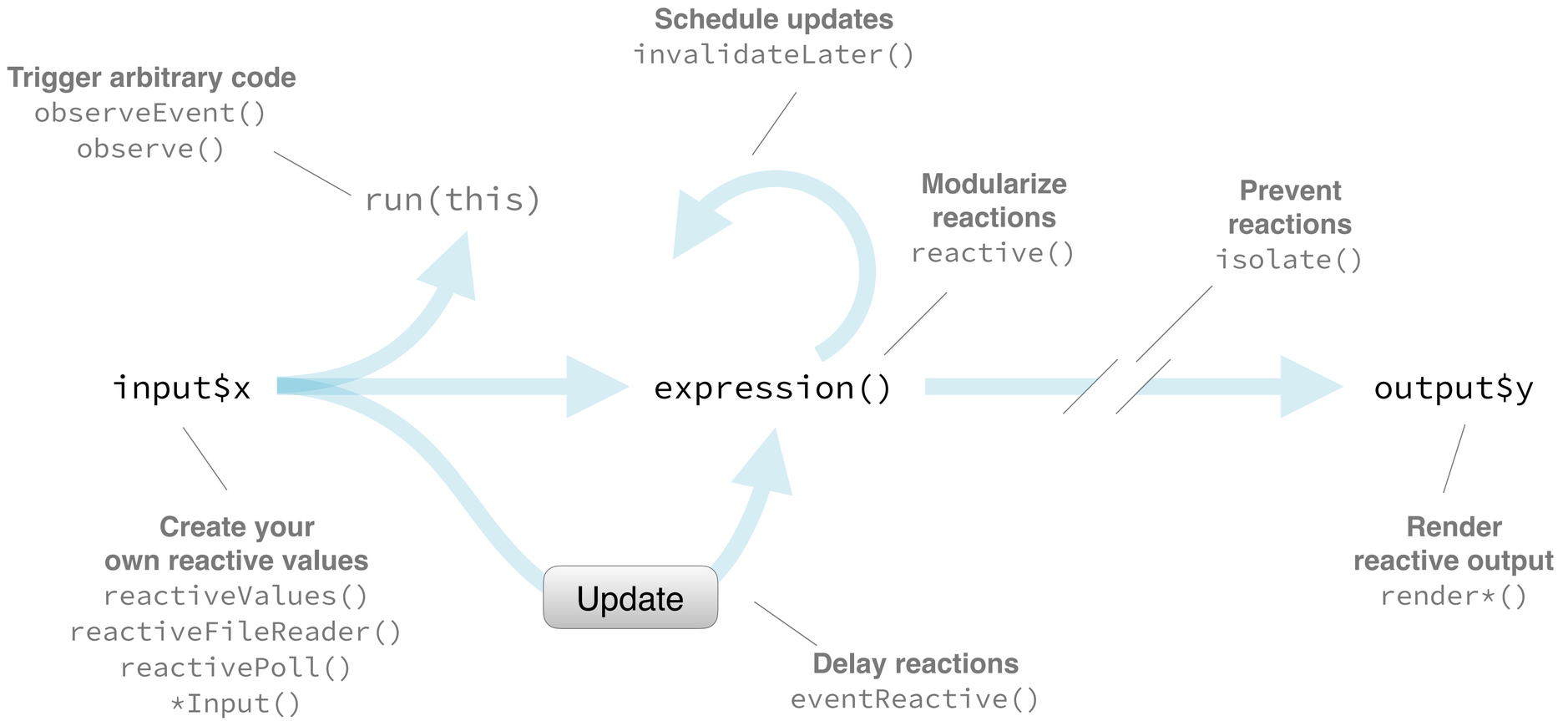
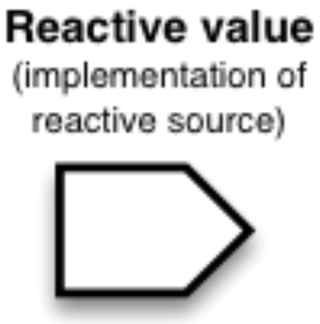
BUILDING WEB APPLICATIONS IN R WITH SHINY

Reactives and observers

Reactive flow



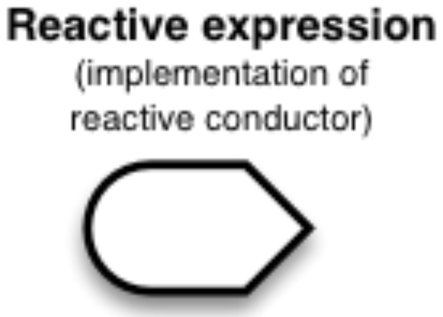
Implementation of reactive sources



`reactiveValues()`

e.g. `input$*:` Reactive value that looks like a list, and contains many individual reactive values that are set by input from the web browser.

Implementation of reactive conductors



`reactive()`

e.g. Reactive data frame subsets we created earlier.

- Can access reactive values or other reactive expressions, and they return a value
- Useful for caching the results of any procedure that happens in response to user input

Implementation of reactive endpoints

`observe()`

e.g. An `output$*` object is an observer. Actually what's going on under the hood is that a render function returns a reactive expression, and when you assign it to an `output$*` value, Shiny automatically creates an observer that uses the reactive expression.

- Can access reactive sources and reactive expressions, but they don't return a value
- They are used for their **side effects**, typically sending data to the web browser

Reactives vs. observers

- Similarities: Both store expressions that can be executed
- Differences:
  - Reactive expressions return values, but observers don't
  - Observers (and endpoints in general) eagerly respond to changes their dependencies, but reactive expressions (and conductors in general) do not
  - Reactive expressions must not have side effects, while observers are only useful for their side effects
- Most importantly:
  - `reactive()` is for calculating values, without side effects
  - `observe()` is for performing actions, with side effects
  - Do not use an `observe()` when calculating a value, and especially don't use `reactive()` for performing actions with side effects

Reactives vs. observers

	`reactive()`	`observer()`
Purpose	Calculations	Actions
Side effects	Forbidden	Allowed

BUILDING WEB APPLICATIONS IN R WITH SHINY

Let's practice!