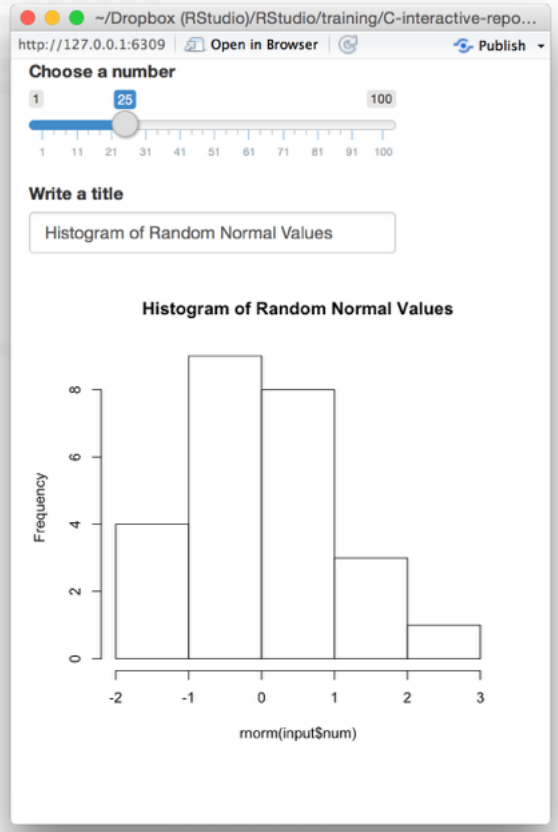
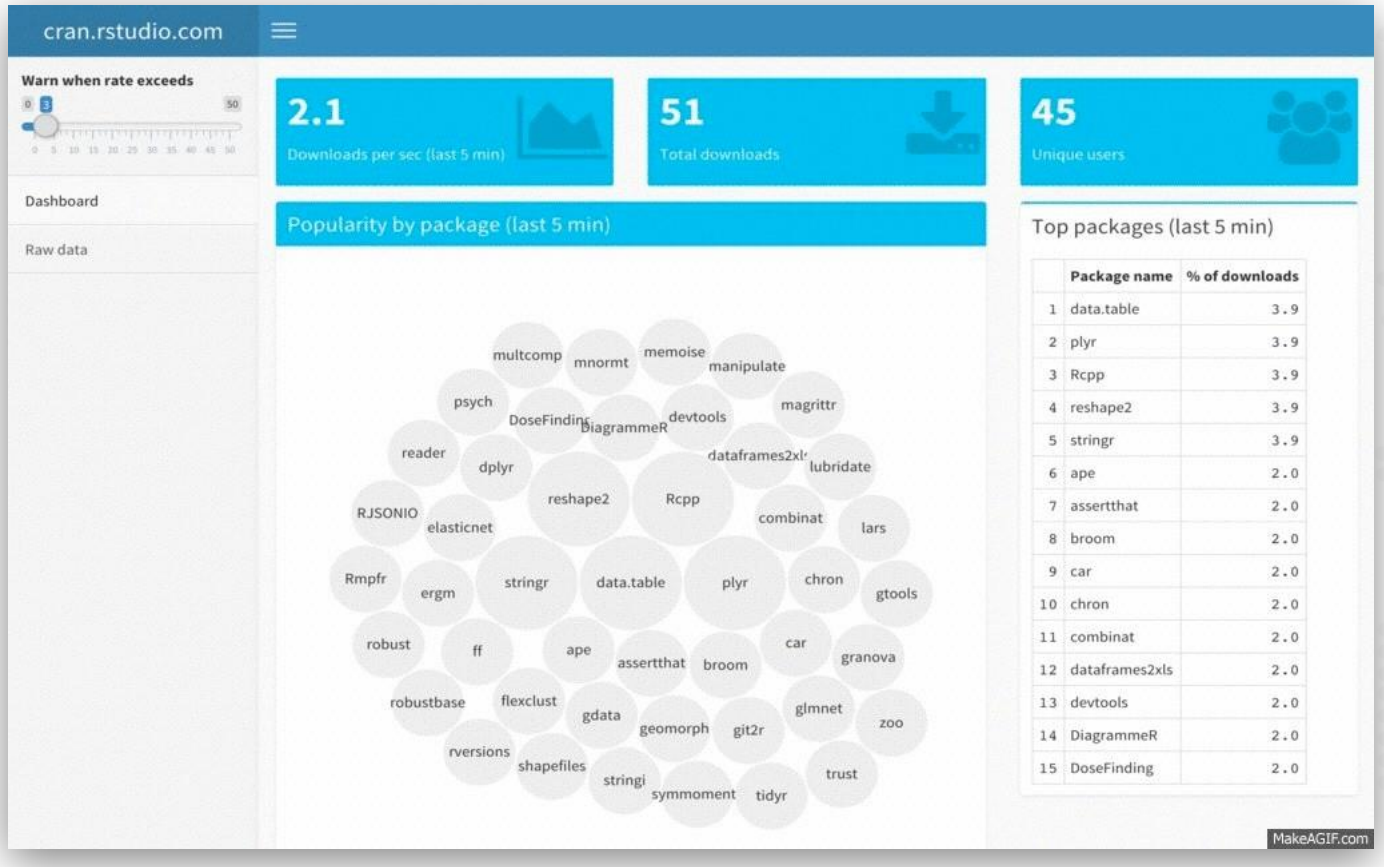
NobleProg

Customizing Appearance

User Interface





Work with the HTML UI

NobleProg

The HTML that builds the user interface for your app

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

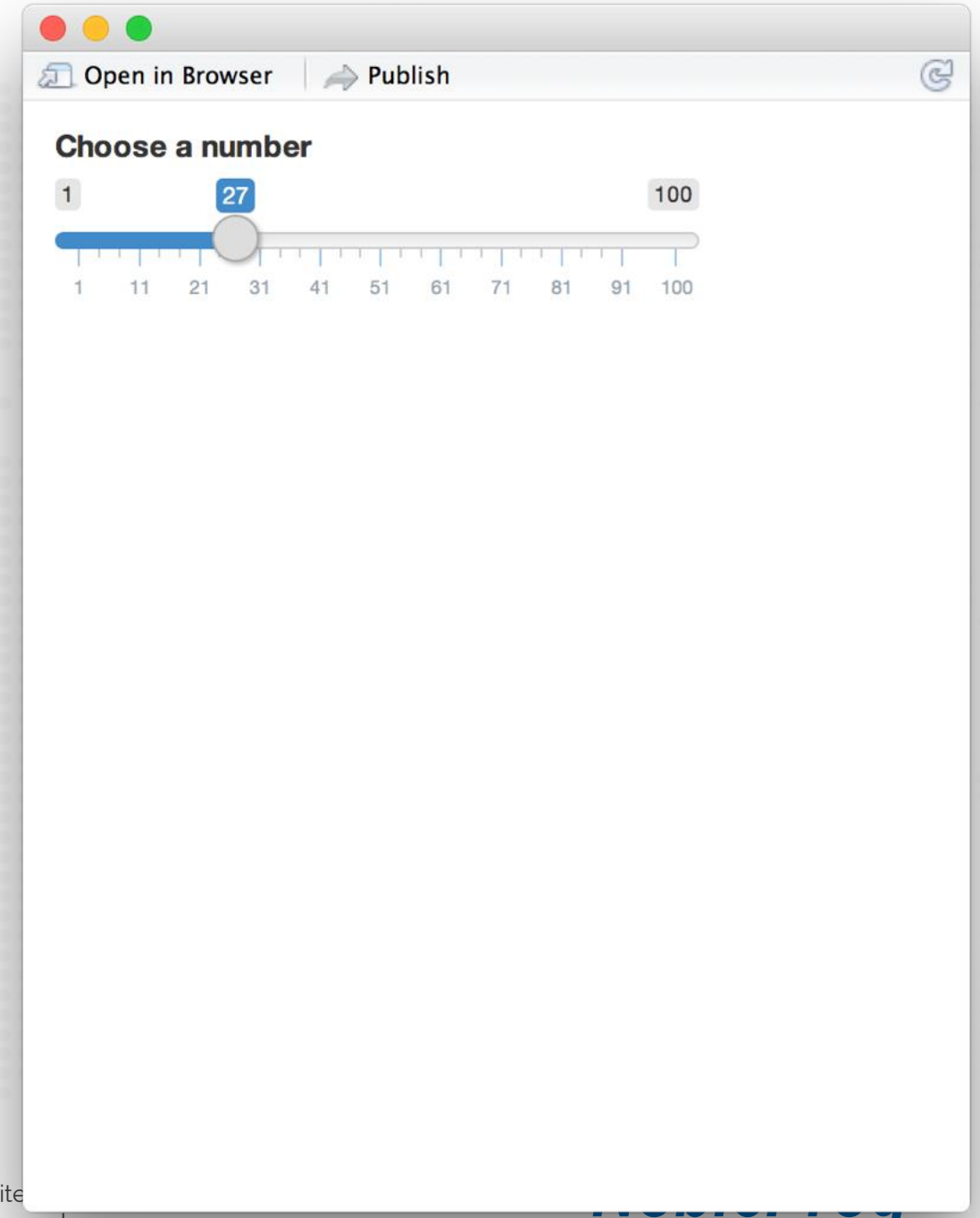
```
fluidPage()
```

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

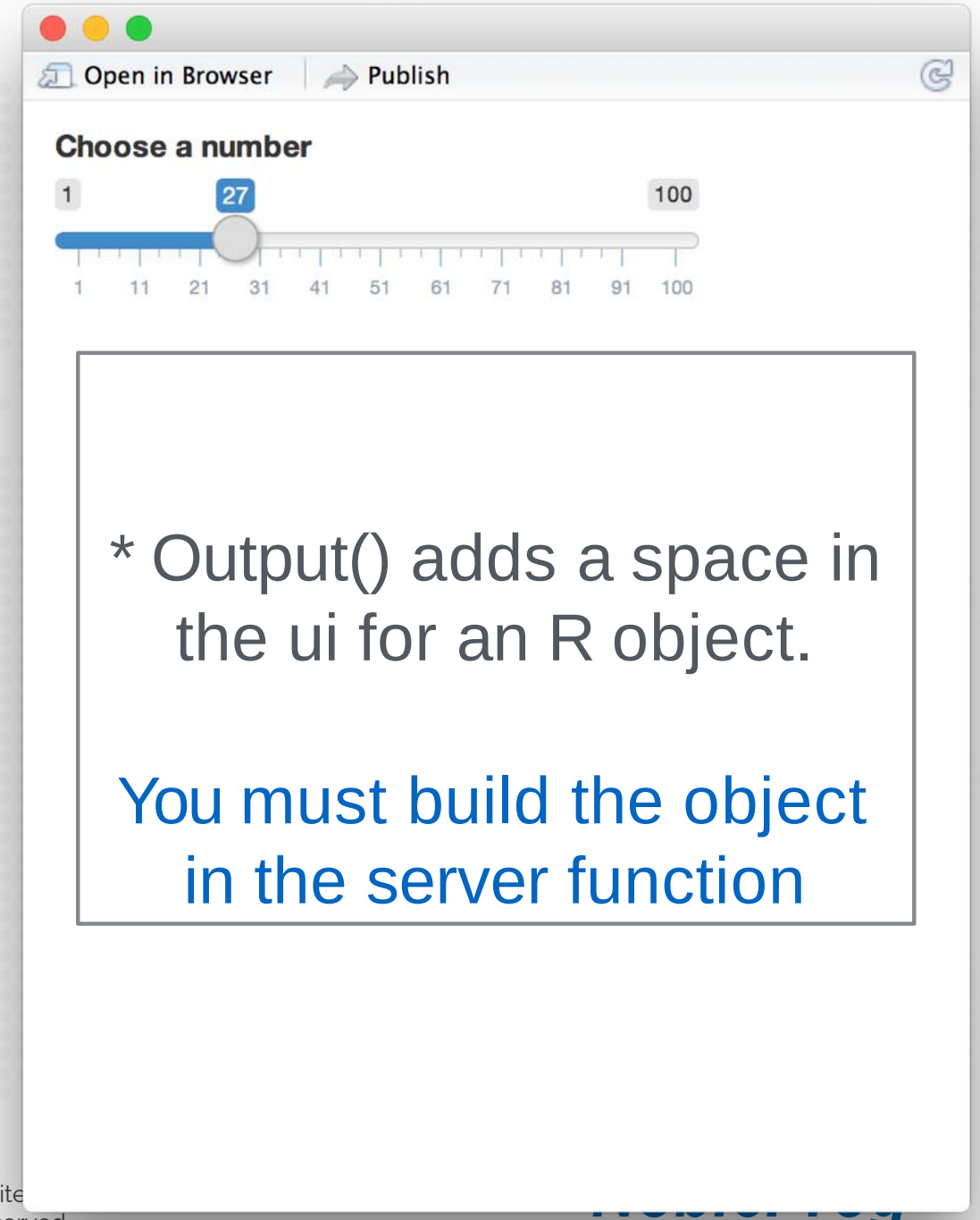

NobleProg

```
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)
```



```
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

```
sl 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>
```


NobleProg

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>
```


NobleProg

Shiny HTML tag functions

Shiny provides R functions to recreate HTML tags.

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

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>
```


NobleProg

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)

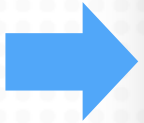
```
<a href="www.rstudio.com">RStudio</a>
```

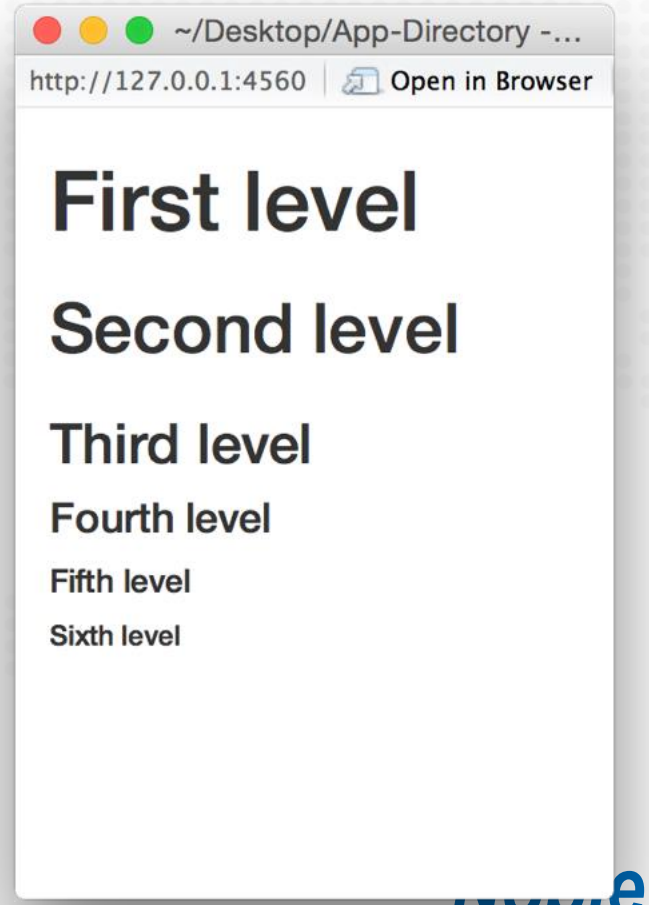

NobleProg

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")
)
```

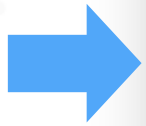


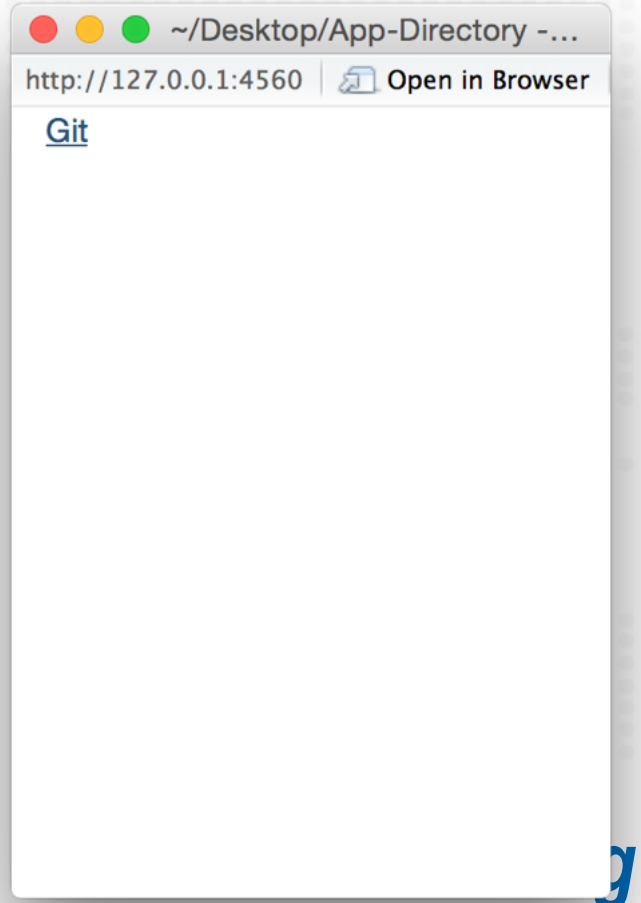


a()

hyperlink with the href argument

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

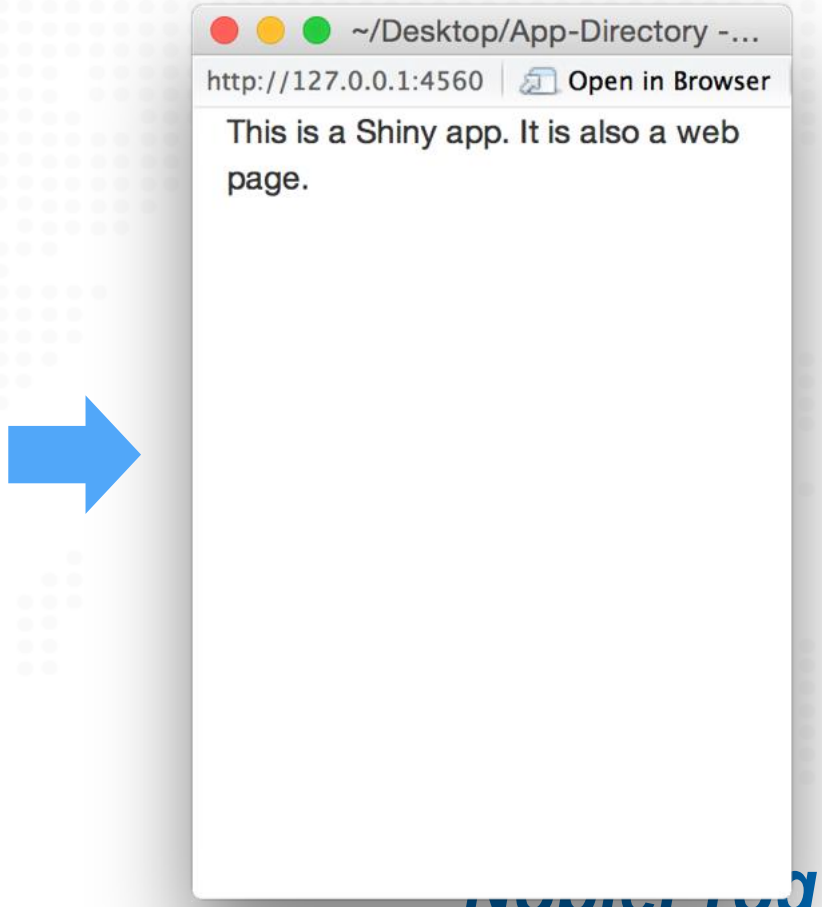




text

Character strings do not need a tag.

