

Different Approaches to Shiny App Development

By Andrés F. Quintero

Who am I?



- Software Engineer from Colombia
- Specialise in building Shiny Apps
- Currently work building Shiny Applications for use in healthcare institutions
- Free/Open Source advocate and enthusiast



andresquinterom



andyquinterom

What is this presentation about?

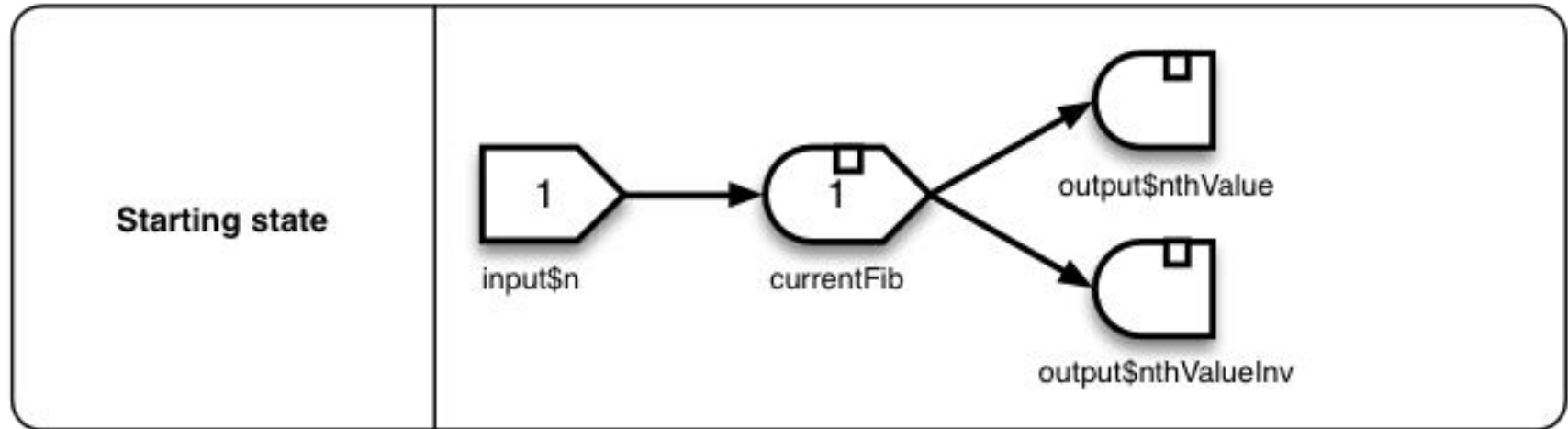
- The different approaches to server-side logic inside Shiny Applications.
- Geared toward Shiny developers looking to learn about how different teams or people could write Shiny Apps.
- Three different methods I have personally used extensively.

What do I mean by approach?

- Consistency and restrictions

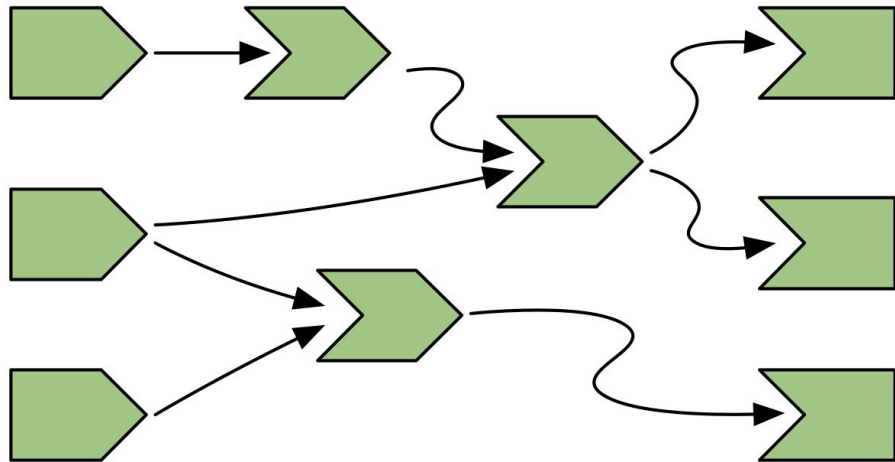
“Functional” approach

- Reactives are immutable and “pre-defined”.



