

DATA VIZ STAGE

SHINY AVANZADO

Joshua Kunst Fuentes @jbkunst



BIENVENIDOS!

ANTES DE PARTIR

- Respositorio: <https://github.com/jbkunst/datapalooza-uc-2023-shiny-avanzado>, para clonarlo en posit.cloud.
- Presentación: <https://jkunst.com/datapalooza-uc-2023-shiny-avanzado/>.
- Un repaso: <https://jkunst.com/shiny-visualizacion-de-datos-con-R/clase-01.html>.

Lo QUE VEREMOS

Reactividad:

Expresiones reactivas, *Observers* y Eventos.

Módulos:

Definición, utilidades, patrón de uso.

HTMLWidgets:

Proxies y eventos.

HTML, CSS, JS:

Introducción, definiciones, etc.

COMO LO VEREMOS

- Definiciones y motivaciones.
- Revisión de ejemplos.
- Ejercicios!

QUE PAQUETES UTILIZAREMOS

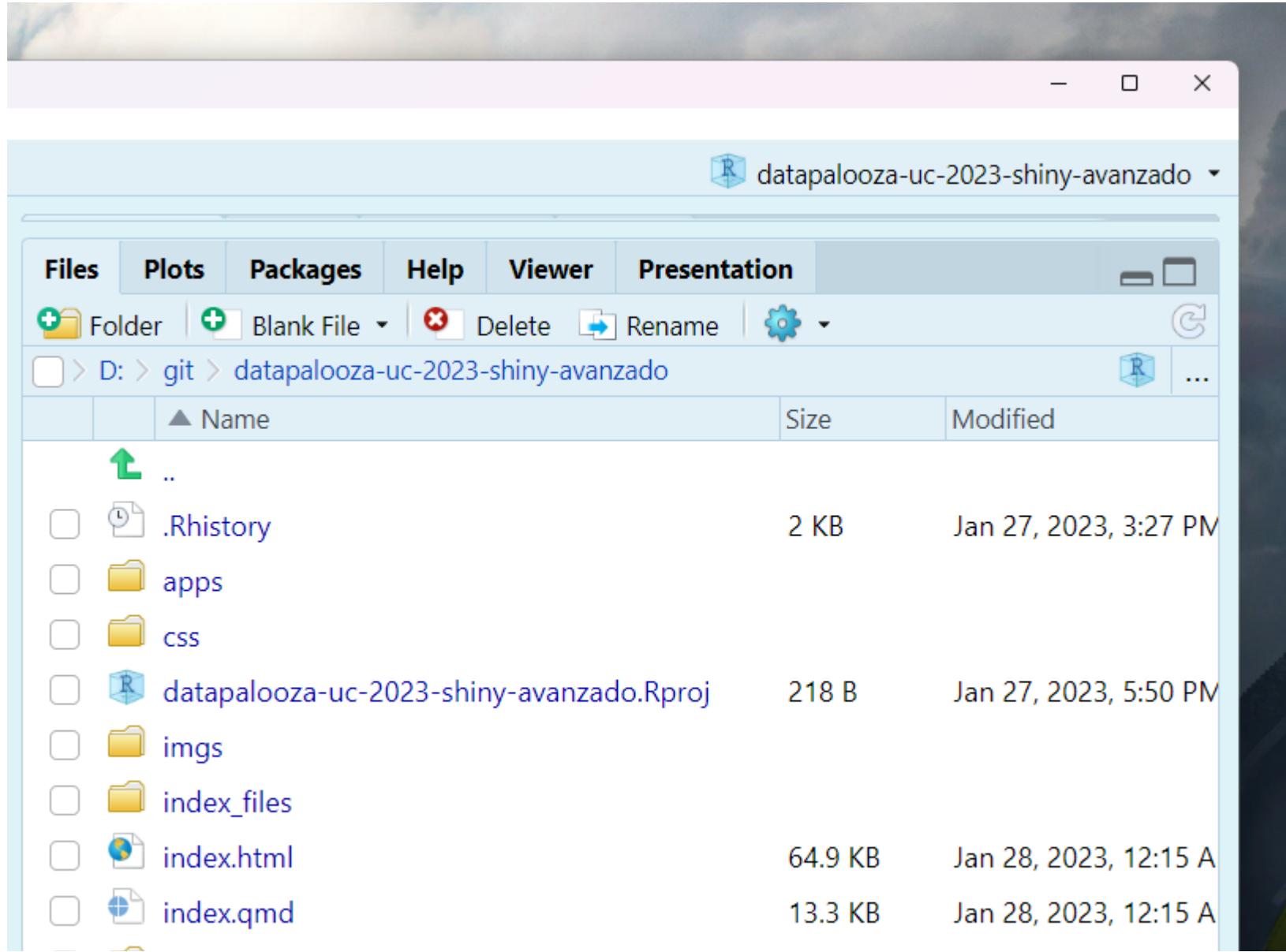
```
1 install.packages(  
2   c("shiny",  
3     "tidyverse",  
4     "rvest",  
5     "leaflet",  
6     "leaflet.minicharts",  
7     "DT",  
8     "highcharter",  
9     "plotly",  
10    "datos")  
11 )
```