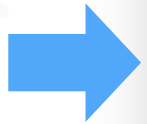
```
fluidPage(
 "This is a Shiny app.",
 "It is also a web page."
)
```

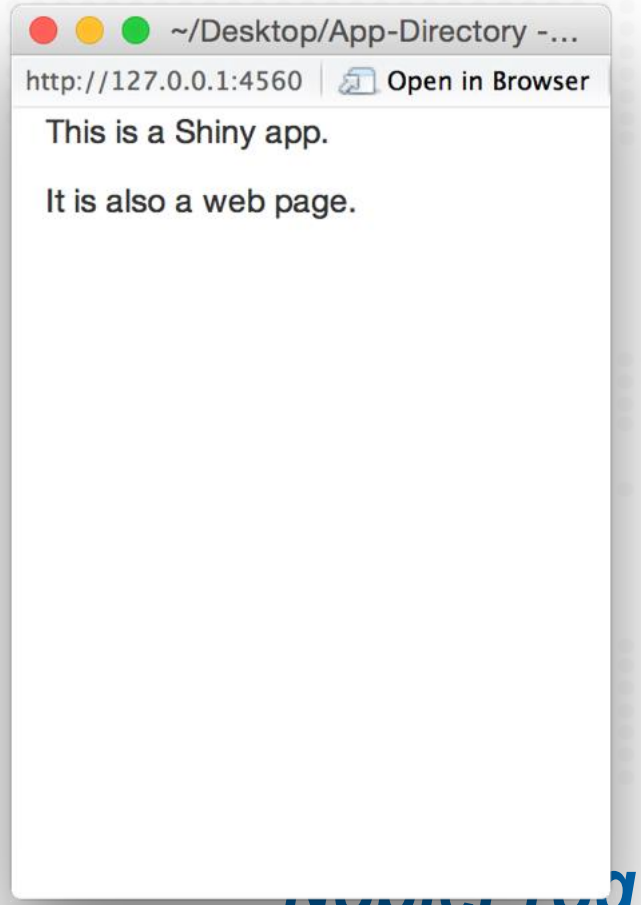


p()

A new paragraph

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

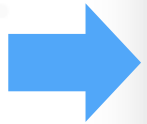


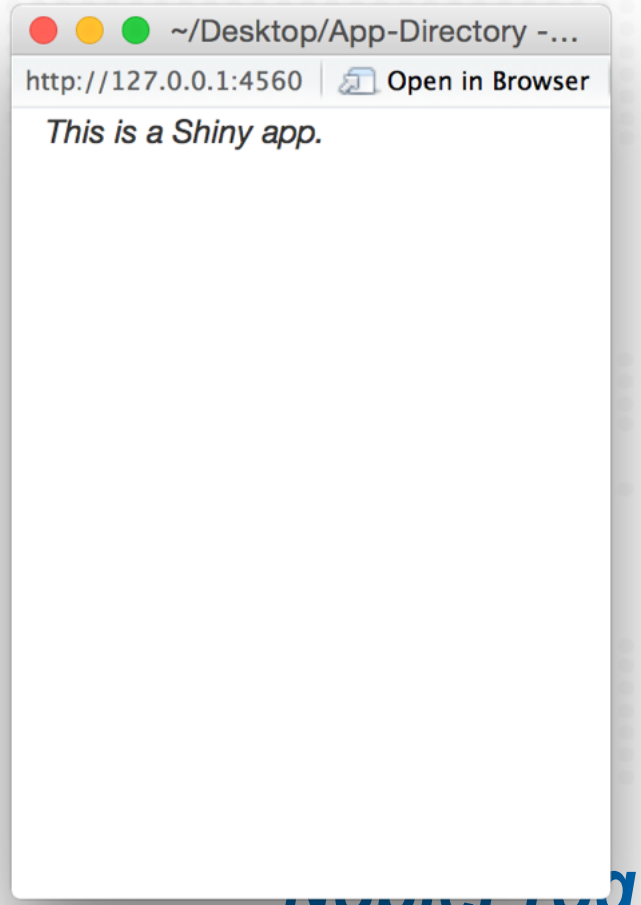


em()

Emphasized (*italic*) text

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

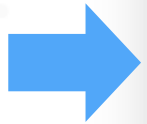


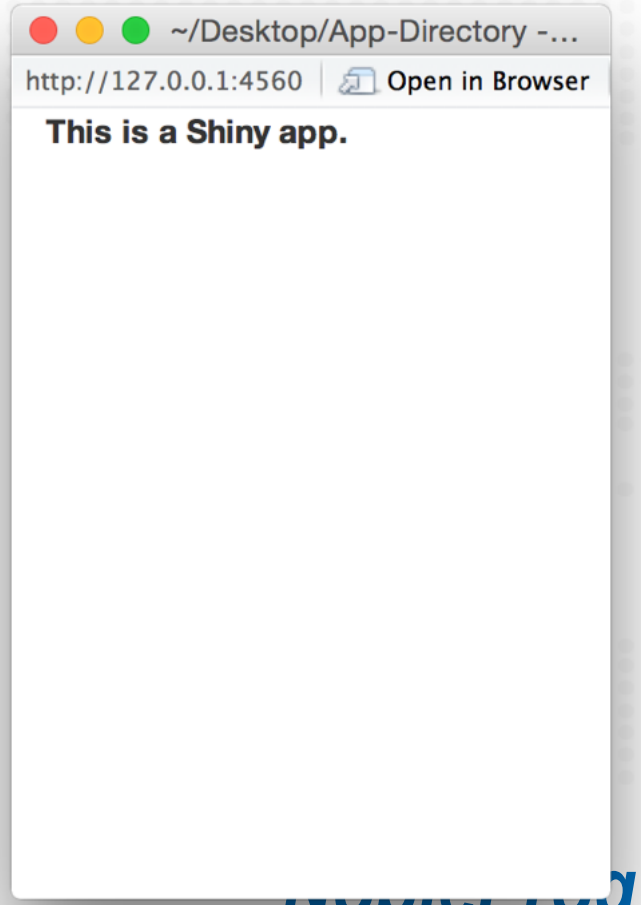


strong()

Strong (**bold**) text

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

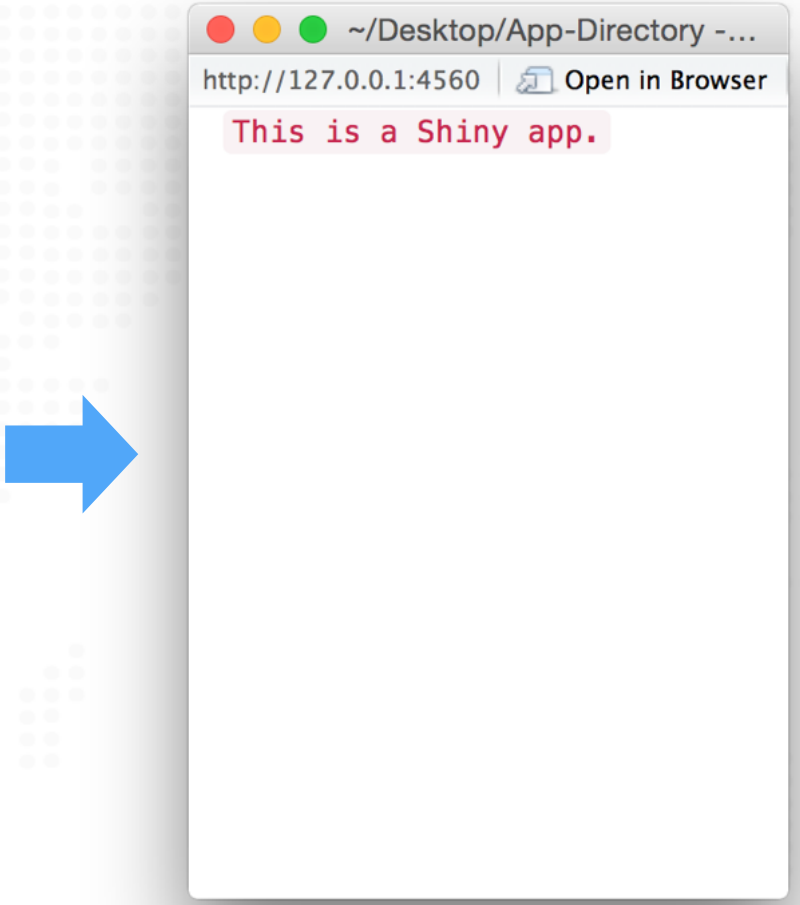




code()

Monospaced text (`code`)

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

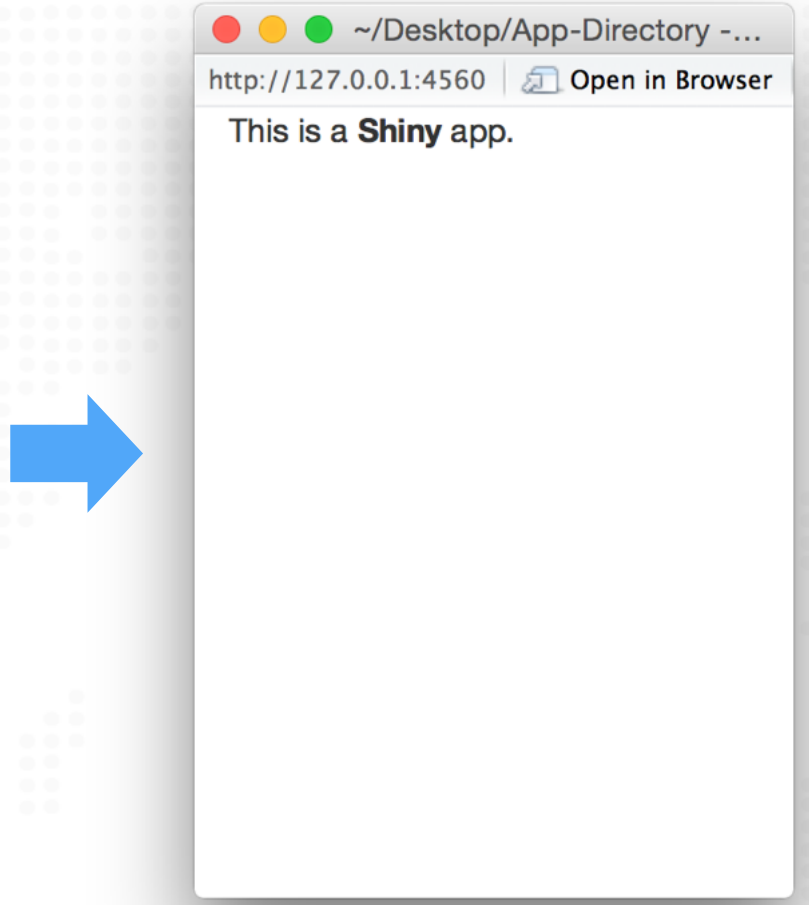


NobleProg

nesting

You can also nest functions inside of others

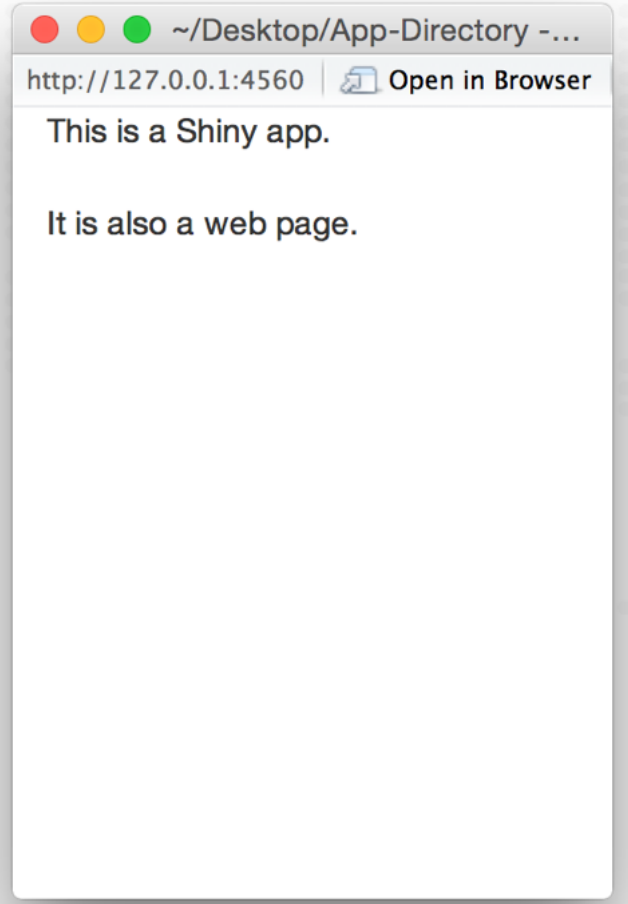
```
fluidPage(
  tags$p("This is a",
    tags$strong("Shiny"),
    "app.")
)
```



br()

A line break

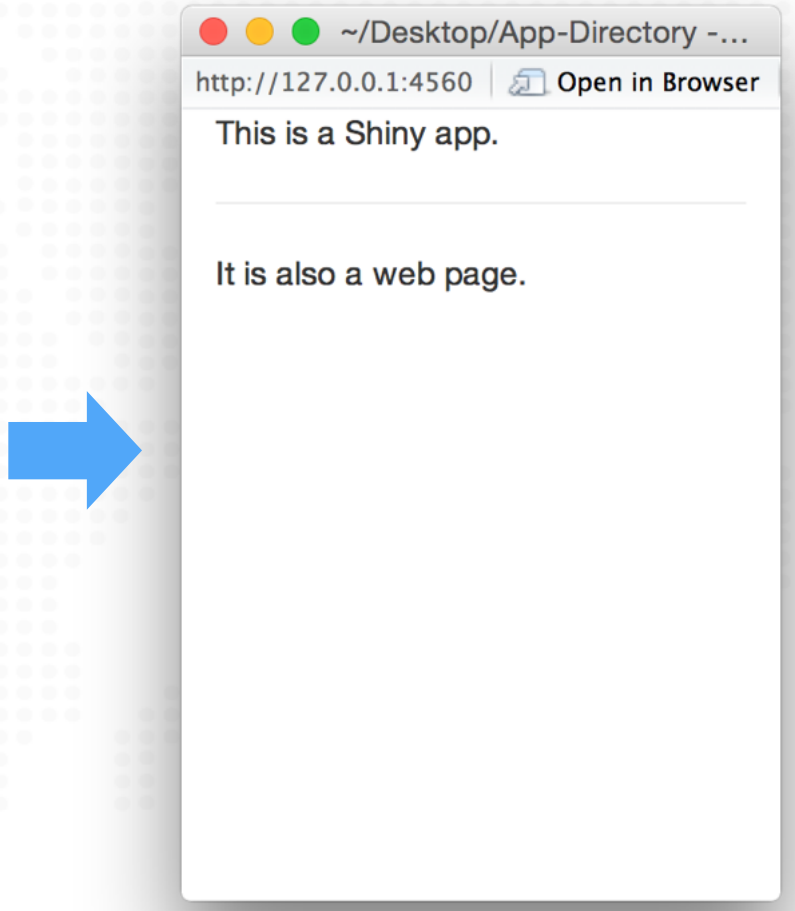
```
fluidPage(
  "This is a Shiny app.",
  tags$br(),
  "It is also a web page."
)
```



hr()