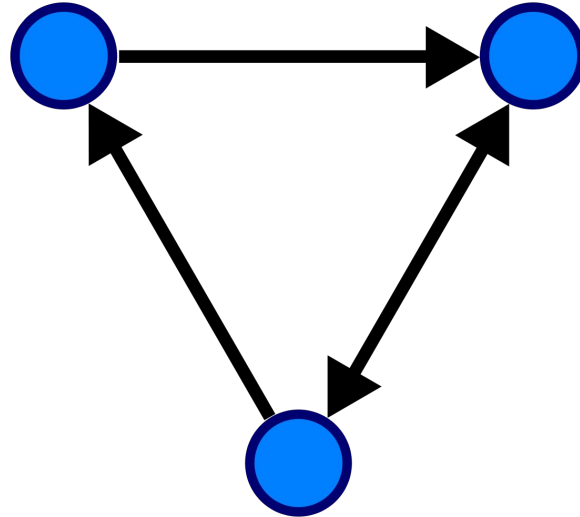
Benefits to the functional approach

- Most predictable
- Clean reactive graph
- “Cleaner” code



Downsides to functional approach

- Can be hard to manage non-linear reactive graphs.



```

4 ui <- fluidPage(
5
6   titlePanel("Functional"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   # Read dataset based on input file name
34   dataset <- reactive({
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%
36     readr::read_csv()
37   })
38
39   # Update select input based on the dataset selected by user
40   observe({
41     updateSelectizeInput(
42       inputId = "variables",
43       choices = colnames(dataset())
44     )
45   })
46
47   # Select only the variables of the data set the user selects
48   filtered_data <- reactive({
49     dataset() %>%
50     dplyr::select(!!!rlang::syms(input$variables))
51   }) %>%
52     bindEvent(input$apply_changes)
53
54   # Render de filtered dataset
55   output$data <- renderDataTable({
56     filtered_data()
57   })
58
59 }

```



```

4 ui <- fluidPage(
5
6   titlePanel("Functional"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   # Read dataset based on input file name
34   dataset <- reactive({
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%
36     readr::read_csv()
37   })
38
39   # Update select input based on the dataset selected by user
40   observe({
41     updateSelectizeInput(
42       inputId = "variables",
43       choices = colnames(dataset())
44     )
45   })
46
47   # Select only the variables of the data set the user selects
48   filtered_data <- reactive({
49     dataset() %>%
50     dplyr::select(!!!rlang::syms(input$variables))
51   }) %>%
52     bindEvent(input$apply_changes)
53
54   # Render de filtered dataset
55   output$data <- renderDataTable({
56     filtered_data()
57   })
58
59 }

```

```

4 ui <- fluidPage(
5
6   titlePanel("Functional"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   # Read dataset based on input file name
34   dataset <- reactive({
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%
36     readr::read_csv()
37   })
38
39   # Update select input based on the dataset selected by user
40   observe({
41     updateSelectizeInput(
42       inputId = "variables",
43       choices = colnames(dataset())
44     )
45   })
46
47   # Select only the variables of the data set the user selects
48   filtered_data <- reactive({
49     dataset() %>%
50     dplyr::select(!!!rlang::syms(input$variables))
51   }) %>%
52     bindEvent(input$apply_changes)
53
54   # Render de filtered dataset
55   output$data <- renderDataTable({
56     filtered_data()
57   })
58
59 }