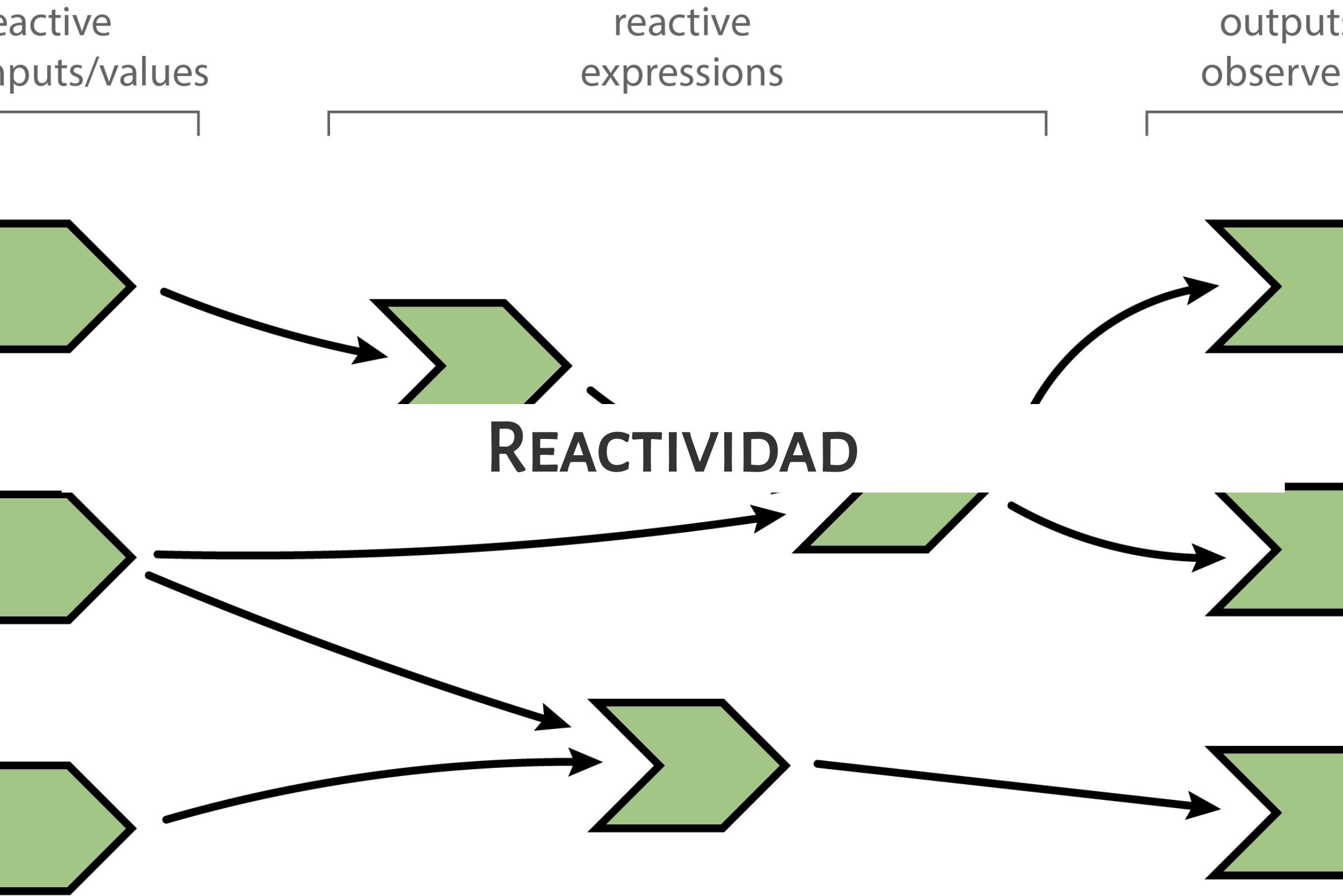
SETUP DE POSITCLOUD (EX RSTUDIO CLOUD)

Cloning a GitHub repo into an rstudio.cloud project



REVISEMOS ANTRES NUESTRO PROYECTO





REACTIVIDAD

En una aplicación shiny, la lógica del *server* se expresa en programación reactiva.

