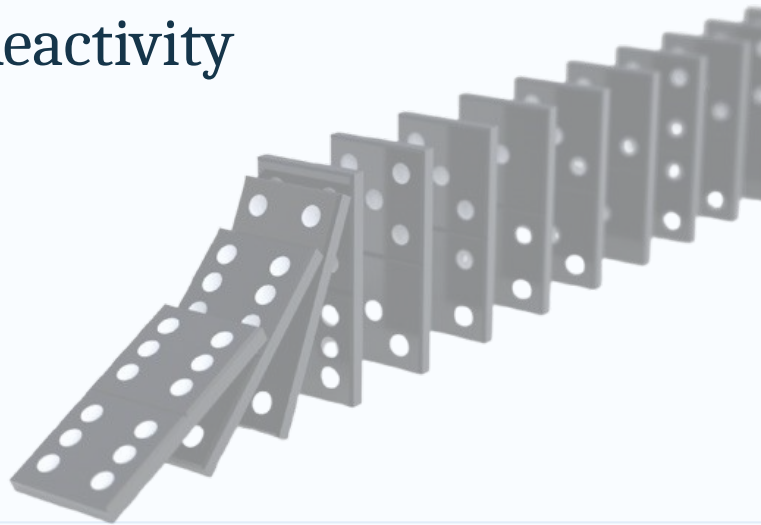
# Why Is It Triggering Twice?

## Understanding Shiny Reactivity



Douglas Mesquita

shiny::conf

09/04/2025

# Overview

# About me

- Master's and PhD in Statistics (UFMG)
- R/Shiny developer for 3 years
- Statistician/Data scientist 10+ years
- Open source contributor


github.com/DouglasMesquita

linkedin.com/in/douglas-mesquita

www.require-r.com

# Wait a Second!

## What should you know before we start?

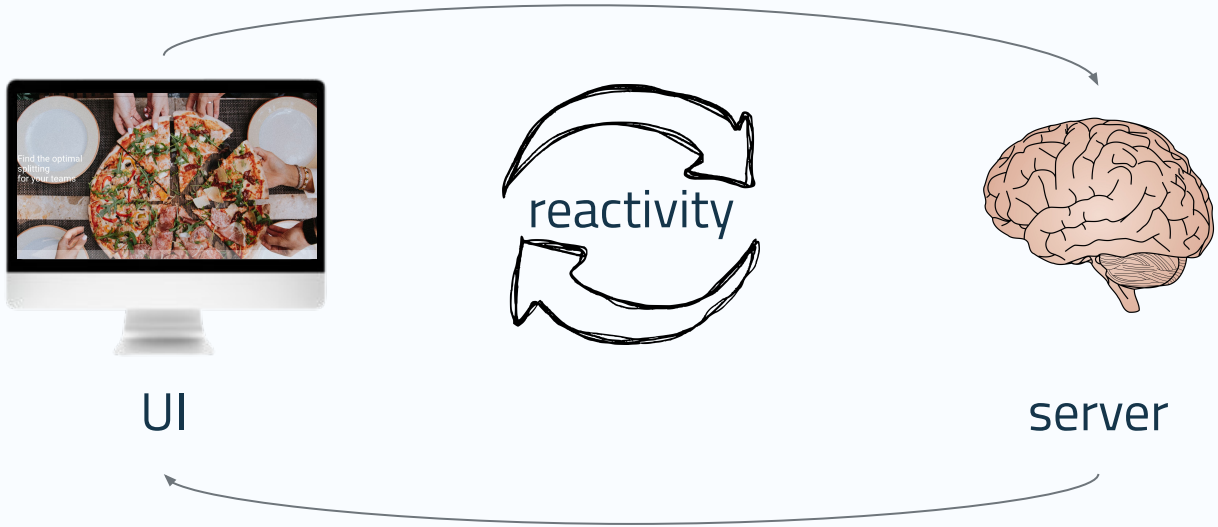
# shiny::conf repo

In this repo you will find all the examples I will be showing today as well as this presentation!