A horizontal rule

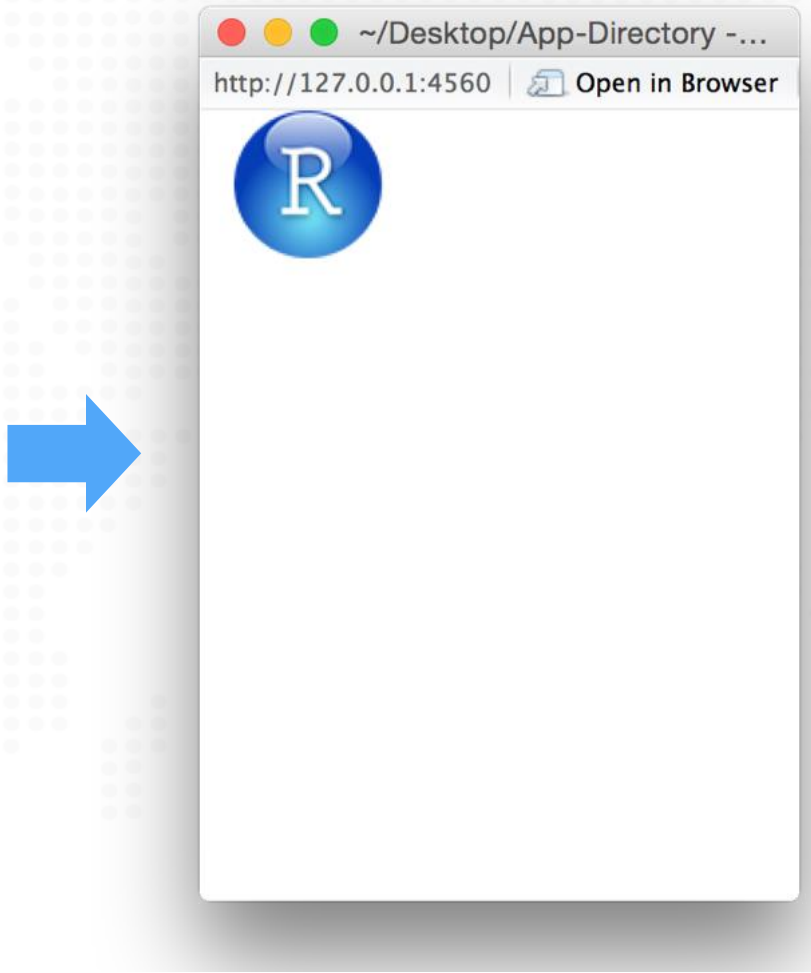
```
fluidPage(
  "This is a Shiny app.",
  tags$hr(),
  "It is also a web page."
)
```



img()

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

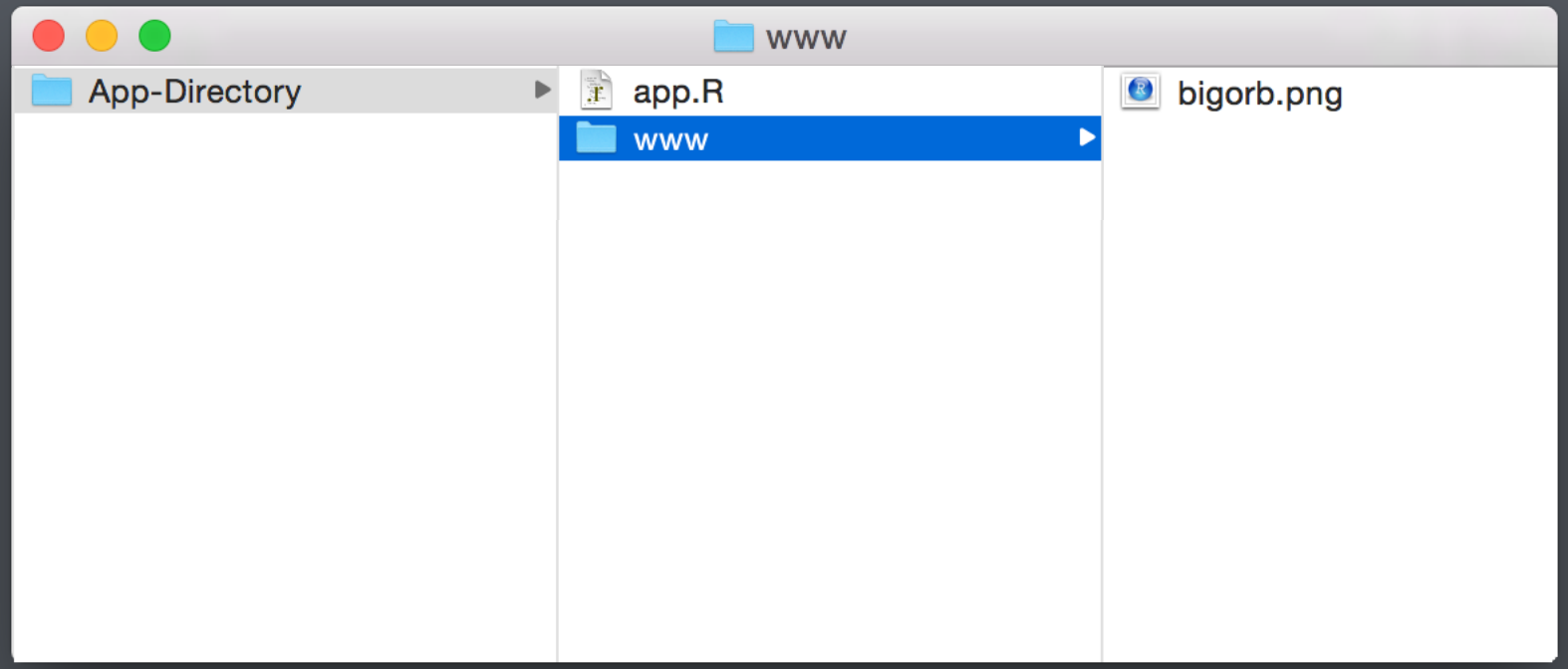
```
fluidPage(
  tags$img(height = 100,
           width = 100,
           src =
                  "http://www.rstudio.co
                  m/
                  images/RStudio.2x.png"
                  )

)
```



NobleProg

Adding Images

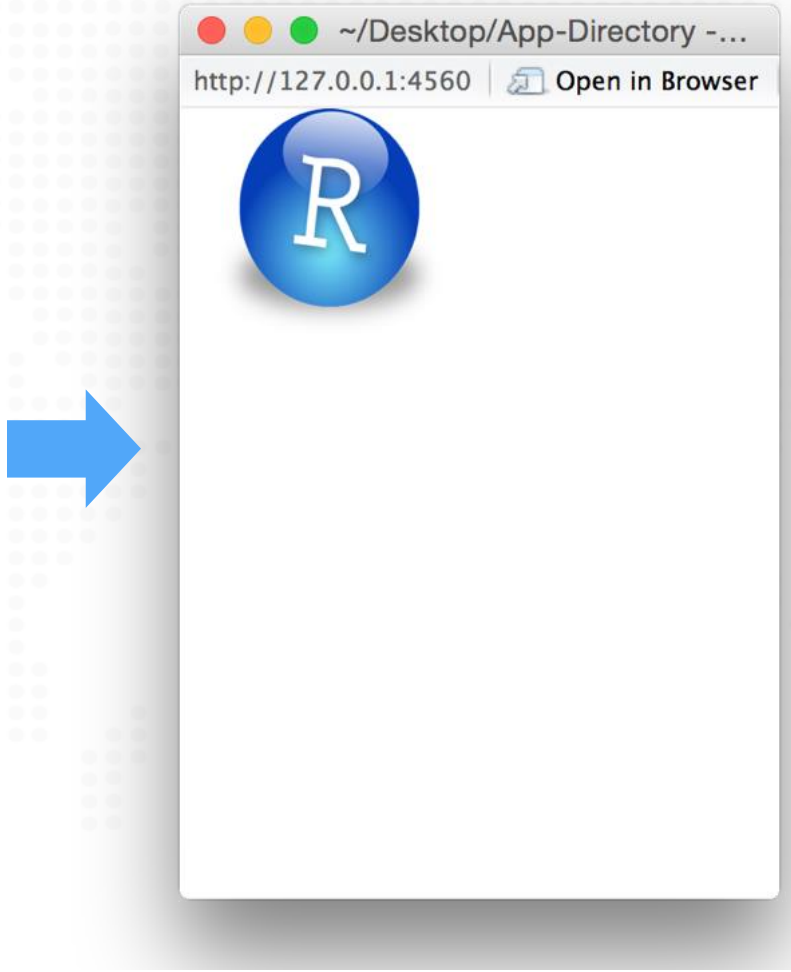
To add an image from a file, save the file in a subdirectory named **www**



img()

Add an img from a file in www

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



NobleProg

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

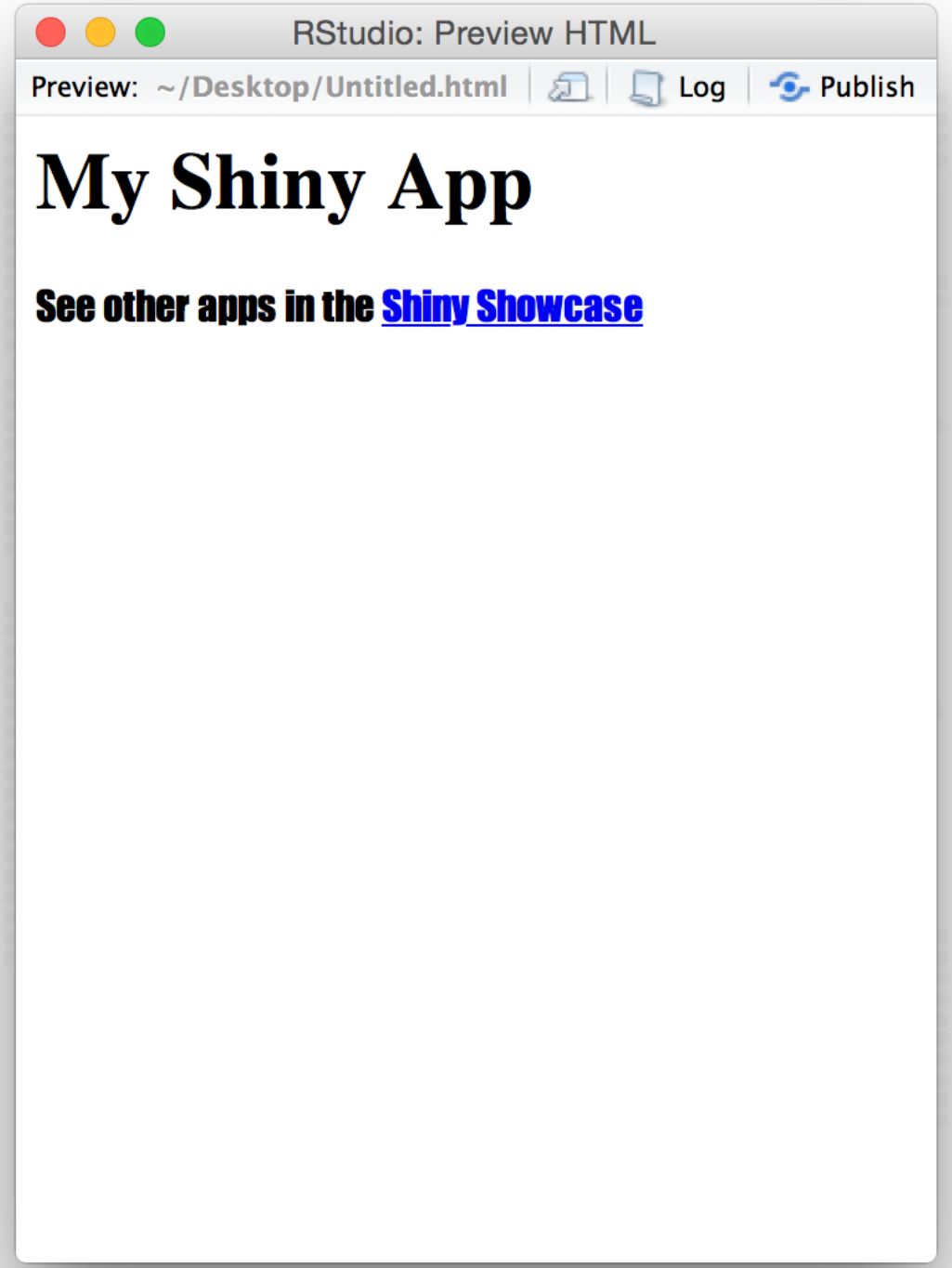
```
h1
```

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

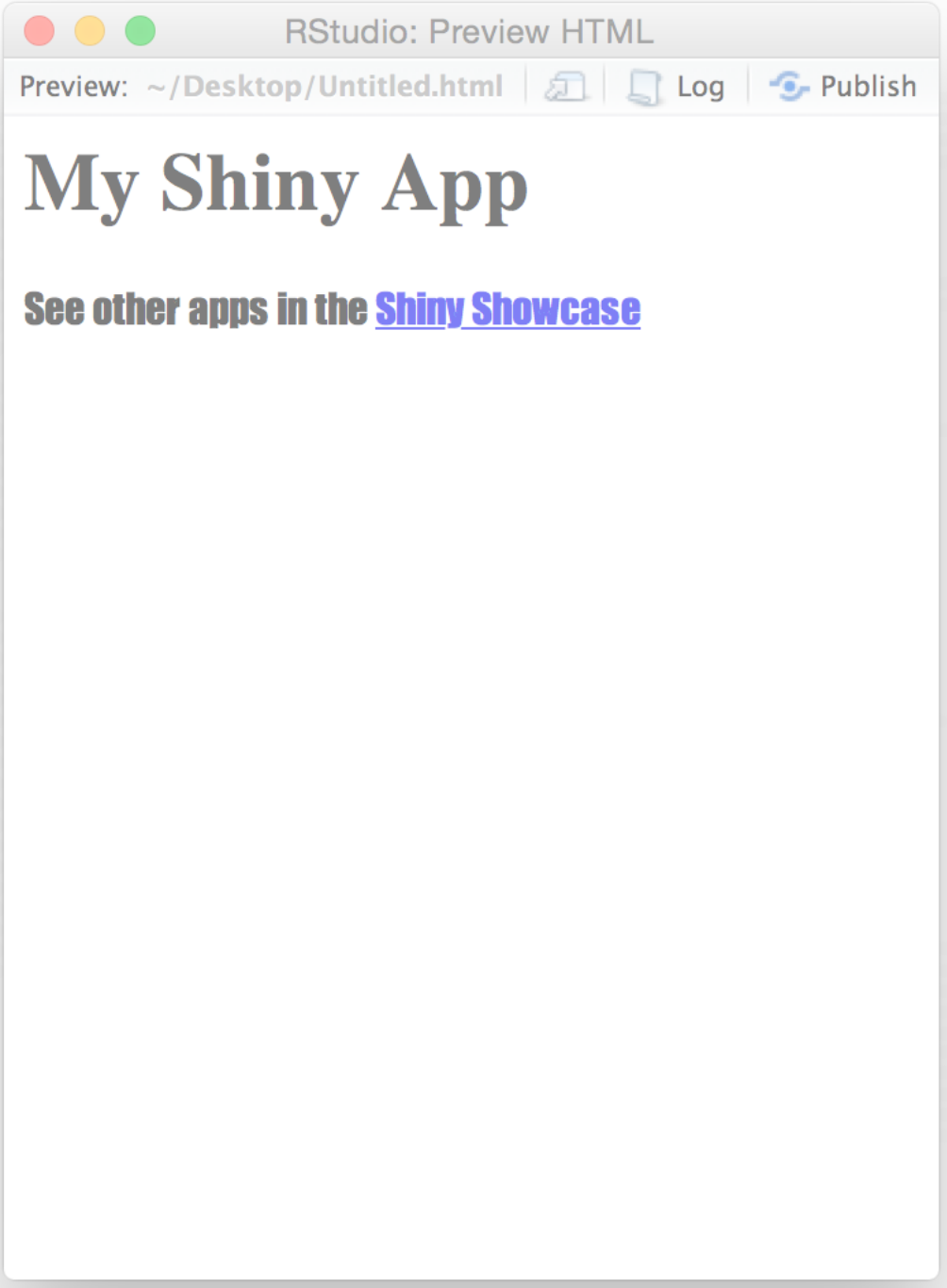

NobleProg

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

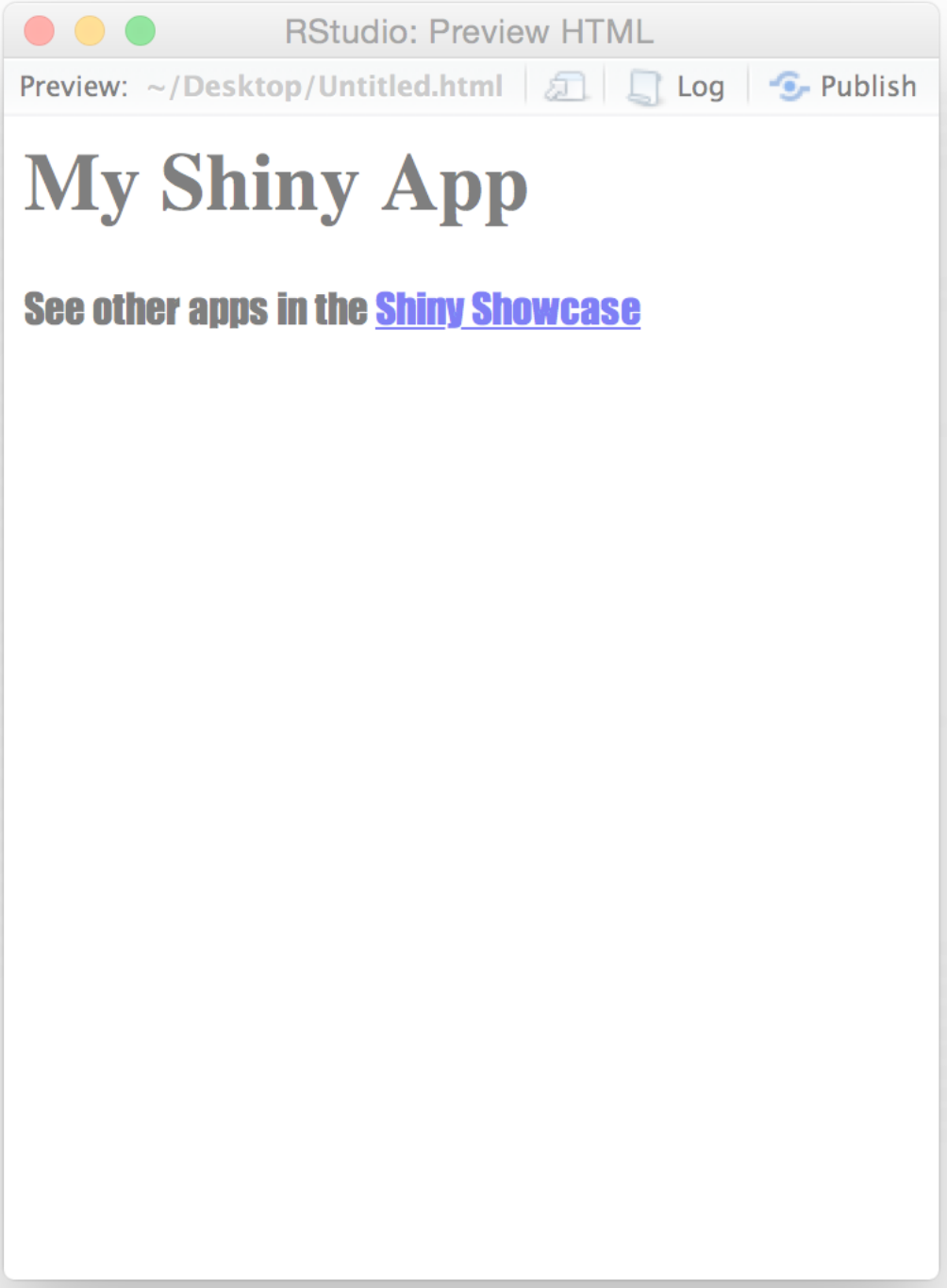
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



