Leveraging Shiny modules

Carson Sievert, PhD
Senior Software Engineer @ Posit

bit.ly/shiny-modules-slides

Why Shiny modules?

- Avoid repeating logic
- Make your work more reusable

Why ~~Shiny modules?~~ R functions?

- Avoid repeating logic
- Make your work more reusable

Why Shiny modules?

- Avoid repeating logic
- Make your work more reusable

Why Shiny modules?

- Avoid repeating Shiny UI/Server logic
- Make your work more reusable

Why Shiny modules?

- Avoid repeating Shiny UI/Server logic
- Make your work more reusable to Shiny app developers

You might not need a module!

- Function(s) might be a better, simpler, more general, way to share your work

```
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color))
```

```r
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
```

```
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2))
```

```
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
```

```r
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data3) +
  geom_point(aes(x=x3, y=y3, color=color3)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
```

```r
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data3) +
  geom_point(aes(x=x3, y=y3, color=color3)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
```

Tedious & error-prone to change theming details!

```r
library(ggplot2)

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )

ggplot(data3) +
  geom_point(aes(x=x3, y=y3, color=color3)) +
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
```

```r
library(ggplot2)

theme_jnj <- function() {
  theme_bw(base_family = "Gill Sans", base_size = 25) +
  theme(
    axis.ticks = element_blank(),
    axis.text = element_text(color = "#D71500"),
  )
}

ggplot(data) +
  geom_point(aes(x=x, y=y, color=color)) +
  theme_jnj()

ggplot(data2) +
  geom_point(aes(x=x2, y=y2, color=color2)) +
  theme_jnj()

ggplot(data3) +
  geom_point(aes(x=x3, y=y3, color=color3)) +
  theme_jnj()
```

```r
library(ggplot2)

jnj_scatterplot ← function(data, x, y, color) {
  ggplot(data) +
    geom_point(aes_string(x=x, y=y, color=color)) +
    theme_bw(base_family = "Gill Sans", base_size = 25) +
    theme(
      axis.ticks = element_blank(),
      axis.text = element_text(color = "#D71500"),
    )
}

jnj_scatterplot(data, "x", "y", "color")
jnj_scatterplot(data2, "x2", "y2", "color2")
jnj_scatterplot(data3, "x3", "y3", "color3")
```

✨ Let's make a Shiny app! ✨

bit.ly/shiny-modules-cloud

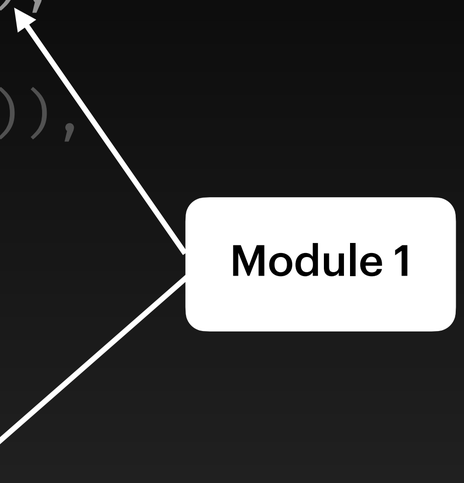
We'll motivate modules via app.R & app2.R

```r
ui ← fluidPage(
  selectInput("color", "Select color", names(mtcars)),
  plotOutput("plot"),
  selectInput("color2", "Select color", names(quakes)),
  plotOutput("plot2")
)

server ← function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
  output$plot2 ← renderPlot({
    jnj_scatterplot(quakes, "long", "lat", input$color2)
  })
}
```

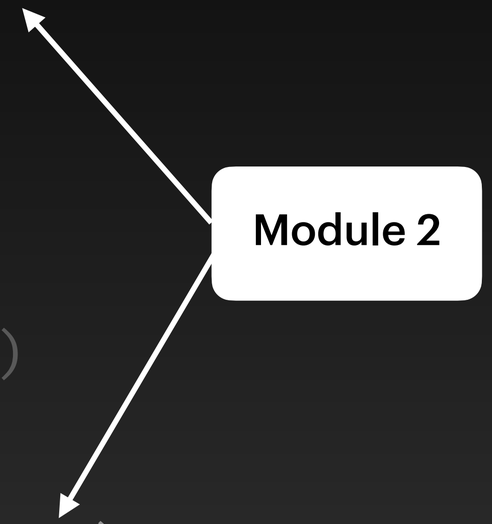
```
ui ← fluidPage(
  selectInput("color", "Select color", names(mtcars)),
  plotOutput("plot"),
  selectInput("color2", "Select color", names(quakes)),
  plotOutput("plot2")
)

server ← function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
  output$plot2 ← renderPlot({
    jnj_scatterplot(quakes, "long", "lat", input$color2)
  })
}
```



