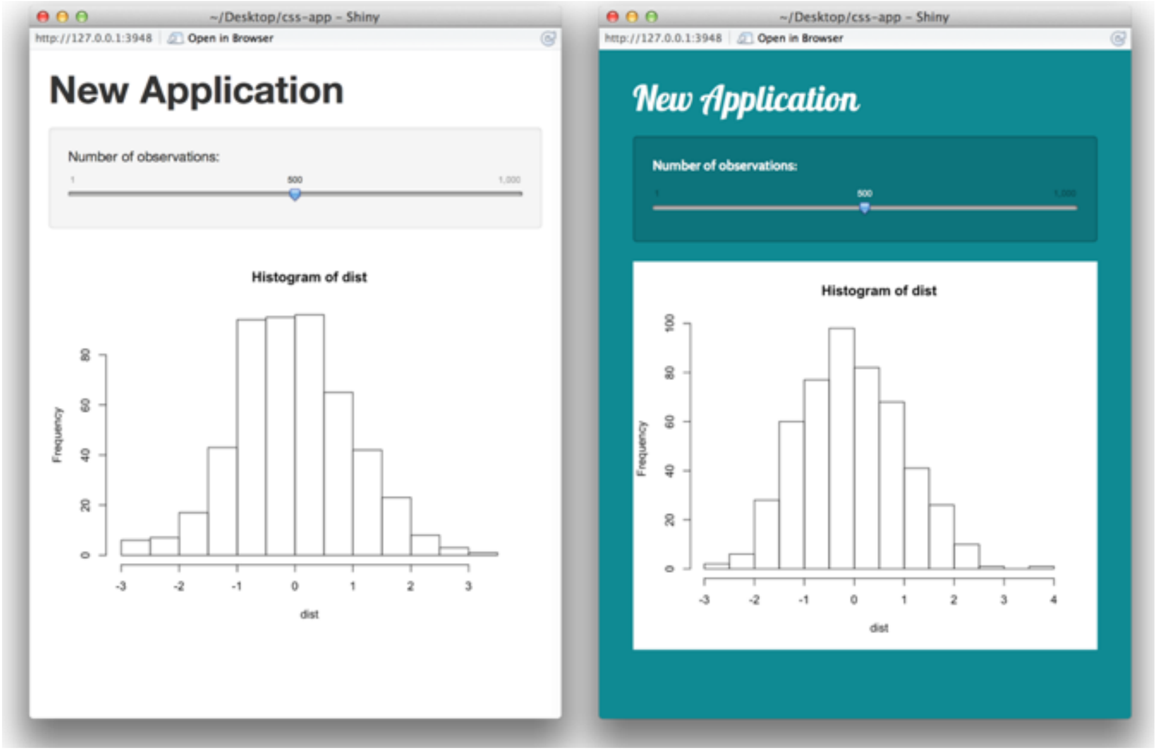
My Shiny App

See other apps in the Shiny Showcase

Add static elements



Lay out elements



New Application

Number of observations:

Histogram of dist

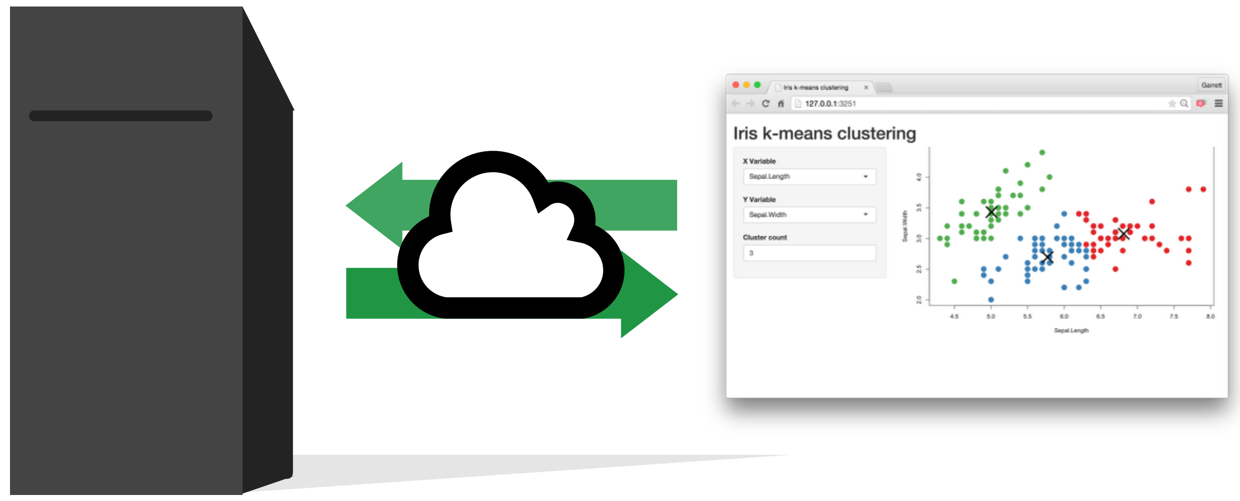
New Application

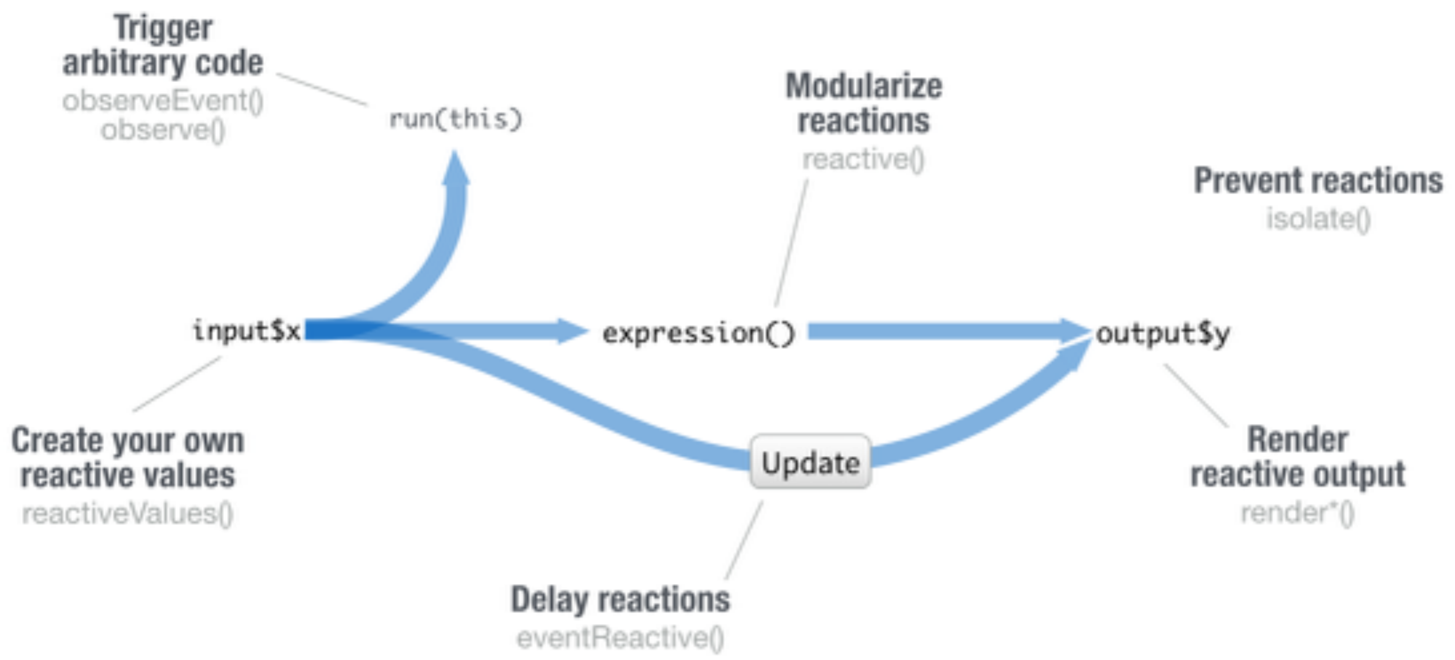
Number of observations:

Histogram of dist

Style elements with CSS

How to start with Shiny





Trigger arbitrary code
observeEvent()
observe()

run(this)

Modularize reactions
reactive()

Prevent reactions
isolate()

input$x

expression()

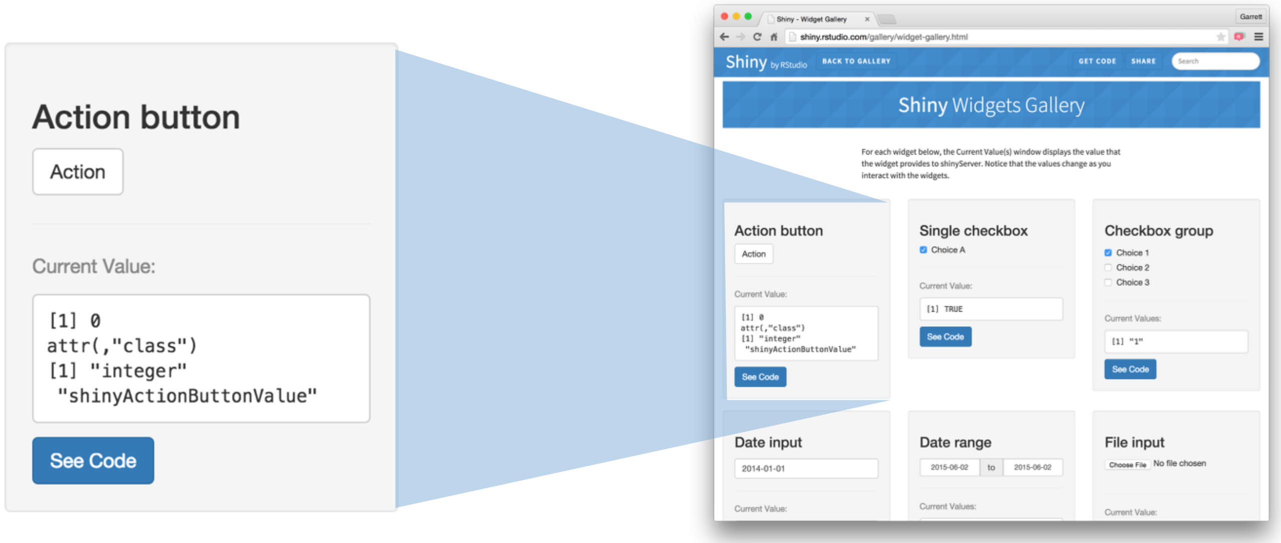
output$y

Update

Create your own reactive values
reactiveValues()