```

```

4 ui <- fluidPage(
5
6   titlePanel("Functional"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   # Read dataset based on input file name
34   dataset <- reactive({
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%
36     readr::read_csv()
37   })
38
39   # Update select input based on the dataset selected by user
40   observe({
41     updateSelectizeInput(
42       inputId = "variables",
43       choices = colnames(dataset())
44     )
45   })
46
47   # Select only the variables of the data set the user selects
48   filtered_data <- reactive({
49     dataset() %>%
50     dplyr::select(!!!rlang::syms(input$variables))
51   }) %>%
52   bindEvent(input$apply_changes)
53
54   # Render de filtered dataset
55   output$data <- renderDataTable({
56     filtered_data()
57   })
58
59 }

```

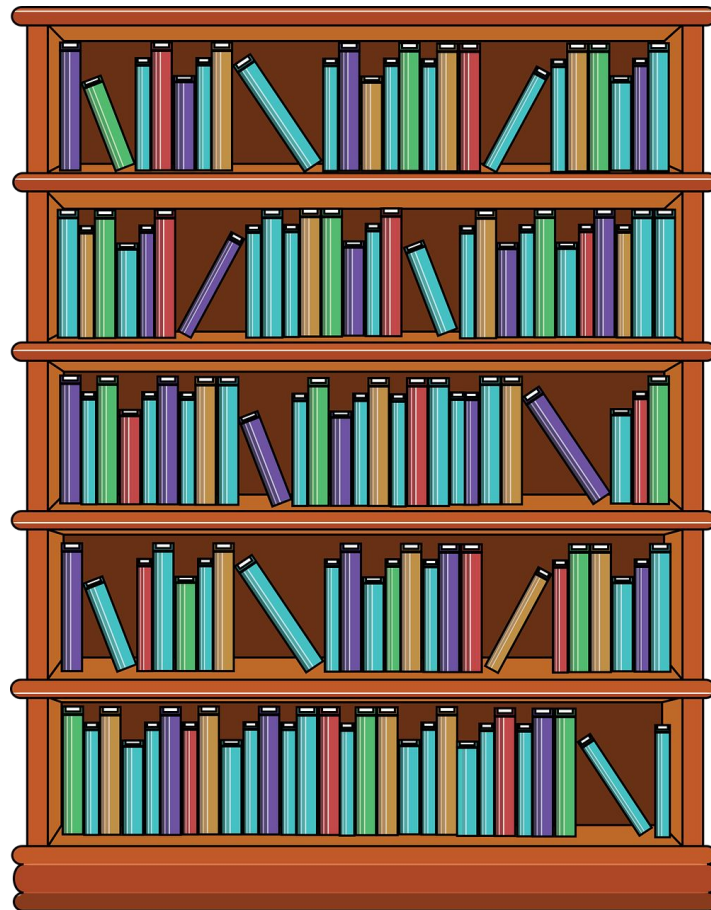


```
4 ui <- fluidPage(  
5  
6   titlePanel("Functional"),  
7   sidebarLayout(  
8     sidebarPanel(  
9       selectizeInput(  
10         inputId = "file_name",  
11         label = "Dataset",  
12         choices = c("mtcars", "iris")  
13       ),  
14       selectizeInput(  
15         inputId = "variables",  
16         label = "Variables",  
17         choices = NULL,  
18         multiple = TRUE  
19       ),  
20       actionButton(  
21         inputId = "apply_changes",  
22         label = "Apply changes"  
23       )  
24     ),  
25     mainPanel(  
26       dataTableOutput("data")  
27     )  
28   )  
29 )
```

```
31 server <- function(input, output) {  
32  
33   # Read dataset based on input file name  
34   dataset <- reactive({  
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%  
36     readr::read_csv()  
37   })  
38  
39   # Update select input based on the dataset selected by user  
40   observe({  
41     updateSelectizeInput(  
42       inputId = "variables",  
43       choices = colnames(dataset())  
44     )  
45   })  
46  
47   # Select only the variables of the data set the user selects  
48   filtered_data <- reactive({  
49     dataset() %>%  
50     dplyr::select(!!!rlang::syms(input$variables))  
51   }) %>%  
52   bindEvent(input$apply_changes)  
53  
54   # Render de filtered dataset  
55   output$data <- renderDataTable({  
56     filtered_data()  
57   })  
58  
59 }
```

Imperative Approach

Saves state in reactive values.