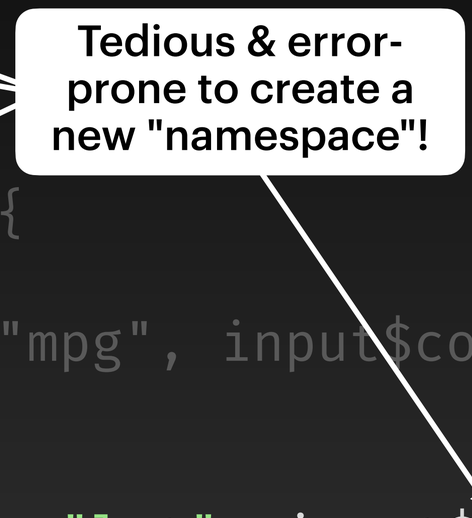
```r
ui ← fluidPage(
  selectInput("color", "Select color", names(mtcars)),
  plotOutput("plot"),
  selectInput("color2", "Select color", names(quakes)),
  plotOutput("plot2")
)

server ← function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
  output$plot2 ← renderPlot({
    jnj_scatterplot(quakes, "long", "lat", input$color2)
  })
}
```



```r
ui ← fluidPage(
  selectInput("color", "Select color", names(mtcars)),
  plotOutput("plot"),
  selectInput("color2", "Select color", names(quakes)),
  plotOutput("plot2")
)

server ← function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
  output$plot2 ← renderPlot({
    jnj_scatterplot(quakes, "long", "lat", input$color2)
  })
}
```



```r
ui ← fluidPage(
  selectInput("color", "Select color", names(mtcars)),
  plotOutput("plot"),
  selectInput("color2", "Select color", names(quakes)),
  plotOutput("plot2")
)

server ← function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
  output$plot2 ← renderPlot({
    jnj_scatterplot(quakes, "long", "lat", input$color2)
  })
}
```

```
function() {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot <- renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 1: Module UI

```
function() {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

By convention,
{module_name}_ui()

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

The module "namespace" (users of the module provide this)

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput("color", "Select color", names(mtcars)),
    plotOutput("plot")
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput(NS(id, "color"), "Select color", names(mtcars)),
    plotOutput(NS(id, "plot"))
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id) {
  list(
    selectInput(NS(id, "color"), "Select color", names(mtcars)),
    plotOutput(NS(id, "plot"))
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Should vary across modules!

Step 1: Module UI

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

function(input, output) {
  output$plot ← renderPlot({
    jnj_scatterplot(mtcars, "wt", "mpg", input$color)
  })
}
```

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  }
}
```

By convention,
{module_name}_server()

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  }
}
```

The module "namespace" (users of the module provide this)

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  })
}
```

Step 2: Module Server

```r
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  })
}
```

Akin to NS(id) in the UI, except it adds the namespace to input$id and output$id

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  })
}
```

Remember to extract out logic that's allowed to vary!

Step 2: Module Server

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(mtcars, "wt", "mpg", input$color)
    })
  })
}
```

Step 2: Module Server

```r
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id, data, x, y) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(data, x, y, input$color)
    })
  })
}
```

The module API

```
jnj_scatterplot_ui ← function(id, data) {
  list(
    selectInput(NS(id, "color"), "Select color", names(data)),
    plotOutput(NS(id, "plot"))
  )
}

jnj_scatterplot_server ← function(id, data, x, y) {
  moduleServer(id, function(input, output) {
    output$plot ← renderPlot({
      jnj_scatterplot(data, x, y, input$color)
    })
  })
}
```

The module API

```
jnj_scatterplot_ui(id, data)
```

```
jnj_scatterplot_server(id, data, x, y)
```

Using the module

```
jnj_scatterplot_ui("mtcars", mtcars)
```

```
jnj_scatterplot_server("mtcars", mtcars, "wt", "mpg")
```

Using the module

```
ui ← fluidPage(
  jnj_scatterplot_ui("mtcars", mtcars)
)


server ← function(input, output) {
  jnj_scatterplot_server("mtcars", mtcars, "wt", "mpg")
}
```

Using the module

```
ui ← fluidPage(
  jnj_scatterplot_ui("mtcars", mtcars),
  jnj_scatterplot_ui("quakes", quakes)
)

server ← function(input, output) {
  jnj_scatterplot_server("mtcars", mtcars, "wt", "mpg")
  jnj_scatterplot_server("quakes", quakes, "long", "lat")
}
```

Exercise time! 💪

bit.ly/shiny-modules-cloud

Open/edit the exercises.Rmd file

In summary

- Modules, like functions, help avoid duplication
- Modules, unlike functions, assumes users wants a Shiny app
- A module is just a pair of (UI/server) functions
  - Put them in a package for better sharing
  - A package also forces you to document 😉

In summary

- Leverage function parameters to vary things across modules
- To share reactive state, pass `reactive()`s from the top-level server down to each module server

Learn more

- Mastering Shiny
  - Chapter 19: Shiny modules
- Shiny articles
  - Modules overview
  - Module communication
- Shiny for Python
  - Modules overview

Questions?

- Ask me anything Shiny-related
  - What's new in bslib?
  - What's new with Shiny for Python?