No tenemos que darle un comando a una aplicación shiny para que se actualice. Reaccionará por sí solo.

Si un **input** cambia, automáticamente actualizará **output(s)** (expresiones reactivas u **observers**) que dependan de dicho **input(s)**.

TIPOS DE ELEMENTOS

De manera conceptual una *expresión reactiva* es una expresión que cambiará en el tiempo.

- **reactive** crea una expresión reactiva, la cual cambia si una de los elementos reactivos que depende cambia.
- **eventReactive** igual que el anterior pero se gatilla por el cambio de una variable reactiva.
- **observe** Crear una expresión reactiva que se ejecuta cuando una variable de la cual depende cambia.
- **observeEvent** ídem al anterior pero se gatilla por una expresión reactiva.

SIMILITUDES Y DIFERENCIAS ENTRE REACTIVOS Y OBSERVERS.

Similitudes: Se gatillan por el cambio de alguna variable o expresión reactiva.

Diferencias: Expresiones reactivas retornan valores, los **observer**s no lo hacen.

PATRÓN DE USO

```
1 server <- function(input, output) {  
2  
3   variable_reactiva <- reactive({  
4     # calcular y retornar algo, como una función.  
5     ...  
6   })  
7  
8   variable_reactiva <- reactiveEvent(expresion_reactiva, {  
9     # calcular y retornar algo, si cambia `expresion_reactiva`.  
10    ...  
11  })  
12  
13  observe({  
14    # hacer algo si cambia algo que depende esta expresión.  
15    ...  
16  })  
17  
18  observeEvent(expresion_reactiva, {  
19    # hacer algo si cambia `expresion_reactiva`.  
20  })
```

SHINY 1.6.0

Se agrega una función **bindEvent** el cual convierte **reactives** y **observers** en su versión **Event**:

```
1 server <- function(input, output) {  
2  
3   # Equivalentes!  
4   var_react <- eventReactive(input$boton, { ... })  
5   var_react <- reactive({ ... }) |> bindEvent(input$boton)  
6  
7   # Lo mismo!  
8   observeEvent(input$boton, { ... })  
9   observe({ ... }) |> bindEvent(input$boton)  
10 }
```

RESUMEN: TAREA/función

- Calcular valores: **reactive** / eventReactive
- Ejecutar tareas: **observe** / observeEvent
- Almacenar valores: **reactiveVal** / input / reactiveValues
- Prevenir reactividad: **isolate**
- Chequear condiciones: **req**
- Reejecutar tareas cada cierto tiempo: **invalidateLater**

REVISAR LAS SIGUIENTES APPS

- apps/reactividad/01-app-reactividad-basica.R
- apps/reactividad/02-app-reactive.R
- apps/reactividad/03-app-eventReactive.R
- apps/reactividad/04-app-observe.R
- apps/reactividad/04-app-observeEvent.R