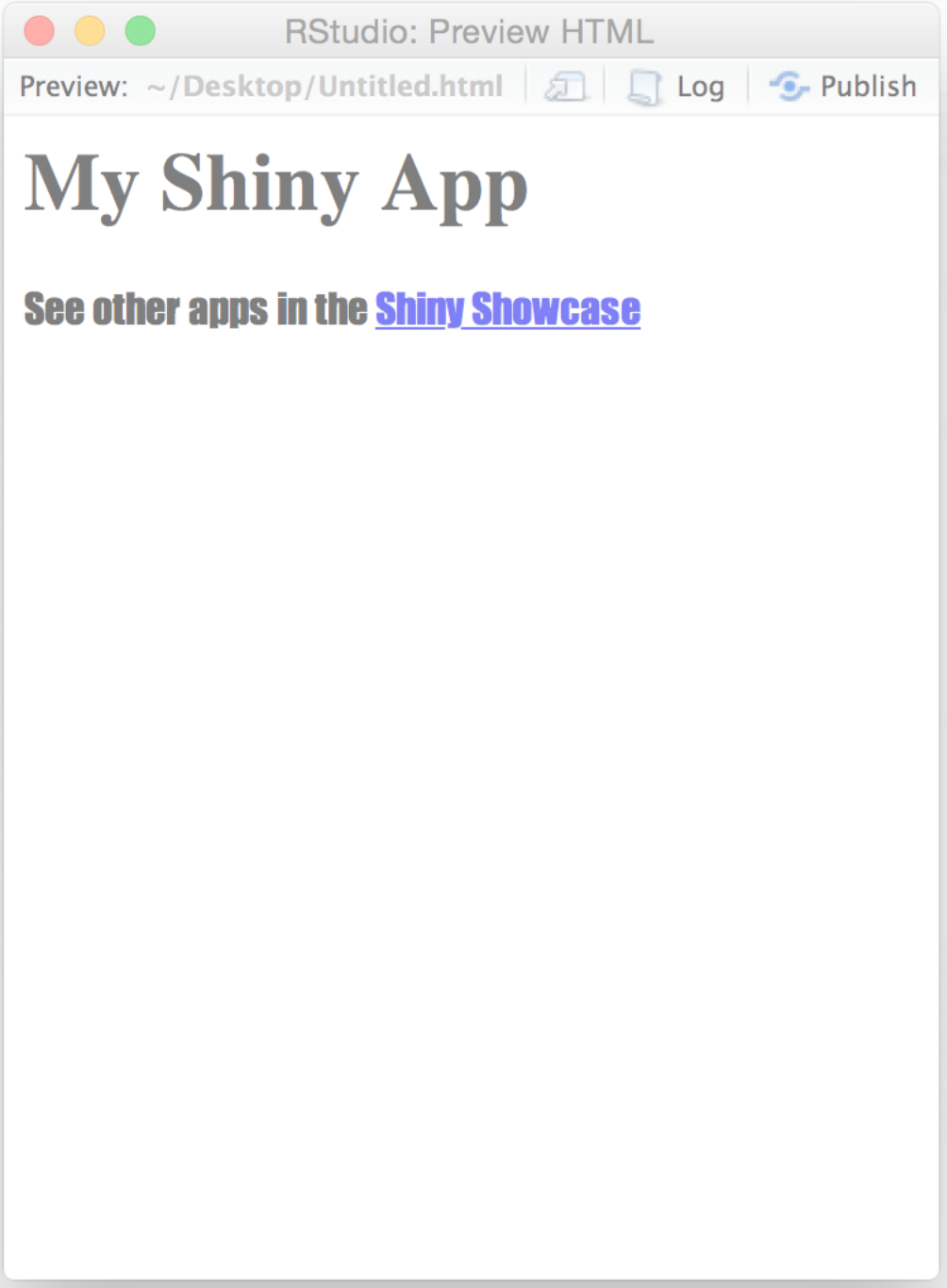
```
fluidPage(
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



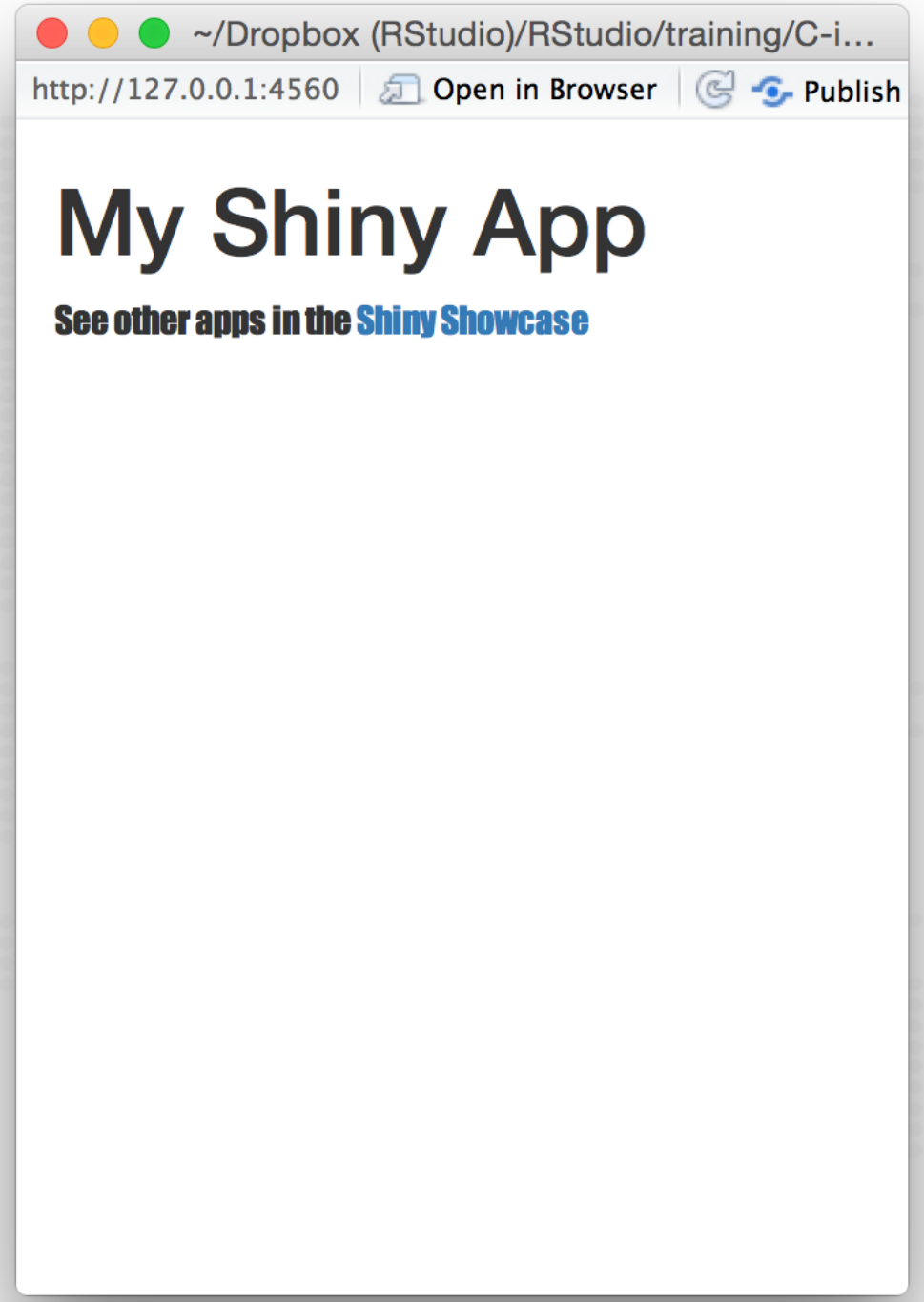
```
fluidPage(
  h1("My Shiny App"),
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



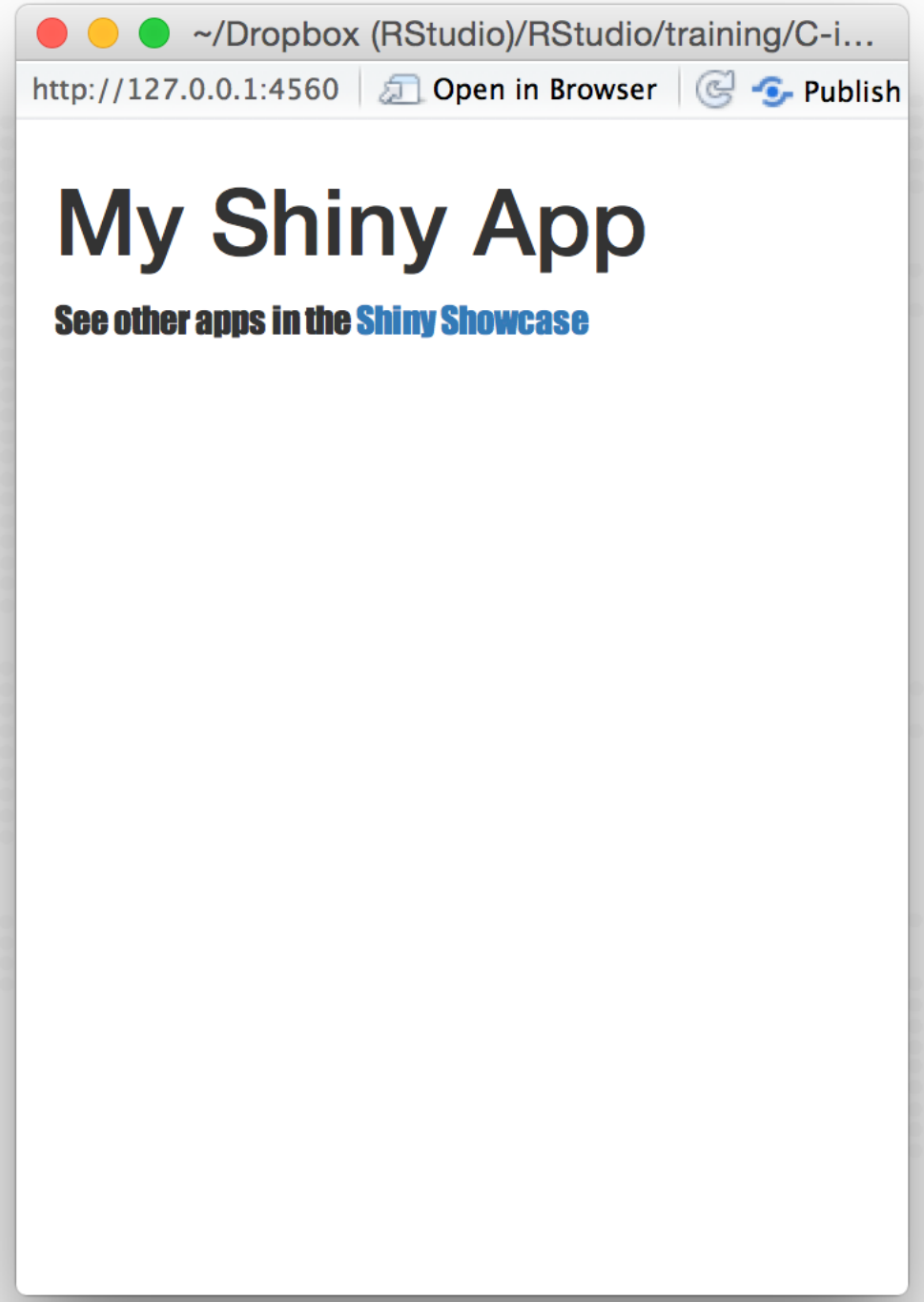
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  )
)
```



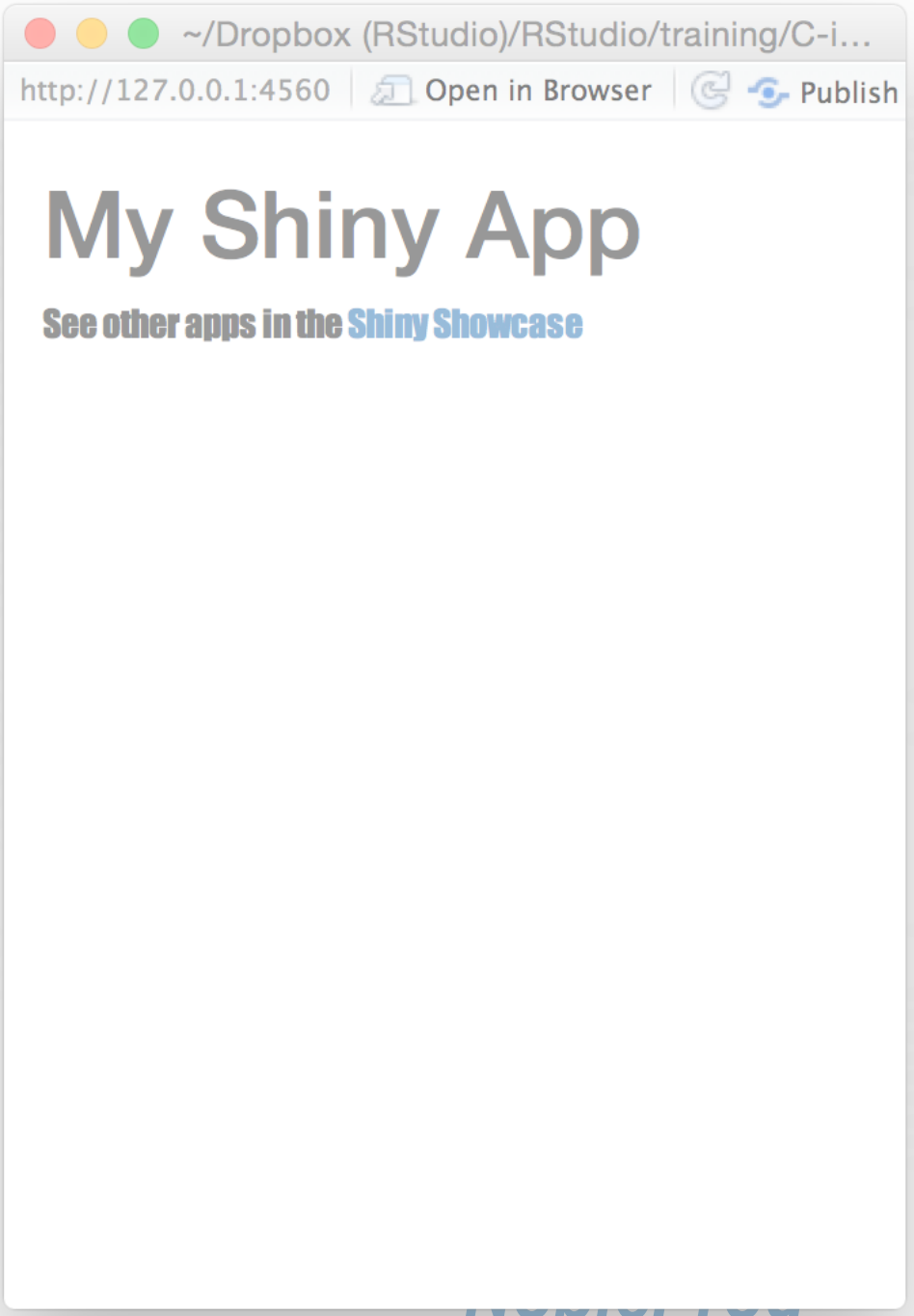
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



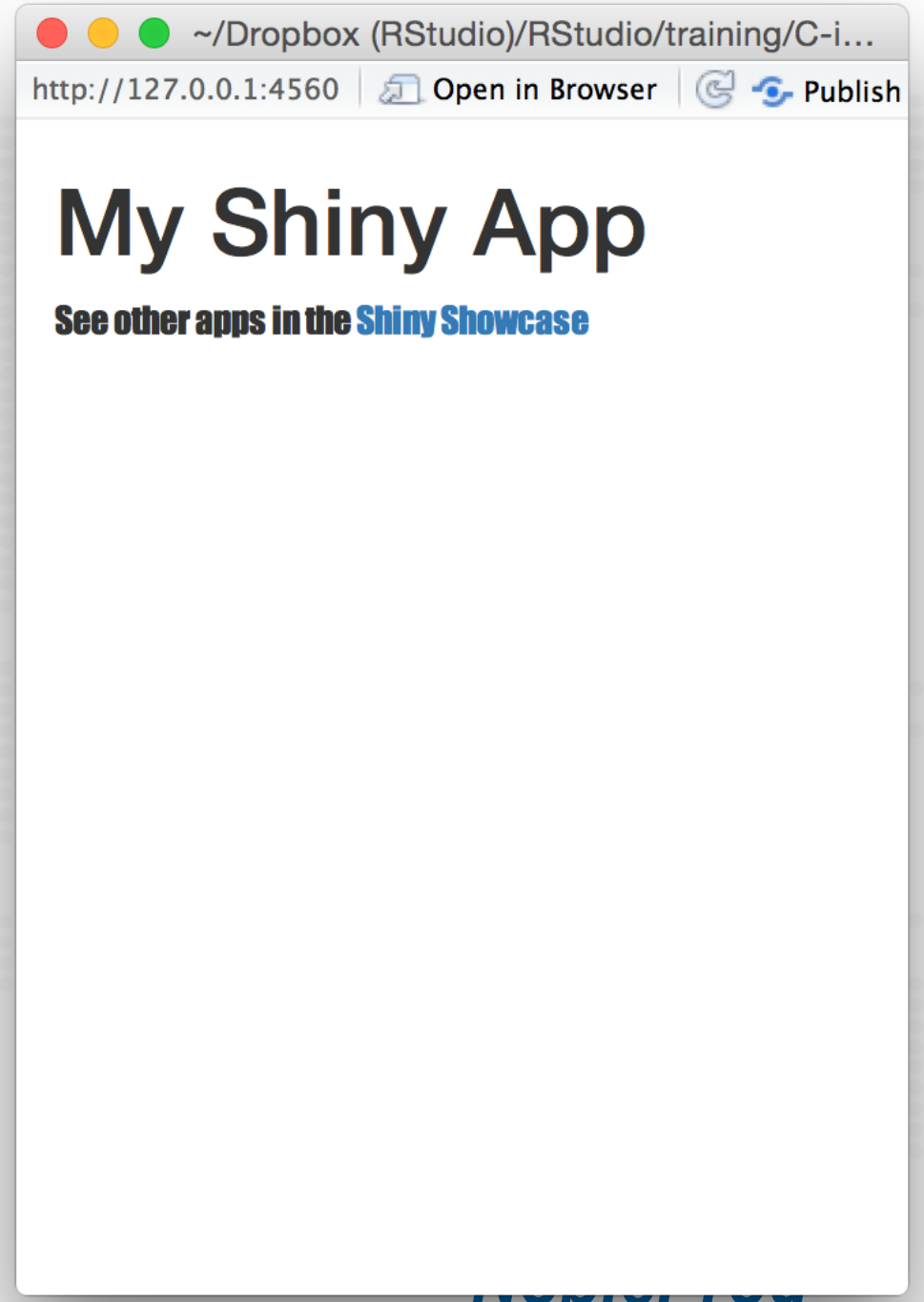
```
fluidPage(
  tags$h1("My Shiny App"),
  tags$p(style = "font-family:Impact",
    "See other apps in the",
    tags$a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