Benefits of imperative approach

- Easier to manage non-linear workflows
- Reactives are mutable and can be created dynamically



Downsides to imperative approach

- “Complicated” reactive graphs.
- If not managed carefully, bugs can creep in.
- Keeping track of messy data gets hard.




```

4 ui <- fluidPage(
5
6   titlePanel("Imperative"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40   })
41
42   # Update select input based on the dataset selected by user
43   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48   })
49
50   # Select only the variables of the data set the user selects
51   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54       dplyr::select(!!!rlang::syms(input$variables))
55   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62   })
63
64 }

```



```

4 ui <- fluidPage(
5
6   titlePanel("Imperative"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40   })
41
42   # Update select input based on the dataset selected by user
43   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48   })
49
50   # Select only the variables of the data set the user selects
51   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54       dplyr::select(!!!rlang::syms(input$variables))
55   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62   })
63
64 }

```

```

4 ui <- fluidPage(
5
6   titlePanel("Imperative"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40   })
41
42   # Update select input based on the dataset selected by user
43   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48   })
49
50   # Select only the variables of the data set the user selects
51   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54       dplyr::select(!!!rlang::syms(input$variables))
55   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62   })
63
64 }

```

```

4 ui <- fluidPage(
5
6   titlePanel("Imperative"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40   })
41
42   # Update select input based on the dataset selected by user
43   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48   })
49
50   # Select only the variables of the data set the user selects
51   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54       dplyr::select(!!!rlang::syms(input$variables))
55   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62   })
63
64 }

```



```

4 ui <- fluidPage(
5
6   titlePanel("Imperative"),
7   sidebarLayout(
8     sidebarPanel(
9       selectizeInput(
10         inputId = "file_name",
11         label = "Dataset",
12         choices = c("mtcars", "iris")
13       ),
14       selectizeInput(
15         inputId = "variables",
16         label = "Variables",
17         choices = NULL,
18         multiple = TRUE
19       ),
20       actionButton(
21         inputId = "apply_changes",
22         label = "Apply changes"
23       )
24     ),
25     mainPanel(
26       dataTableOutput("data")
27     )
28   )
29 )

```

```

31 server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40   })
41
42   # Update select input based on the dataset selected by user
43   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48   })
49
50   # Select only the variables of the data set the user selects
51   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54       dplyr::select(!!!rlang::syms(input$variables))
55   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62   })
63
64 }

```

```

31 ▾ server <- function(input, output) {
32
33   # Read dataset based on input file name
34 ▾   dataset <- reactive({
35     file.path("data", glue::glue("{ input$file_name }.csv")) %>%
36     readr::read_csv()
37 ▲   })
38
39   # Update select input based on the dataset selected by user
40 ▾   observe({
41     updateSelectizeInput(
42       inputId = "variables",
43       choices = colnames(dataset())
44     )
45 ▲   })
46
47   # Select only the variables of the data set the user selects
48 ▾   filtered_data <- reactive({
49     dataset() %>%
50     dplyr::select(!!!rlang::syms(input$variables))
51 ▲   }) %>%
52     bindEvent(input$apply_changes)
53
54   # Render de filtered dataset
55 ▾   output$data <- renderDataTable({
56     filtered_data()
57 ▲   })
58
59 ▲ }

```

```

31 ▾ server <- function(input, output) {
32
33   globals <- reactiveValues()
34
35   # Read dataset based on input file name
36 ▾   observe({
37     globals$dataset <- readr::read_csv(
38       file.path("data", glue::glue("{ input$file_name }.csv"))
39     )
40 ▲   })
41
42   # Update select input based on the dataset selected by user
43 ▾   observe({
44     updateSelectizeInput(
45       inputId = "variables",
46       choices = colnames(globals$dataset)
47     )
48 ▲   })
49
50   # Select only the variables of the data set the user selects
51 ▾   observe({
52     req(globals$dataset)
53     globals$filtered_data <- globals$dataset %>%
54     dplyr::select(!!!rlang::syms(input$variables))
55 ▲   }) %>%
56     bindEvent(input$apply_changes)
57
58   # Render de filtered dataset
59 ▾   output$data <- renderDataTable({
60     req(globals$filtered_data)
61     globals$filtered_data
62 ▲   })
63
64 ▲ }

```

Object oriented approach

Data is stored in objects which can be accessed from anywhere in the app
(Objects are not reactive).



