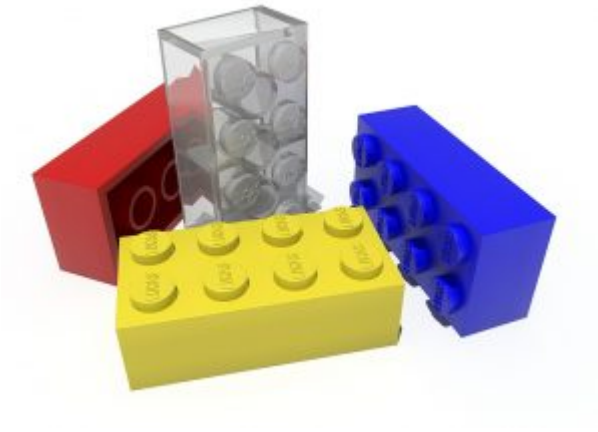
# Modularize your Shiny Apps: Exercises



Shiny modules are short (well, usually short) server and UI functions, that can be connected to each other by a common namespace, and be embedded within a regular Shiny app. You can't run a Shiny module without a parent Shiny app. The modules can contain both inputs and outputs, and are usually centered around a single operation or theme.

The biggest advantage of modules is the ability to efficiently reuse Shiny code, which can save a great deal of time. In addition, modules can help you standardize and scale your Shiny operations. Lastly, even if not reused, Shiny modules can help with organizing the code and break it into smaller pieces – which is very much needed in many complex Shiny apps. Some more information on Shiny modules can be found here.

In the following exercise set, you will practice the not-so-straightforward use of Shiny modules. The first four exercises are a warm-up, and will help you "refresh" on how to build each part of a Shiny module. In each of the last six exercises you will build a complete end-to-end module and run a minimal Shiny app to test it. Answers to the exercises are available here.

Two reminders before we begin:
* A typical UI function would take the argument id and start with the line ns <- NS(id).
* All input and output IDs within a UI function should be wrapped with the function like ns().

**Exercise 1**
Build a module-UI-function that provides a selectInput control, where the choices are LETTERS.

**Exercise 2**
Build the corresponding module-server-function, that prints the selected letter to the console.

**Exercise 3**
Build a regular UI object that contains the module-UI-function.

**Exercise 4**
Build a regular server function that calls the module you built in exercises 1 and 2.
Put together a minimal Shiny app that runs everything (e.g, shinyApp(ui = ui, server = server)).

**Exercise 5**
Adjust the module you built to show the selecter letter as a UI textOutput instead of printing it to the console.

**Learn more** about Shiny apps in the online courses Create Interactive Web Applications with the R Shiny Package and R Shiny Interactive Web Apps.

**Exercise 6**
In some cases you'll need the same module twice in a single Shiny app.
Build a minimal Shiny app that uses the module from exercise 5 twice.

**Exercise 7**
Still with the same "letters" module, add an option for the app developer to select the label of the selectInput, with the default value being "Select a letter". Call the module with a different value for the label than the default.

**Exercise 8**

Build a “contact form” module that contains a name (textInput), a subject (selectInput with dynamic choices, namely the person who uses the module can choose which choices to display) a message (textAreaInput) and a button (actionButton with a dynamic label). Upon clicking the button, all of the form information should be saved in a text file.

**Exercise 9**

Build a module that takes a number *n* using numericInput, and generates *n* elements of type textInput.

**Exercise 10**

Modules can also be nested within each other. Namely, one module can call anther module.

To practice that, create two modules, an “inner” one and an “outer” one.

Your minimal Shiny app should call the “outer” module, that in turn will call the “inner” module.

The module-server-functions should take an argument text, and render it as a textOutput.