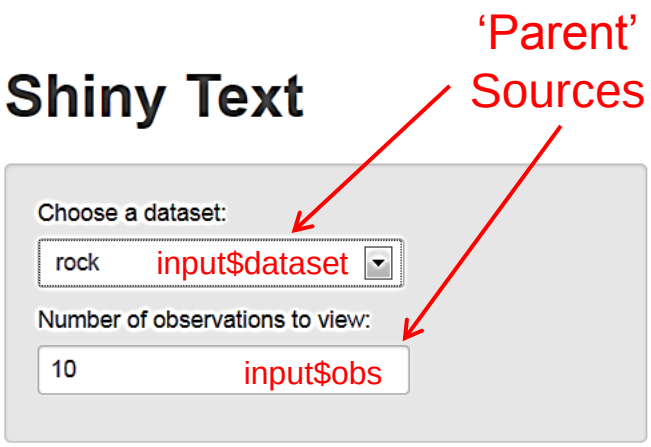
Notes on Initial Shiny Examples

“Shiny Text” Example 2



When **input$dataset** changes, both **output$summary** and **output$view** change.

```
      area            peri
 Min.   : 1016   Min.   : 308.6
 1st Qu.: 5305   1st Qu.:1414.9
 Median : 7487   Median :2536.2
 Mean   : 7188   Mean   :2682.2
 3rd Qu.: 8870   3rd Qu.:3989.5
 Max.   :12212   Max.   :4864.2
     shape             perm
 Min.   :0.09033   Min.   :   6.30
 1st Qu.:0.16226   1st Qu.:  76.45
 Median :0.19886   Median : 130.50
 Mean   :0.21811   Mean   : 415.45
 3rd Qu.:0.26267   3rd Qu.: 777.50
 Max.   :0.46413   Max.   :1300.00
```

‘Dependent’ Endpoints

output$summary

	area	peri	shape	perm
1	4990	2791.90	0.09	6.30
2	7002	3892.60	0.15	6.30
3	7558	3930.66	0.18	6.30
4	7352	3869.32	0.12	6.30
5	7943	3948.54	0.12	17.10
6	7979	4010.15	0.17	17.10

output$view

But when **input$obs** changes, only **output$view** changes.

“Shiny Text” Example 2 ui.R

```
## Shiny Text Example 2 ui.R
library(shiny)

# Define UI for dataset viewer application
shinyUI(pageWithSidebar(

  # Application title
  headerPanel("Shiny Text"),

  # Sidebar with controls to select a dataset and
specify the number
  # of observations to view
  sidebarPanel(
    selectInput("dataset", "Choose a dataset:",
                choices = c("rock", "pressure", "cars")),

    numericInput("obs", "Number of observations to
view:", 10)
  ),

  # Show a summary of the dataset and an HTML
table with the requested
  # number of observations
  mainPanel(
    verbatimTextOutput("summary"),

    tableOutput("view")
  )
))
```

"Shiny Text" Example 2 server.R

When **input$dataset** changes, both **output$summary** and **output$view** change.

Why? Because **datasetInput** serves as a ***reactive conductor*** implemented as a ***reactive expression.***

When **input$obs** changes, only **output$view** changes.

```
## Shiny Text Example 2 server.R

library(shiny)
library(datasets)

# Define server logic required to summarize and
view the selected dataset
shinyServer(function(input, output) {

  # Return the requested dataset
  datasetInput <- reactive({
    switch(input$dataset,
           "rock" = rock,
           "pressure" = pressure,
           "cars" = cars)
  })

  # Generate a summary of the dataset
  output$summary <- renderPrint({
    dataset <- datasetInput()
    summary(dataset)
  })

  # Show the first "n" observations
  output$view <- renderTable({
    head(datasetInput(), n = input$obs)
  })
})
```

Change to **input$dataset** changes **datasetInput**

datasetInput is called here reactively

datasetInput is called here reactively

“Reactivity” Example 3

Reactivity

Caption input change immediately updates output “caption” label

Caption:

Data Summary

Choose a dataset:

rock

Number of observations to view:

10

Data Summary

```
      area            peri
 Min.   : 1016   Min.   : 308.6
 1st Qu.: 5305   1st Qu.:1414.9
 Median : 7487   Median :2536.2
 Mean   : 7188   Mean   :2682.2
 3rd Qu.: 8870   3rd Qu.:3989.5
 Max.   :12212   Max.   :4864.2
     shape              perm
 Min.   :0.09033   Min.   :   6.30
 1st Qu.:0.16226   1st Qu.:  76.45
 Median :0.19886   Median : 130.50
 Mean   :0.21811   Mean   : 415.45
 3rd Qu.:0.26267   3rd Qu.: 777.50
 Max.   :0.46413   Max.   :1300.00
```

	area	peri	shape	perm
1	4990	2791.90	0.09	6.30
2	7002	3892.60	0.15	6.30
3	7558	3930.66	0.18	6.30
4	7352	3869.32	0.12	6.30

“Reactivity” Example 3 ui.R

Changes from Example 2 “Shiny Text” ui.R file highlighted

```
## Reactivity Example 3 ui.R
library(shiny)

# Define UI for dataset viewer application
shinyUI(pageWithSidebar(

  # Application title
  headerPanel("Reactivity"),

  # Sidebar with controls to provide a caption, select a dataset, and
  # specify the number of observations to view. Note that changes made
  # to the caption in the textInput control are updated in the output
  # area immediately as you type
  sidebarPanel(
    textInput("caption", "Caption:", "Data Summary"),

    selectInput("dataset", "Choose a dataset:",
                choices = c("rock", "pressure", "cars")),

    numericInput("obs", "Number of observations to view:", 10)
  ),
```

Using “Reactivity” as header instead of “Shiny Text”

Added textInput() function added

“Reactivity” Example 3 ui.R (cont’d)

```
  # Show the caption, a summary of the dataset and an HTML table with
  # the requested number of observations
  mainPanel(
    h3(textOutput("caption")),

    verbatimTextOutput("summary"),

    tableOutput("view")
  )
))
```

“caption” string from textInput() immediately assigned to textOutput() expression

“Reactivity” Example 3 server.R

```
## Reactivity Example 3 server.R
library(shiny)
library(datasets)
```

Changes from Example 2 “Shiny Text” server.R file highlighted

```
# Define server logic required to summarize and view the selected dataset
shinyServer(function(input, output) {

  # By declaring datasetInput as a reactive expression we ensure that:
  #
  #  1) It is only called when the inputs it depends on changes
  #  2) The computation and result are shared by all the callers (it
  #     only executes a single time)
  #  3) When the inputs change and the expression is re-executed, the
  #     new result is compared to the previous result; if the two are
  #     identical, then the callers are not notified
  #
  datasetInput <- reactive({
    switch(input$dataset,
           "rock" = rock,
           "pressure" = pressure,
           "cars" = cars)
  })
```

"Reactivity" Example 3 server.R

Changes from Example 2 "Shiny Text" server.R file highlighted

```
## Reactivity Example 3 server.R (continued from previous slide)
# The output$caption is computed based on a reactive expression that
# returns input$caption. When the user changes the "caption" field:
#
#  1) This expression is automatically called to recompute the output
#  2) The new caption is pushed back to the browser for re-display
#
# Note that because the data-oriented reactive expression below don't
# depend on input$caption, those expression are NOT called when
# input$caption changes.
output$caption <- renderText({
  input$caption
})

# The output$summary depends on the datasetInput reactive expression,
# so will be re-executed whenever datasetInput is re-executed
# (i.e. whenever the input$dataset changes)
output$summary <- renderPrint({
  dataset <- datasetInput()
  summary(dataset)
})

# The output$view depends on both the databaseInput reactive expression
# and input$obs, so will be re-executed whenever input$dataset or
# input$obs is changed.
output$view <- renderTable({
  head(datasetInput(), n = input$obs)
})
})
```

"caption" string in input object (input$caption) immediately assigned to output object (output$caption) in a reactive expression.