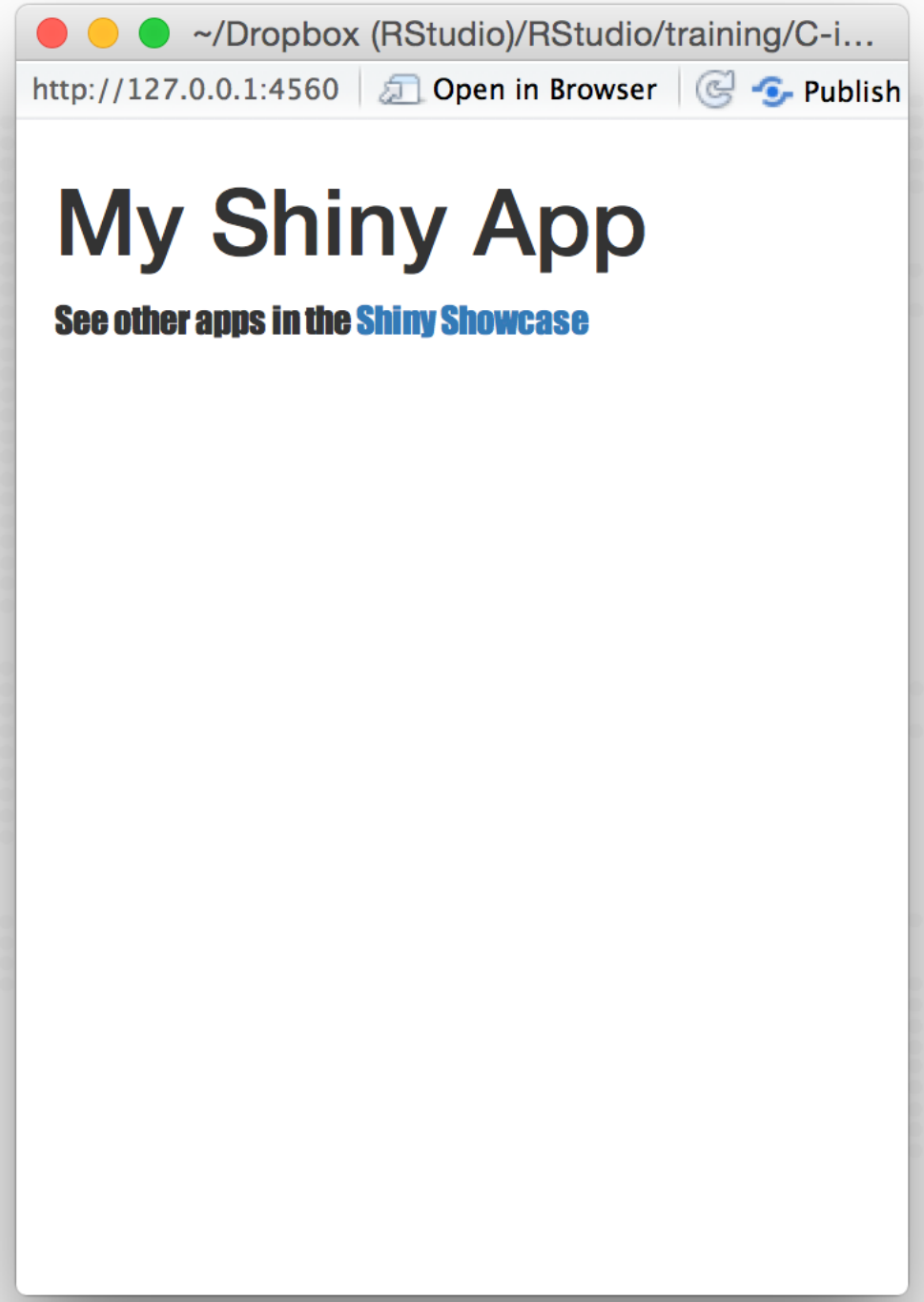
```
fluidPage(

    <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>

)
```



```
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>'

)
```



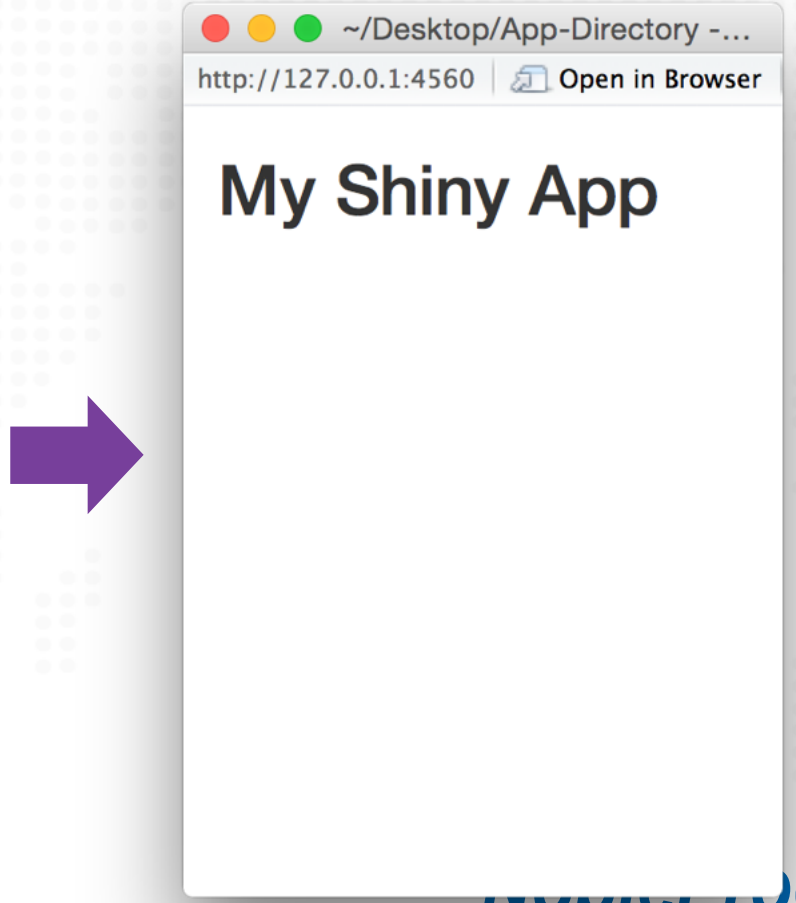
```
fluidPage(
  HTML(
    '<div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>'
  )
)
```



Raw HTML

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

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



Recap: Reactive values

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()**

NobleProg

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 />
```

Layout functions

Add HTML that divides the UI into a grid

```
fluidRow()
```

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

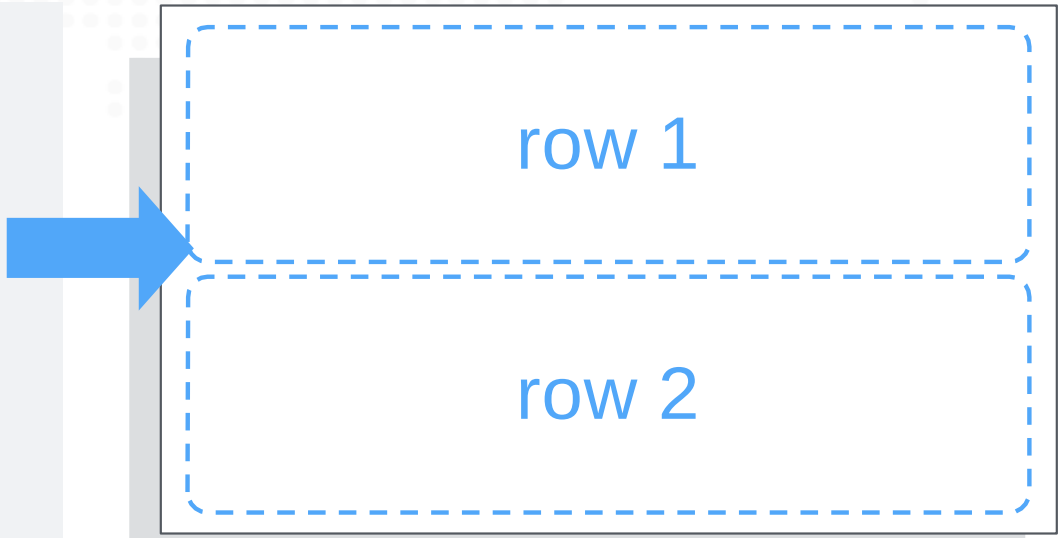
```
column(width = 2)
```

```
<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()
```



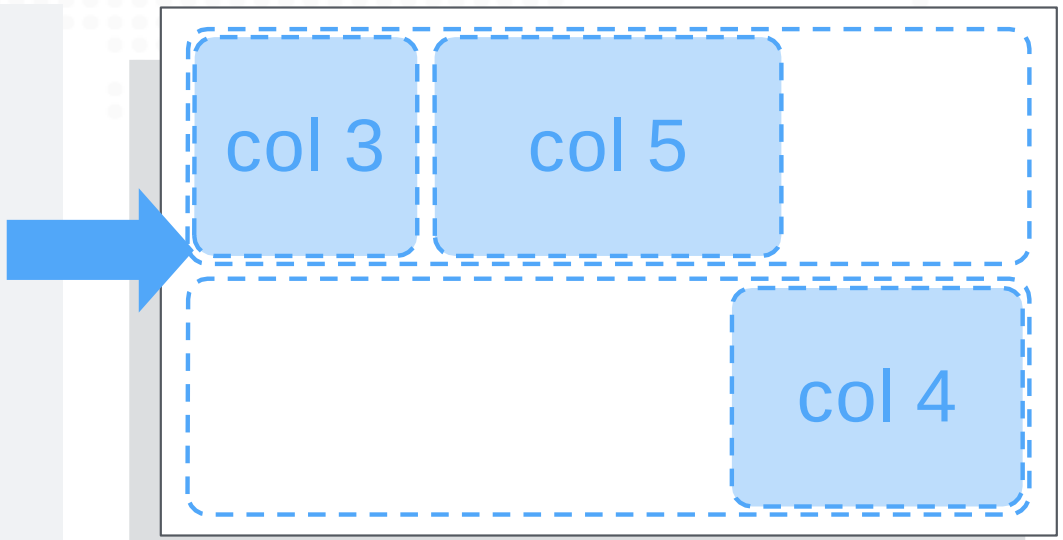
NobleProg

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>
```


NobleProg

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")
```

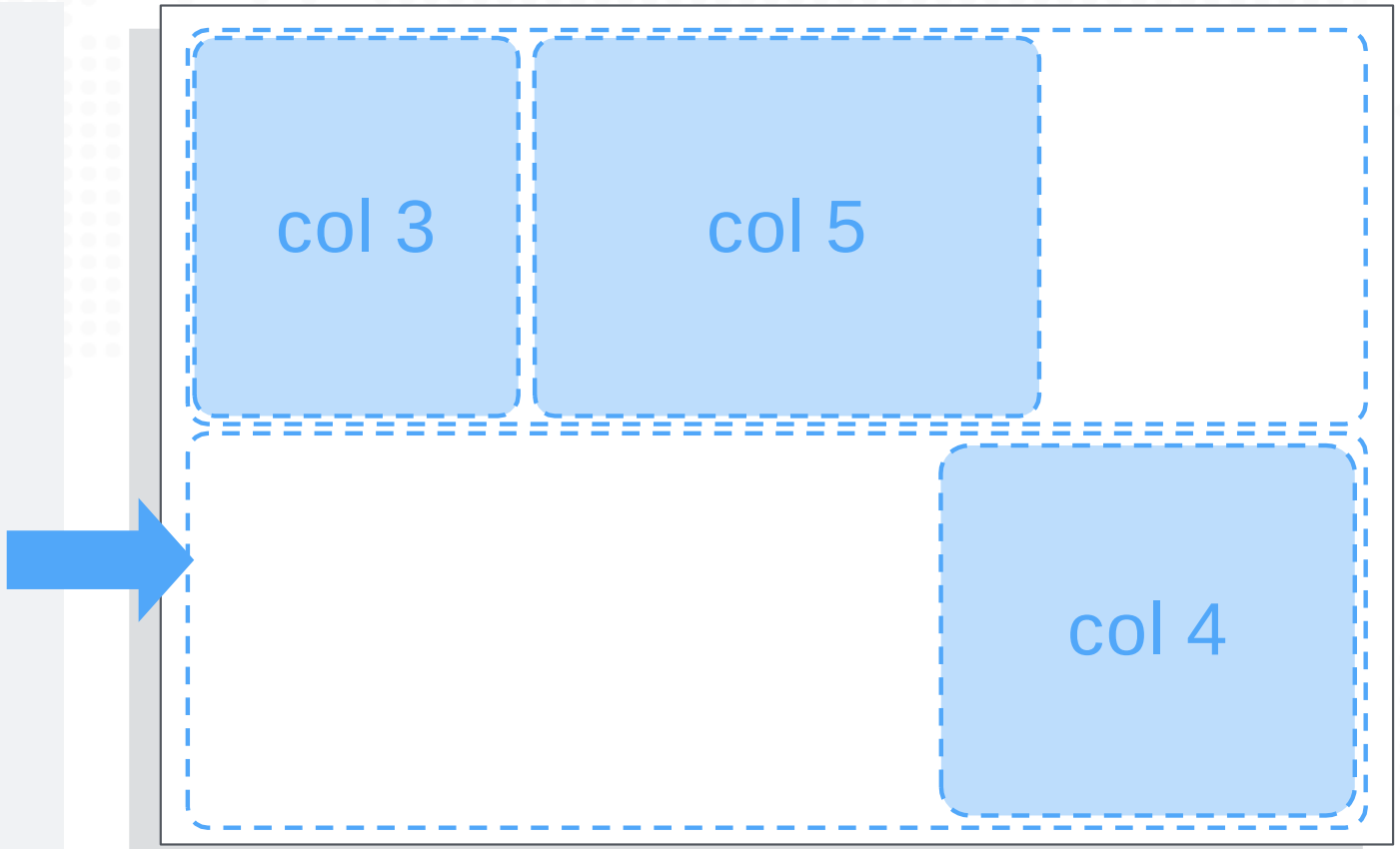
```
<div class="row">In the row</div>
```