Render reactive output
render*()

Delay reactions
eventReactive()



Action button

Action

Current Value:

```
[1] 0
attr(,"class")
[1] "integer"
"shinyActionButtonValue"
```

See Code

Build and share apps
(Part 1)

Control reactions
(Part 2)

Customize appearance
(Part 3)

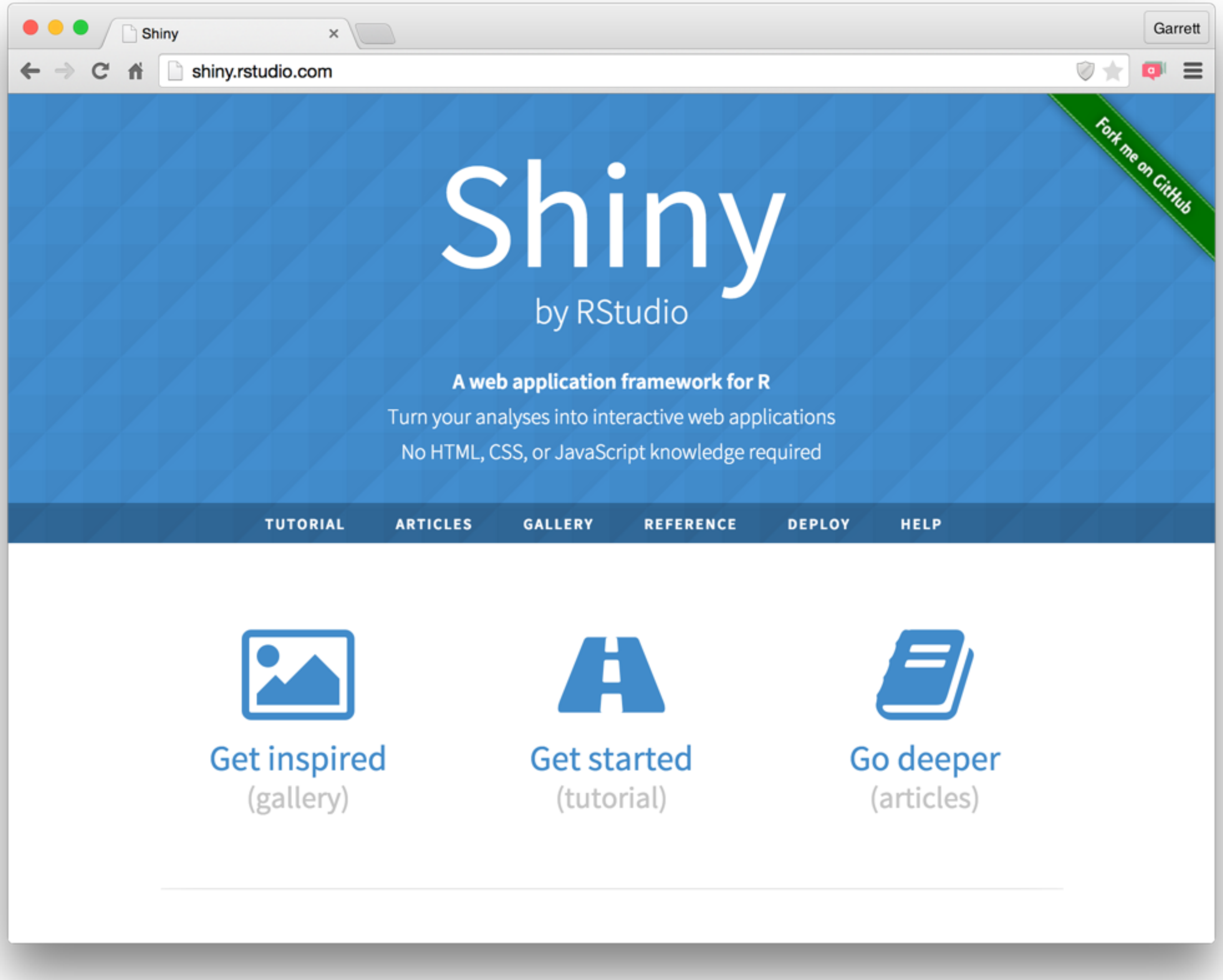
Where to go from here

Experiment and practice

build your own apps

The Shiny Development Center

shiny.rstudio.com



Shiny

by RStudio

A web application framework for R

Turn your analyses into interactive web applications

No HTML, CSS, or JavaScript knowledge required