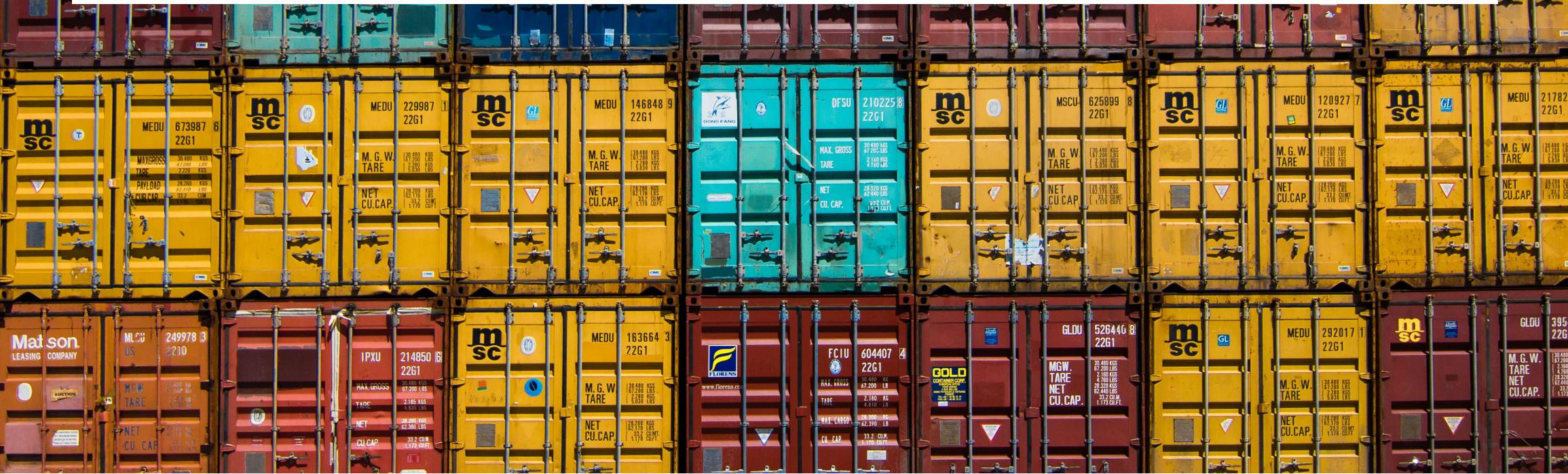
EJERCICIO: PAQUETE `reactlog`

El paquete `reactlog` permite registrar las acciones de una aplicación para conocer el flujo de la reactividad.

1. Revise y ejecute el script `R/01-reactlog-example.R`. Comente.



MÓDULOS



MÓDULOS

- Un módulo es un componente autocontenido y compuesto de una aplicación Shiny.
 - Autocontenido como una función.
 - Se pueden combinar para hacer una aplicación.
- Tienen su propia interfaz de usuario y servidor (además de la interfaz de usuario y servidor de la **aplicación**).
- Útil para la reutilización.
 - En lugar de copiar y pegar código, puede usar módulos. Incorporar en paquetes.
 - Esencial para administrar la complejidad del código en aplicaciones más grandes.

PATRÓN DE USO EN UI: ANTES

```
1 ui <- fluidPage(  
2   fluidRow(  
3     column(2, sliderInput("nrand", ...)),  
4     column(2, selectInput("dist", ...))),  
5     column(8, plotOutput("plot"))  
6   )  
7 )
```

PATRÓN DE USO EN UI: AHORA

```
1 moduloUI <- function(id) {  
2  
3   ns <- NS(id)  
4  
5   tagList(  
6     column(2, sliderInput(ns("rand"), ...)),  
7     column(2, selectInput(ns("dist"), ...))),  
8     column(8, plotOutput(ns("plot"))))  
9   )  
10 }  
11  
12 ui <- fluidPage(  
13   fluidRow(  
14     moduloUI("idmod")  
15   ))  
16 )  
17 )
```

PATRÓN DE USO EN SERVER: ANTES

```
1 server <- function(input, output, session) {  
2  
3     output$plot <- renderPlot({  
4  
5         rdist <- switch(input$dist, "norm" = rnorm, "exp" = rexp)  
6         hist(rdist(input$rand), xlim = c(-5, 5))  
7     } )  
8  
9 }  
10 }
```

PATRÓN DE USO EN SERVER: AHORA

```
1 moduloServidor <- function(id) {  
2   moduleServer(id, function(input, output, session) {  
3  
4     output$plot <- renderPlot({  
5  
6       rdist <- switch(input$dist, "norm" = rnorm, "exp" = rexp)  
7       hist(rdist(input$rand), xlim = c(-5, 5))  
8     })  
9   })  
10 }  
11  
12 }  
13  
14 server <- function(input, output, session) {  
15   moduloServidor("idmod")  
16  
17 } )
```

PATRÓN DE USO

Al cambiar a este flujo/framework de programación de aplicaciones shiny:

Módulo UI: El cambio en `ids` es explícito. Cambiar `unInput("id")` por `unInput(NS("id"))`. Se puede separar para tener distintas ubicaciones (ver ejemplos).

Modulo Server: La funcionalidad es implícita, uno *copia y pega*, y el módulo hace el intercambio correspondiente, `moduleServer` se encarga de todo.

BENEFICIO?

```
1 source("ruta/hacia/modulo.R")
2
3 ui <- fluidPage(
4   fluidRow(
5     moduloUI("modulo1"),
6     moduloUI("modulo2"),
7     ...
8   )
9 )
10
11 server <- function(input, output, session) {
12   moduloServidor("modulo1")
13   moduloServidor("modulo2")
14   ...
15 }
```

REVISAR LAS SIGUIENTES APPS

- apps/modulos/01-app-modulos.R
- apps/modulos/02-app-modulos.R
- apps/modulos/03-app-modulos.R
- apps/modulos/04-app-modulos.R
- apps/modulos/05-app-modulos.R
- apps/modulos/06-app-modulos.R
- apps/modulos/07-app-modulos.R

EJERCICIO: CREANDO APP, LUEGO UN MÓDULO

1. Considerando el data set `iris`, genere una aplicación que tenga un selector que liste las variables continuas y dado una variable realice un histograma.
2. Ahora transfórmelo en módulo y agregue tres instancias de dicho módulo con diferentes tablas (`mtcars`, `diamonds`, `mtautos`, `diamantes`).
3. Extienda el módulo para incluir un output que muestre el `summary` de la variable seleccionada.
4. Ideas? Preguntas?