```
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

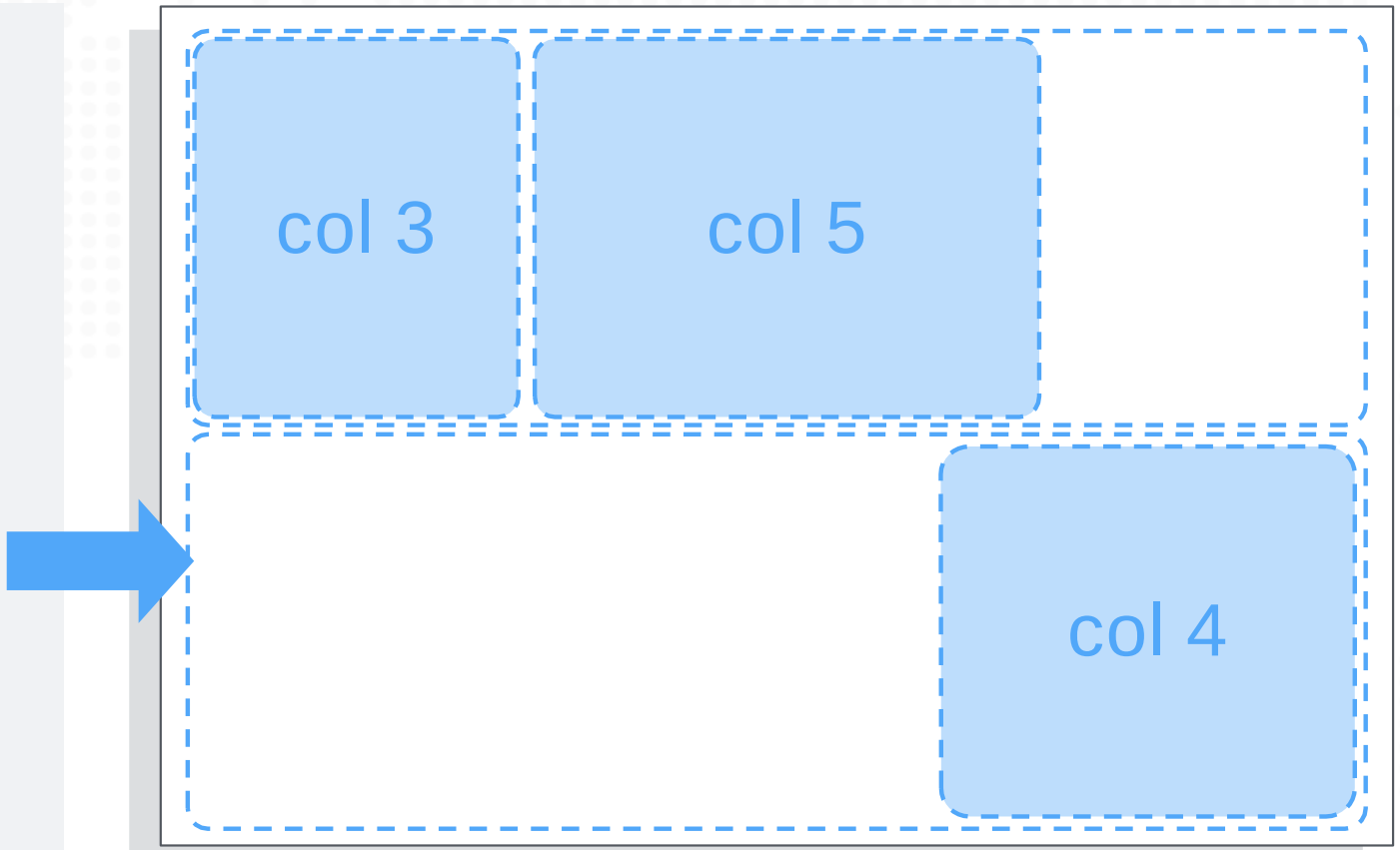
```
fluidPage(
  fluidRow(
   column(3),
   column(5)
  ),
  fluidRow(
    column(4, offset = 8,)
  )
)
```



NobleProg

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

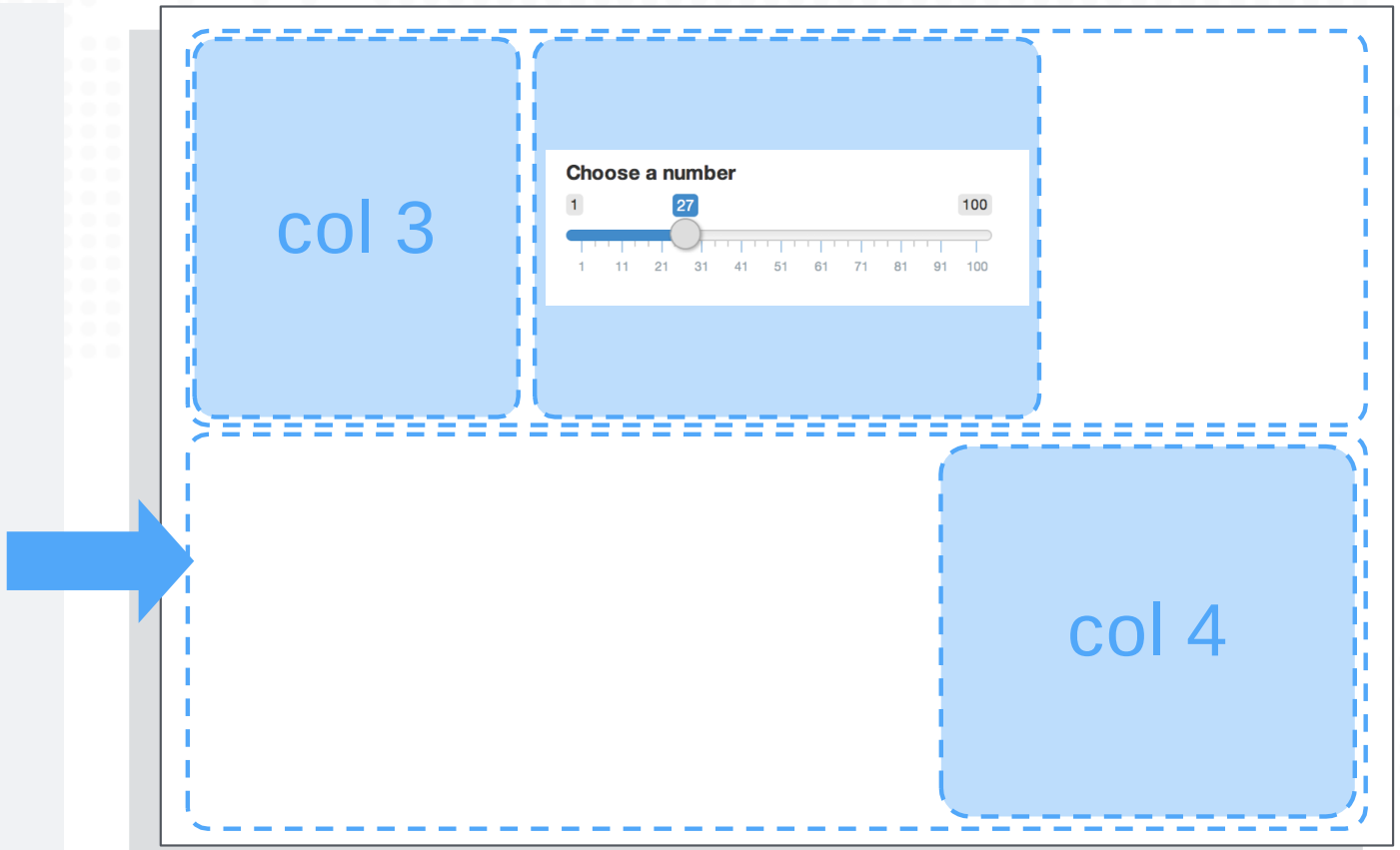
```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8)
  )
)
```



NobleProg

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

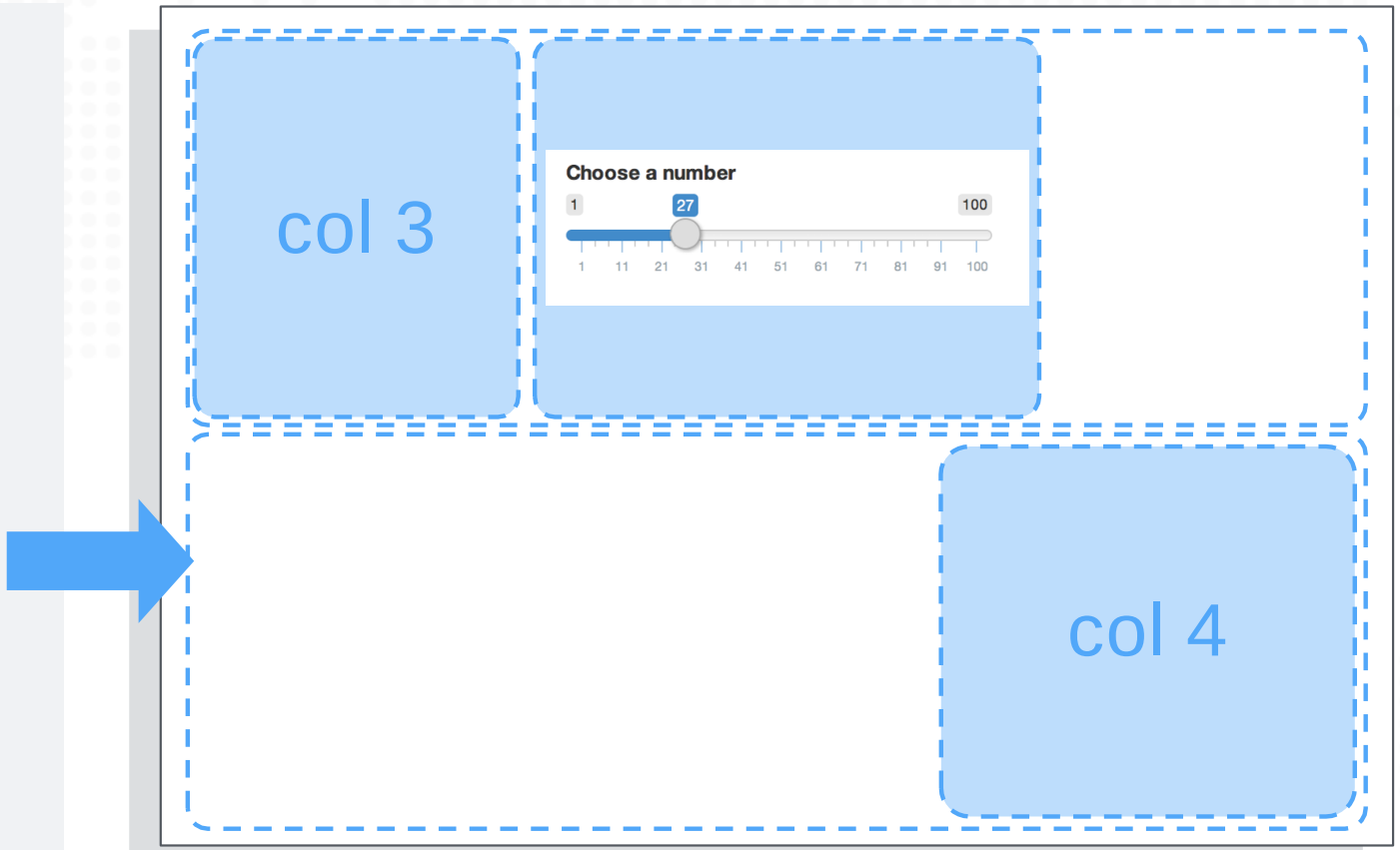
```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8)
  )
)
```



NobleProg

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")
    )
  )
)
```



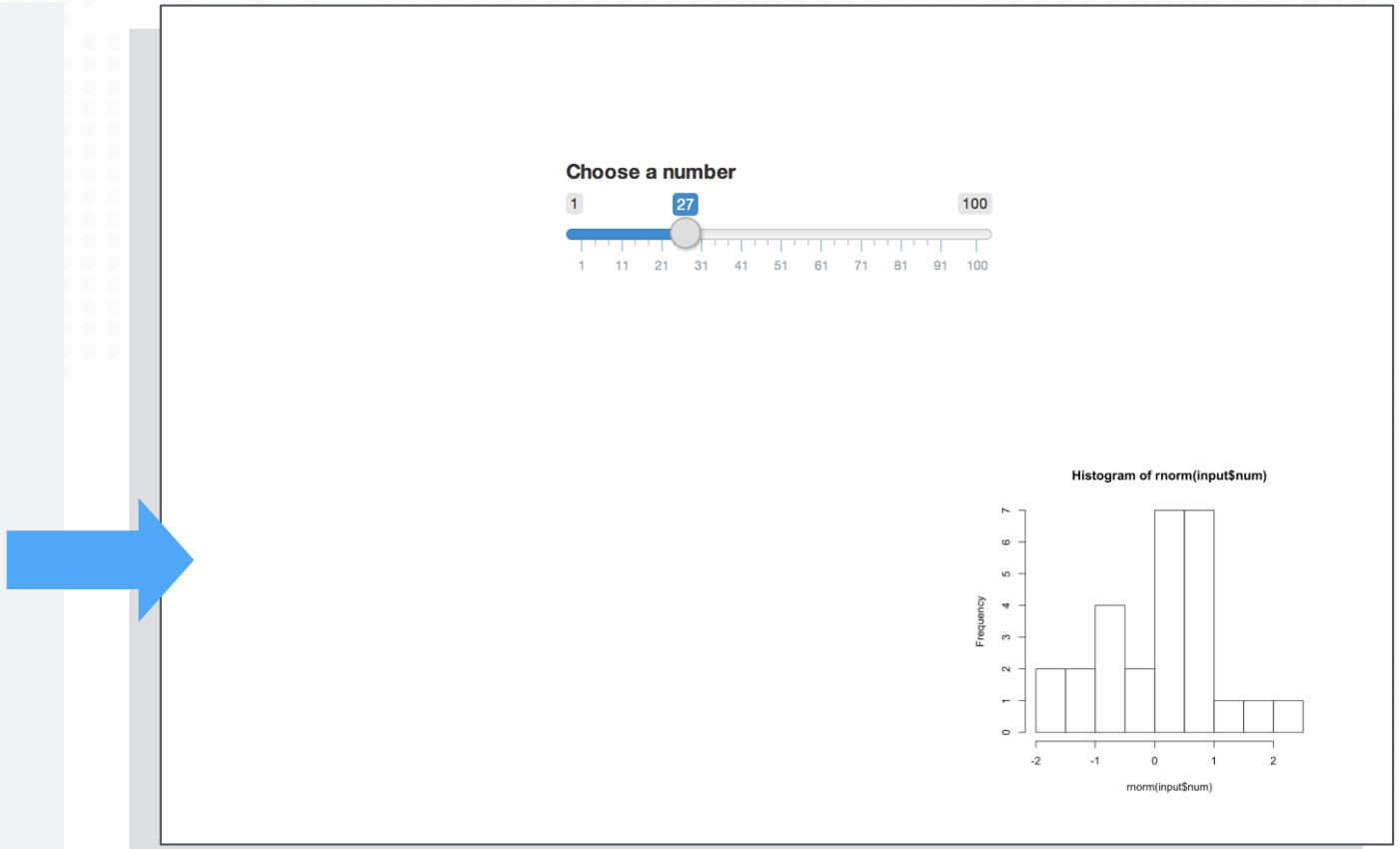
NobleProg

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

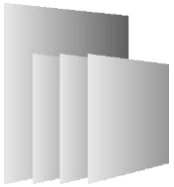
```
fluidPage(
  fluidRow(
   column(3),
   column(5, sliderInput(…))
  ),
  fluidRow(
    column(4, offset = 8,
      plotOutput("hist")
    )
  )
)
```

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")
    )
  )
)
```

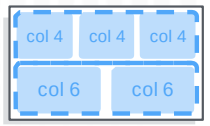


Recap: Layout functions

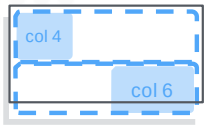


Position elements in a grid, or stack them in layers.