Benefits of the object oriented approach

- “Easier” to organize data
- Data can be shared throughout the application without interfering with the reactive graph
- Great for bigger apps with many data sources and types.

Downsides to the object oriented approach

- Can be more work.
- Works against the general notion of how R code is usually written.
- Does not interfere the reactive graph.
- Harder to mix with other approaches.


```
6 dataset <- R6Class(  
7   "APP Data",  
8   list(  
9     data = mtcars,  
10    filtered_data = data.frame(),  
11    set = function(data) {  
12      self$data <- data  
13    },  
14    select_vars = function(vars) {  
15      self$filtered_data <- self$data %>%  
16        dplyr::select({{ vars }})  
17    }  
18  )  
19 )
```

```
6 dataset <- R6Class(  
7   "APP Data",  
8   list(  
9     data = mtcars,  
10    filtered_data = data.frame(),  
11    set = function(data) {  
12      self$data <- data  
13    },  
14    select_vars = function(vars) {  
15      self$filtered_data <- self$data %>%  
16        dplyr::select({{ vars }})  
17    }  
18  )  
19 )
```

```
6 dataset <- R6Class(  
7   "APP Data",  
8   list(  
9     data = mtcars,  
10    filtered_data = data.frame(),  
11    set = function(data) {  
12      self$data <- data  
13    },  
14    select_vars = function(vars) {  
15      self$filtered_data <- self$data %>%  
16        dplyr::select({{ vars }})  
17    }  
18  )  
19 )
```

```
6 dataset <- R6Class(  
7   "APP Data",  
8   list(  
9     data = mtcars,  
10    filtered_data = data.frame(),  
11    set = function(data) {  
12      self$data <- data  
13    },  
14    select_vars = function(vars) {  
15      self$filtered_data <- self$data %>%  
16        dplyr::select({{ vars }})  
17    }  
18  )  
19 )
```

```

50 server <- function(input, output, session) {
51
52   app_data <- dataset$new()
53   init("selected_dataset", "selected_variables")
54   # Read dataset based on input file name
55   observe({
56     app_data$set(
57       file.path("data", glue::glue("{ input$file_name }.csv")) %>%
58       readr::read_csv()
59     )
60     trigger("selected_dataset")
61   }) %>%
62     bindEvent(input$file_name)
63
64   # Update select input based on the dataset selected by user
65   observe({
66     updateSelectizeInput(
67       inputId = "variables",
68       choices = colnames(app_data$data)
69     )
70   }) %>%
71     bindEvent(watch("selected_dataset"))
72
73   # Select only the variables of the data set the user selects
74   observe({
75     app_data$select_vars(input$variables)
76     trigger("selected_variables")
77   }) %>%
78     bindEvent(input$apply_changes)
79
80   # Render de filtered dataset
81   output$data <- renderDataTable({
82     app_data$filtered_data
83   }) %>%
84     bindEvent(watch("selected_variables"))
85
86 }

```

Initialize a new object

Init triggers

Set the data to that which the user selected

Trigger "selected_dataset"

Run when "selected_dataset" is triggered

Set selected variables to those in input\$variables and trigger "selected_variables"

Render the data when "selected_variables" is triggered

Conclusion

- Most approaches are completely valid for production grade applications.
- It is important to determine which approach is easier to work with for a particular project.
- Consistency is key, code is understandable as long as it is consistent.





- Software Engineer
- Specialise in building Shiny Apps
- Currently work building Shiny Applications for use in healthcare institutions
- Free/Open Source advocate and enthusiast



andresquinterom



andyquinterom