

This tutorial is deprecated. Learn more about Shiny at our new location, shiny.rstudio.com.

GETTING STARTED

Welcome
Hello Shiny
Shiny Text
Reactivity

BUILDING AN APP

UI & Server
Inputs & Outputs
Run & Debug

TOOLING UP

Sliders
Tabsets
DataTables
More Widgets

Uploading Files

Downloading Data

HTML UI
Dynamic UI

ADVANCED SHINY

Scoping
Client Data
Sending Images

UNDERSTANDING REACTIVITY

Reactivity Overview
Execution Scheduling
Isolation

DEPLOYING AND SHARING APPS

Deploying Over the Web
Sharing Apps to Run Locally

EXTENDING SHINY

Building Inputs
Building Outputs

Choose CSV File

/home/jcheng/Downloads/NST_I

Browse...

Upload complete

☒ Header

Separator

☒ Comma

☐ Semicolon

☐ Tab

Quote

☐ None

☒ Double Quote

☐ Single Quote

	Sumlev	Region	Division	State	Name	CENSUS2010POP	EST
1	10	0	0	0	United States	308745538	
2	20	1	0	0	Northeast Region	55317240	
3	20	2	0	0	Midwest Region	66927001	
4	20	3	0	0	South Region	114555744	
5	20	4	0	0	West Region	71945553	
6	40	3	6	1	Alabama	4779736	
7	40	4	9	2	Alaska	710231	
8	40	4	8	4	Arizona	6392017	
9	40	3	7	5	Arkansas	2915918	
10	40	4	9	6	California	37253956	
11	40	4	8	8	Colorado	5029196	
12	40	1	1	9	Connecticut	3574097	
13	40	3	5	10	Delaware	897934	
14	40	3	5	11	District of Columbia	601723	
15	40	3	5	12	Florida	18801310	
16	40	3	5	13	Georgia	9687653	

Sometimes you’ll want users to be able to upload their own data to your application. Shiny makes it easy to offer your users file uploads straight from the browser, which you can then access from your server logic.

Important notes:

- This feature does not work with Internet Explorer 9 and earlier (not even with Shiny Server).
- By default, Shiny limits file uploads to 5MB per file. You can modify this limit by using the shiny.maxRequestSize option. For example, adding options(shiny.maxRequestSize=30*1024^2) to the top of server.R would increase the limit to 30MB.

To run this example, type:

```
> library(shiny)
> runExample("09_upload")
```

File upload controls are created by using the fileInput function in your ui.R file. You access the uploaded data similarly to other types of input: by referring to input\$inputId. The fileInput function takes a multiple parameter that can be set to TRUE to allow the user to select multiple files, and an accept parameter can be used to give the user clues as to what kind of files the application expects.

ui.R

```
shinyUI(pageWithSidebar(
  headerPanel("CSV Viewer"),
  sidebarPanel(
    fileInput('file1', 'Choose CSV File',
      accept=c('text/csv', 'text/comma-separated-values,text/plain', '.csv')),
    tags$hr(),
    checkboxInput('header', 'Header', TRUE),
    radioButtons('sep', 'Separator',
      c(Comma=',',
        Semicolon=';',
        Tab='\t'),
      'Comma'),
    radioButtons('quote', 'Quote',
      c(None='',
        'Double Quote'='"',
        'Single Quote'='"'),
      'Double Quote')
  ),
  mainPanel(
    tableOutput('contents')
  )
))
```

server.R

```
shinyServer(function(input, output) {
  output$contents <- renderTable({

    # input$file1 will be NULL initially. After the user selects and uploads a
    # file, it will be a data frame with 'name', 'size', 'type', and 'datapath'
    # columns. The 'datapath' column will contain the local filenames where the
    # data can be found.

    inFile <- input$file1

    if (is.null(inFile))
      return(NULL)

    read.csv(inFile$datapath, header=input$header, sep=input$sep, quote=input$quote)
  })
})
```

This example receives a file and attempts to read it as comma-separated values using read.csv, then displays the results in a table. As the comment in server.R indicates, inFile is either NULL or a dataframe that contains one row per uploaded file. In this case, fileInput did not have the multiple parameter so we can assume there is only one row.

The file contents can be accessed by reading the file named by the datapath column. See the ?fileInput help topic to learn more about the other columns that are available.

← Previous

Next →