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



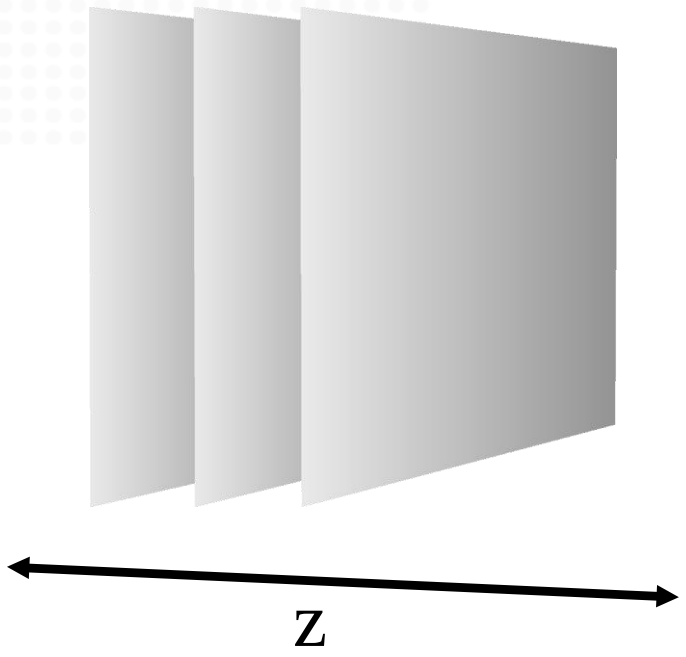
Use **column()** to arrange elements in columns



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

NobleProg

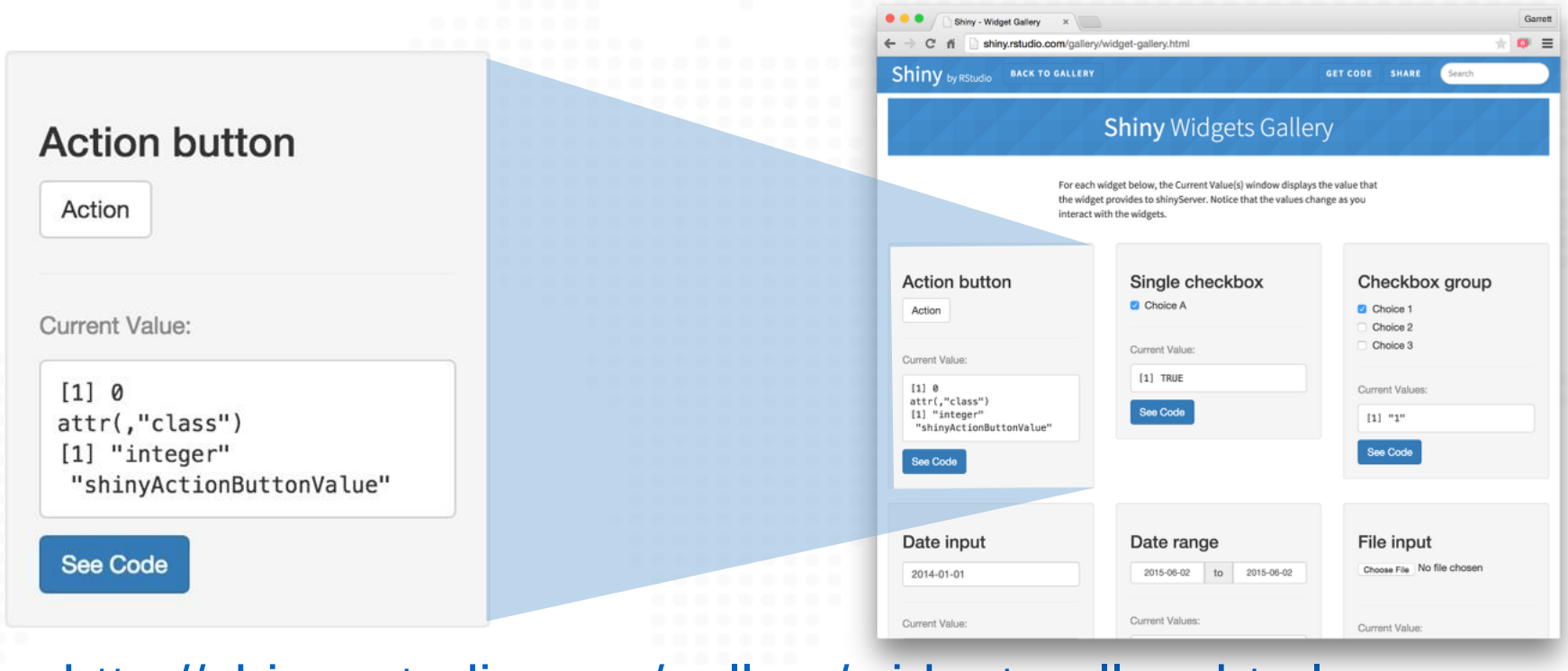
Use layout functions to position elements within your app



NobleProg

Panels

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

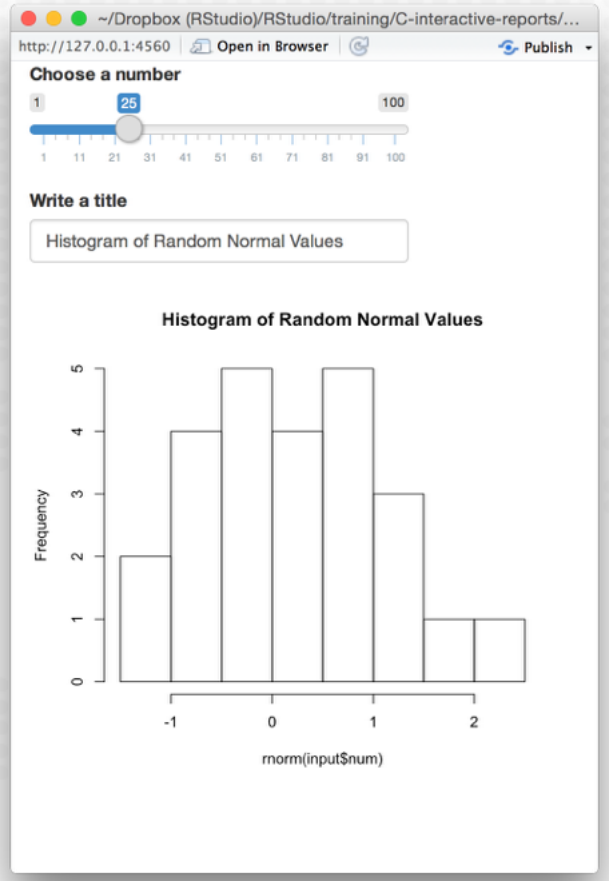


http://shiny.rstudio.com/gallery/widget-gallery.html

wellPanel()

Groups elements into a grey "well"

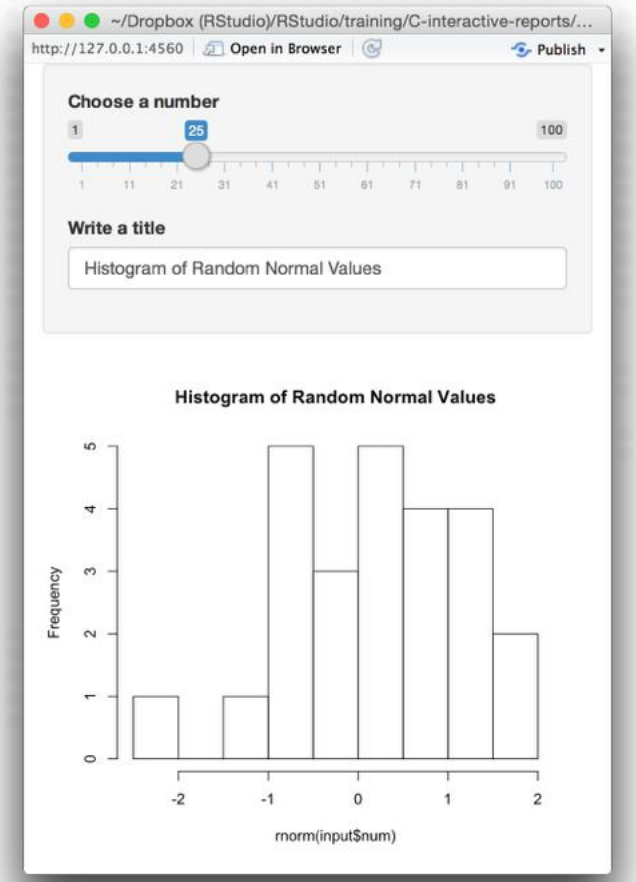
```
# 04-well.R

ui <- fluidPage(

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

 plotOutput("hist")
)
```



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")
)
```



NobleProg

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.

NobleProg

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

NobleProg

Combine tabPanel()'s with one of:

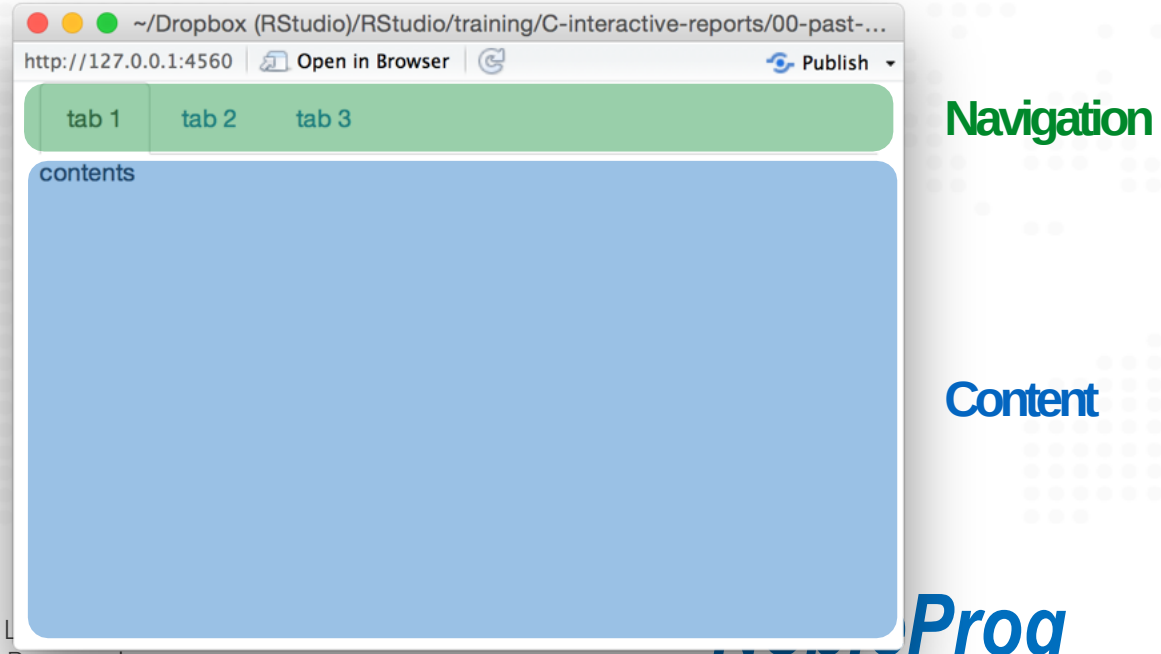
- tabsetPanel()
- navlistPanel()
- navbarPage()

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")
  )
)
```

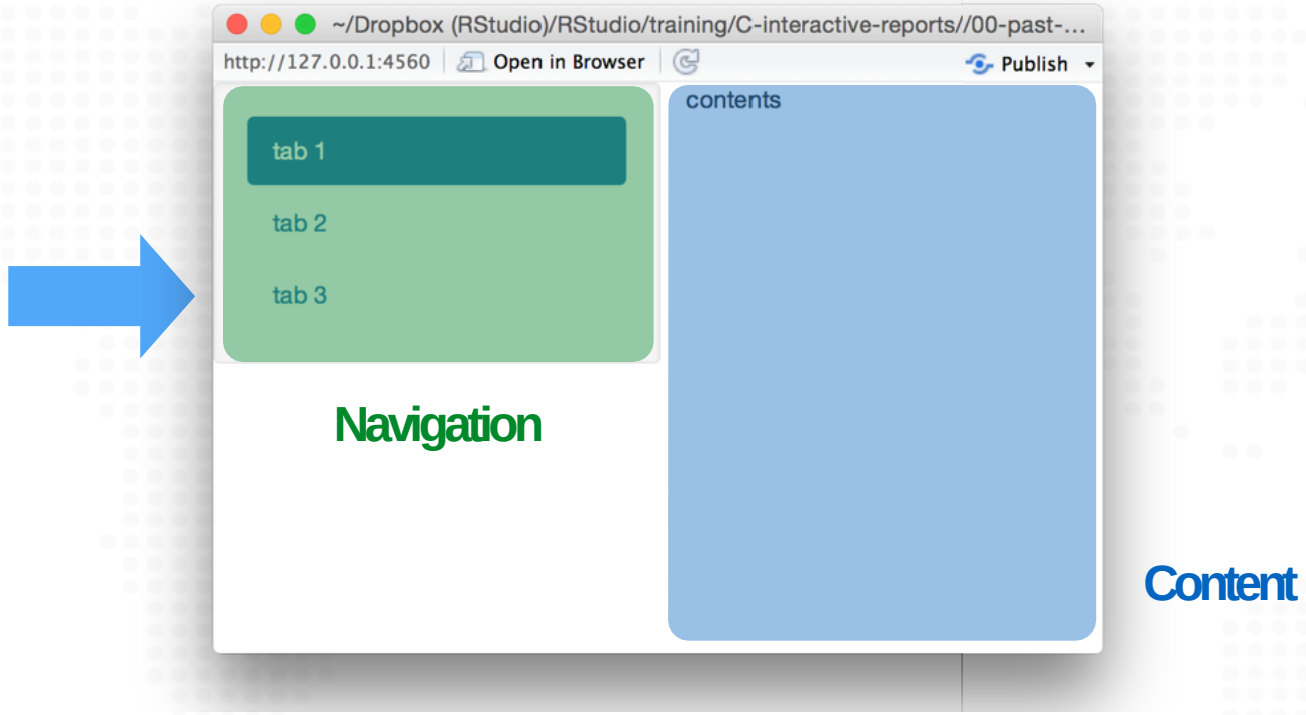


navlistPanel()

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

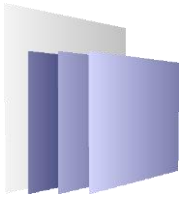
Use *links* to navigate between tabs.

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

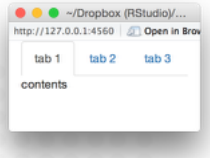


Recap: Panels

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



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



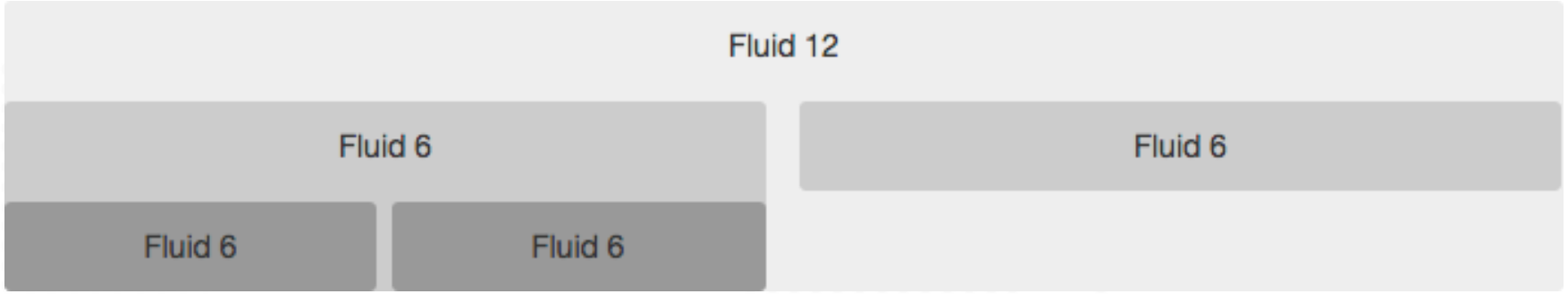
Use **tabsetPanel()** to arrange tab panels into a stack with tab navigation

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

NobleProg

The Shiny Layout Guide

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



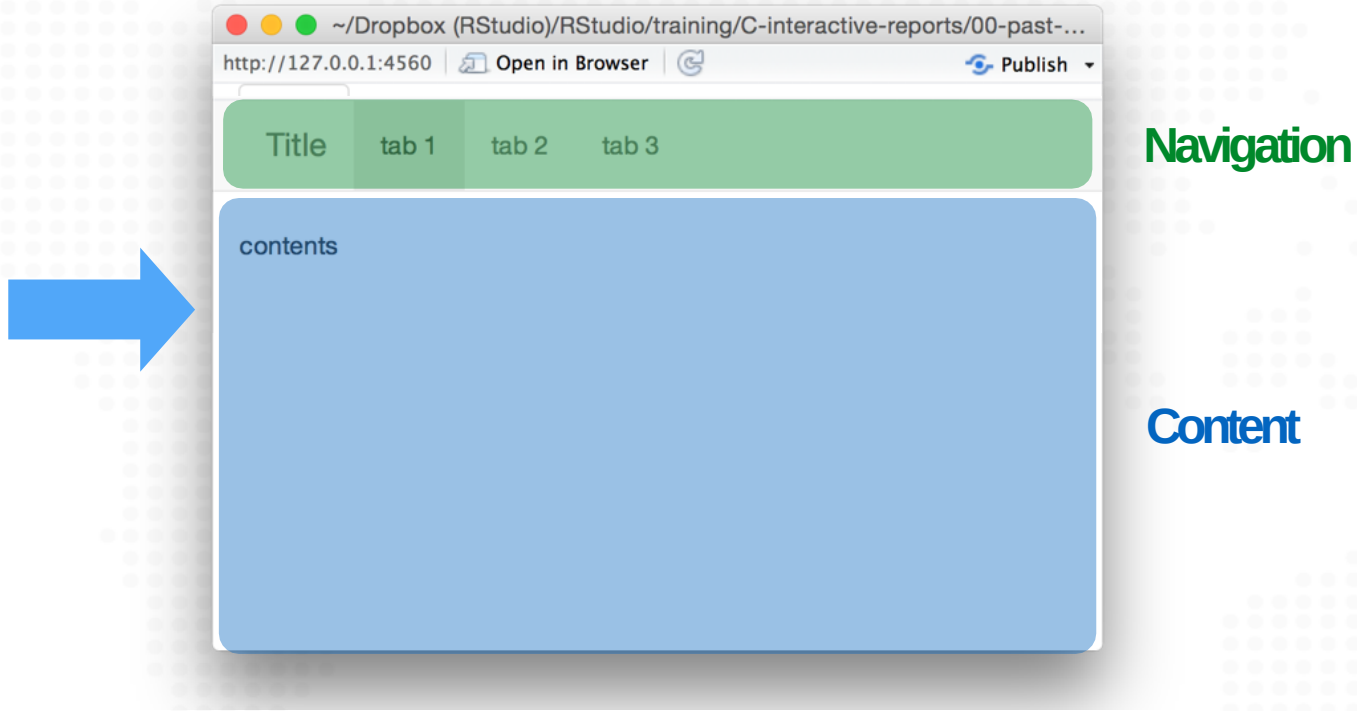
You can build sophisticated, customized layouts with Shiny's grid system.