HTMLWIDGETS



HTMLWIDGETS

Usar librerías de visualización Javascript en R.

Los widgets se pueden incluir en documentos RMarkdown y aplicaciones web Shiny.

HTMLwidgets pueden ser una parte fundamental de una aplicación dado que muestran visualizaciones.

HTMLWIDGETS, ESO ES TODO?

No!

Los widget pueden utilizarse como `inputs` a través de eventos Javascript!
Funcionan si se integran a la aplicación.

Ejemplo 1 <https://jbkunst.shinyapps.io/02-proxy-functions/>.

Ejemplo 2 <https://odes-chile.org/app/estaciones/>.

REVISAR LAS SIGUIENTES APPS

- apps/htmlwidgets/01-app-htmlwidgets.R
- apps/htmlwidgets/02-app-htmlwidgets-loaders.R
- apps/htmlwidgets/03-app-htmlwidgets-proxies.R
- apps/htmlwidgets/04-app-htmlwidgets-events.R

EJERCICIO: HACIENDO MÁS CON LOS EVENTOS

1. Reutilice alguna app de la carpeta [htmlwidgets](#) para que al hacer click sobre un sismo, se escriban todos la información del sismo en la ui.
2. Cree una app que genere un scatter plot de acuerdo a 2 inputs para indicar variable x, variable y. Utilice un proxy para no *rebibujar* el gráfico.
3. Para la aplicación anterior, conviértala en módulo.
4. Ideas? Preguntas?

HTML, CSS, JS

```
<div class="second_content">
  <div class="last">
    <h3>More about the developer
    <img alt="Developer icon" />
  </div>
</div>
```

main.css

style.css

style.css > ...

HTML, CSS, JS

HTML, CSS y JS son tres elementos (lenguajes?) que conforman una aplicación web.

HTML (HyperText Markup Language) permite estructurar la información.

https://www.w3schools.com/html/html_examples.asp

CSS (Cascading Style Sheets) para darle estilo a la información.

https://www.w3schools.com/css/css_examples.asp

JS (JavasCript) para hacer interactividad con esa información.

https://www.w3schools.com/js/js_examples.asp

Aplicaciones shiny lo *abstiene* y hace funcionar todo *mágicamente*.

HTML EN SHINY

```
1 ui <- fluidPage(  
2   actionButton("boton", "Clickéame")  
3 )  
4  
5 ui
```

```
<div class="container-fluid">  
  <button id="boton" type="button" class="btn btn-default action-  
button">Clickéame</button>  
</div>
```

HTML USANDO tags

```
1 ui2 <- tags$div(class = "container-fluid",
2   tags$button("Clickéame", id="boton", type="button", class="btn btn-default")
3 )
4
5 ui2
```

```
<div class="container-fluid">
  <button id="boton" type="button" class="btn btn-default action-
button">Clickéame</button>
</div>
```

HTML EN BRUTO

```
1 ui3 <- HTML('<div class="container-fluid">
2   <button id="boton" type="button" class="btn btn-default action-button">Cl
3 </div>')
4
5 ui3
```

```
<div class="container-fluid">
  <button id="boton" type="button" class="btn btn-default action-
button">Clickéame</button>
</div>
```

EJERCICIO: HTML

Comprobar si `ui`, `ui2`, y `ui3` son iguales o similares.

EJERCICIO: IMPLEMENTANDO COSAS

1. Revise este link y en la primera app de htmlwidgets realice lo necesario para implementar <https://www.jumpingrivers.com/blog/improving-responsiveness-shiny-applications/>. Implemente de diferentes formas dicha *feature*.

ESO! GRACIAS! UN GUSTAZO!

CRÉDITOS IMÁGENES

Foto de [Stephen Dawson](#) en [Unsplash](#)

Foto de [Guillaume Bolduc](#) en [Unsplash](#)

Foto de [Valery Sysoev](#) en [Unsplash](#)

Foto de [Nick Fewings](#) en [Unsplash](#)

Foto de [Patrick Perkins](#) en [Unsplash](#)