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")
)
```

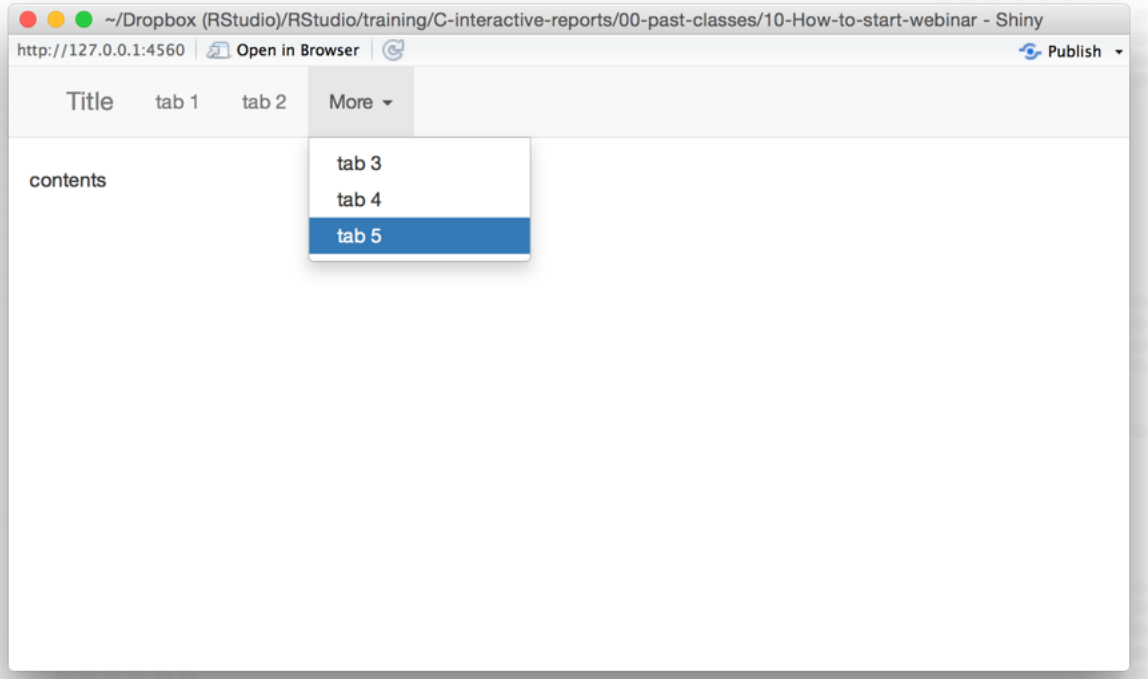


NobleProg

navbarMenu()

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

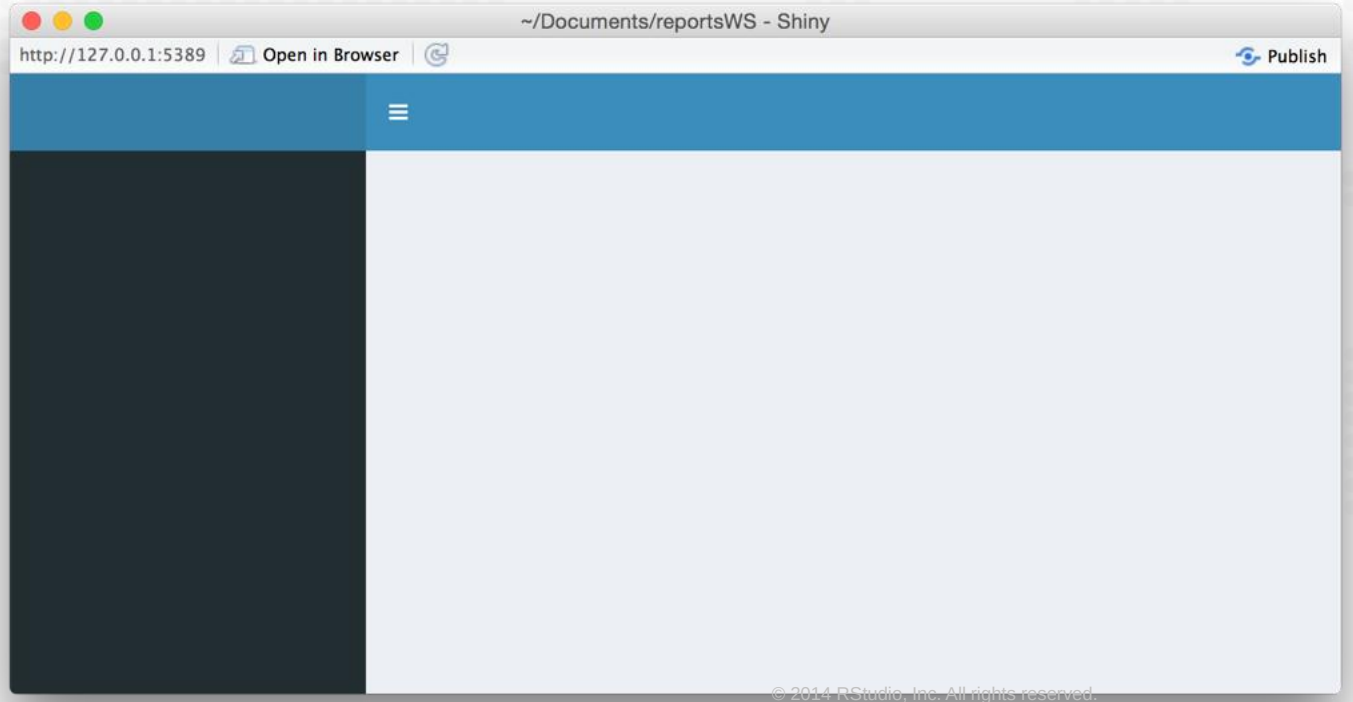
```
navbarPage(title = "Title",
    tabPanel("tab 1", "contents"),
    tabPanel("tab 2", "contents"),
    navbarMenu(title = "More",
      tabPanel("tab 3", "contents"),
      tabPanel("tab 4", "contents"),
      tabPanel("tab 5", "contents")
    )
)
```



dashboardPage()

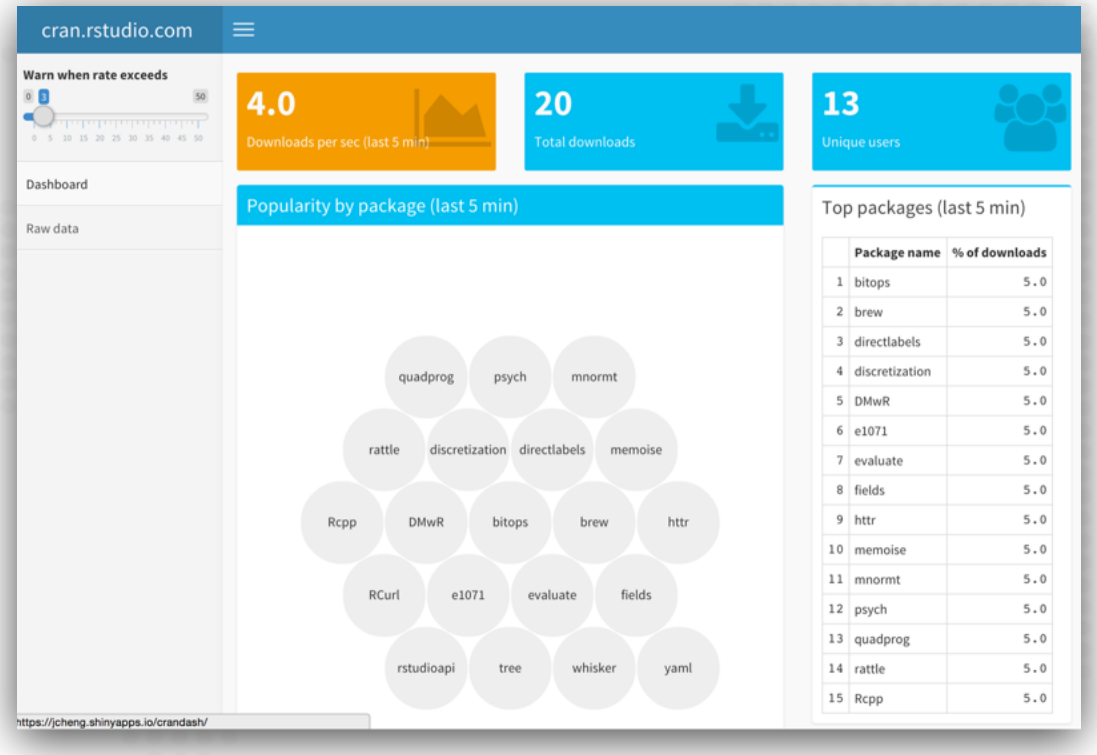
`dashboardPage()` comes in the shinydashboard package

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



shinydashboard

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



A package of layout functions for building administrative dashboards with Shiny

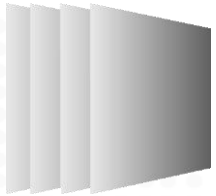
Dynamic Dashboards with Shiny Webinar:

www.rstudio.com/resources/webinars/

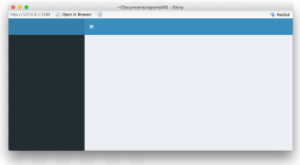
Recap: Prepackaged Layouts

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

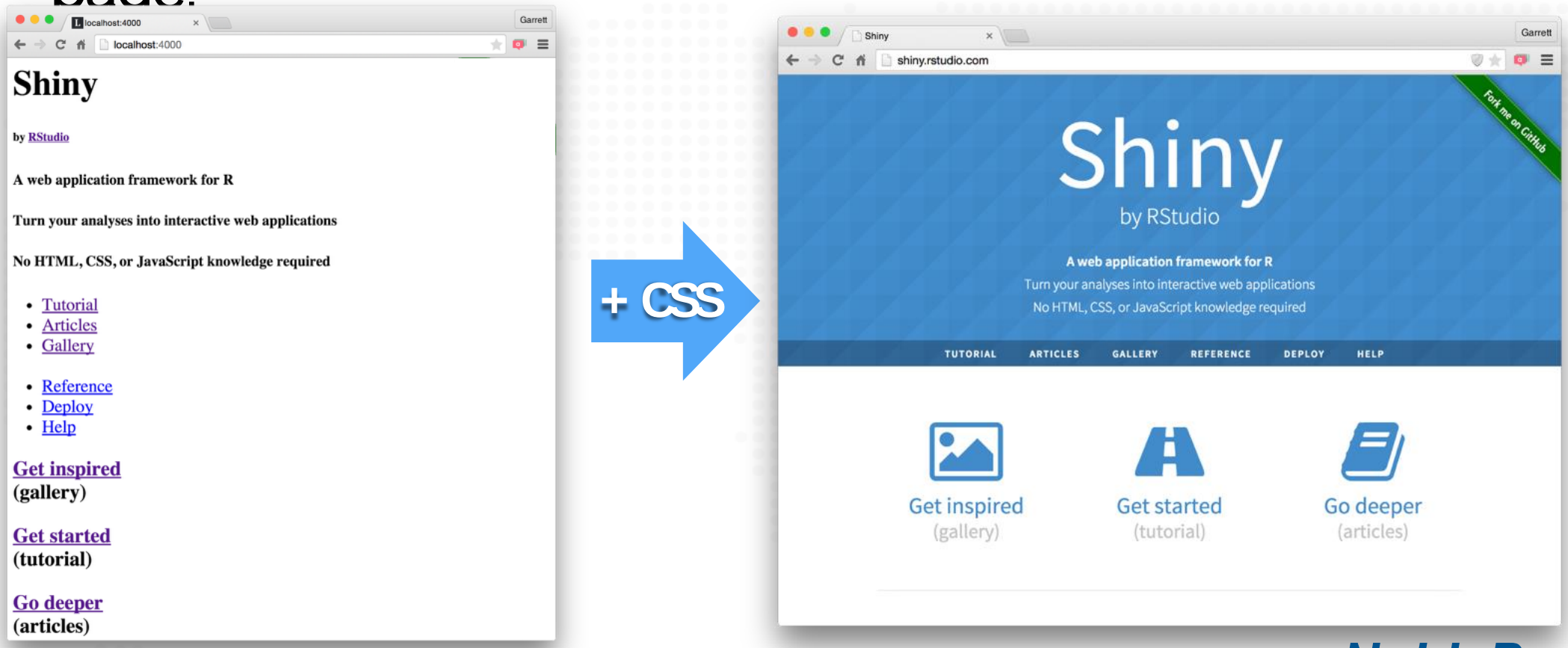
NobleProg

CSS

NobleProg

What is CSS?

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



Bootstrap

Shiny uses the Bootstrap 3 CSS framework,
getbootstrap.com

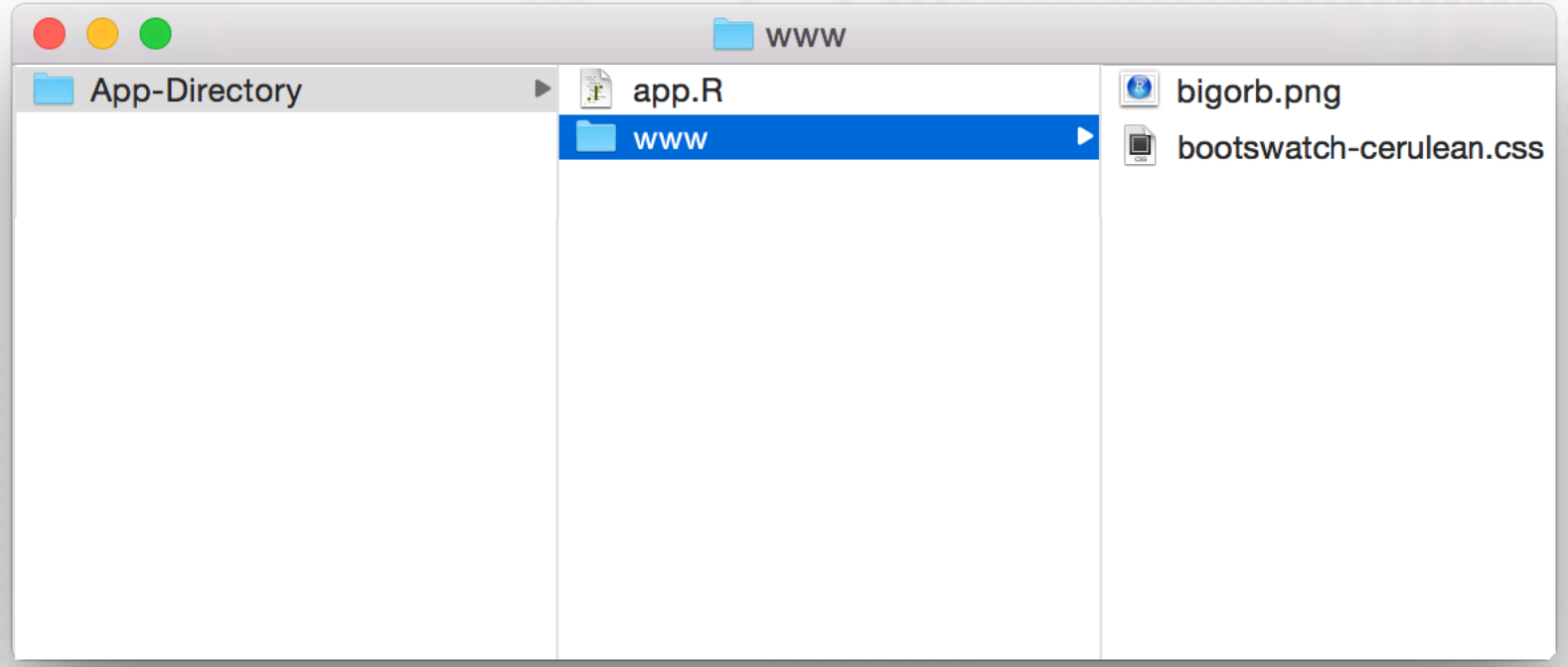
```
fluidPage()
```

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


NobleProg

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$head(
   tags$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>
```


NobleProg

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>
```


NobleProg

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;")
)
```

Experiment and practice

build your own apps

The Shiny Development Center

shiny.rstudio.com