## douglasmesquita/shiny_conf_2025

Open the project and run *renv::restore()*

# Where does reactivity happen?



# Understanding Shiny Reactivity

## Dependencies, invalidations, and execution flow

# Reactivity

a() absolutePanel() actionButton() actionLink() addResourcePath() animationOptions() appendTab() as.shiny.appobj() basicPage() bindCache() bindEvent() bookmarkButton() bootstrapLib() bootstrapPage() br() browserViewer() brushedPoints() brushOpts() busyIndicatorOptions() callModule() captureStackTraces() checkboxGroupInput() checkboxInput() clickOpts() code() column() conditionalPanel() conditionStackTrace() conditionStackTrace<-() createRenderFunction() createWebDependency() dataTableOutput() dateInput() dateRangeInput() dblclickOpts() debounce() devmode() dialogViewer() diskCache() div() downloadButton() downloadHandler() downloadLink() em() enableBookmarking() eventReactive() exportTestValues() exprToFunction() ExtendedTask() fileInput() fillCol() fillPage() fillRow() fixedPage() fixedPanel() fixedRow() flowLayout() fluidPage() fluidRow() freezeReactiveVal() freezeReactiveValue() get_devmode_option() getCurrentOutputInfo() getCurrentTheme() getDefaultReactiveDomain() getQueryString() getShinyOption() getUrlHash() h1() h2() h3() h4() h5() h6() headerPanel() helpText() hideTab() hoverOpts() hr() HTML() htmlOutput() htmlTemplate() httpResponse() icon() imageOutput() img() in_devmode() includeCSS() includeHTML() includeMarkdown() includeScript() includeText() incProgress() inputPanel() insertTab() insertUI() installExprFunction() invalidateLater() is.key_missing() is.reactive() is.reactivevalues() is.shiny.appobj() is.singleton() isolate() isRunning() isTruthy() key_missing() loadSupport() mainPanel() makeReactiveBinding() markdown() markRenderFunction() maskReactiveContext() memoryCache() MockShinySession() modalButton() modalDialog() moduleServer() navbarMenu() navbarPage() navlistPanel() nearPoints() need() NS() ns.sep() numericInput() observe() observeEvent() onBookmark() onBookmarked() onFlush() onFlushed() onReactiveDomainEnded() onRestore() onRestored() onSessionEnded() onStop() onUnhandledError() outputOptions() p() pageWithSidebar() paneViewer() parseQueryString() passwordInput() plotOutput() plotPNG() pre() prependTab() printError() printStackTrace() Progress() quoToFunction() radioButtons() reactive() reactiveConsole() reactiveFileReader() reactivePoll() reactiveTimer() reactiveVal() reactiveValues() reactiveValuesToList() reactlog() reactlogAddMark() reactlogReset() reactlogShow() register_devmode_option() registerInputHandler() registerThemeDependency() removeInputHandler() removeModal() removeNotification() removeResourcePath() removeTab() removeUI() renderCachedPlot() renderDataTable() renderImage() renderPlot() renderPrint() renderTable() renderText() renderUI() repeatable() req() resourcePaths() restoreInput() runApp() runExample() runGadget() runGist() runGitHub() runTests() runUrl() safeError() selectInput() selectizeInput() serverInfo() setBookmarkExclude() setProgress() setSerializer() shinyApp() shinyAppDir() shinyAppFile() shinyAppTemplate() shinyOptions() shinyServer() shinyUI() showBookmarkUrlModal() showModal() showNotification() showTab() sidebarLayout() sidebarPanel() singleton() sizeGrowthRatio() sliderInput() snapshotExclude() snapshotPreprocessInput() snapshotPreprocessOutput() span() splitLayout() stopApp() strong() submitButton() suppressDependencies() tableOutput() tabPanel() tabPanelBody() tabsetPanel() tag() tagAppendAttributes() tagAppendChild() tagAppendChildren() tagGetAttribute() tagHasAttribute() tagList() tags() tagSetChildren() testServer() textAreaInput() textInput() textOutput() throttle() titlePanel() uiOutput() updateActionButton() updateActionLink() updateCheckboxGroupInput() updateCheckboxInput() updateDateInput() updateDateRangeInput() updateNavbarPage() updateNavlistPanel() updateNumericInput() updateQueryString() updateRadioButtons() updateSelectInput() updateSelectizeInput() updateSliderInput() updateTabsetPanel() updateTextAreaInput() updateTextInput() updateVarSelectInput() updateVarSelectizeInput() urlModal() useBusyIndicators() validate() validateCssUnit() varSelectInput() varSelectizeInput() verbatimTextOutput() verticalLayout() wellPanel() with_devmode() withLogErrors() withMathJax() withProgress() withReactiveDomain() withTags()

shiny@1.0.9

# Reactivity

a() absolutePanel() actionButton() actionLink() addResourcePath() animationOptions() appendTab() as.shiny.appobj() basicPage() bindCache() bindEvent() bookmarkButton() bootstrapLib() bootstrapPage() br() browserViewer() brushedPoints() brushOpts() busyIndicatorOptions() callModule() captureStackTraces() checkboxGroupInput() checkboxInput() clickOpts() code() column() conditionalPanel() conditionStackTrace() conditionStackTrace<-() createRenderFunction() createWebDependency() dataTableOutput() dateInput() dateRangeInput() dblclickOpts() debounce() devmode() dialogViewer() diskCache() div() downloadButton() downloadHandler() downloadLink() em() enableBookmarking() eventReactive() exportTestValues() exprToFunction() ExtendedTask() fileInput() fillCol() fillPage() fillRow() fixedPage() fixedPanel() fixedRow() flowLayout() fluidPage() fluidRow() freezeReactiveVal() freezeReactiveValue() get_devmode_option() getCurrentOutputInfo() getCurrentTheme() getDefaultReactiveDomain() getQueryString() getShinyOption() getUrlHash() h1() h2() h3() h4() h5() h6() headerPanel() helpText() hideTab() hoverOpts() hr() HTML() htmlOutput() htmlTemplate() httpResponse() icon() imageOutput() img() in_devmode() includeCSS() includeHTML() includeMarkdown() includeScript() includeText() incProgress() inputPanel() insertTab() insertUI() installExprFunction() invalidateLater() is.key_missing() is.reactive() is.reactivevalues() is.shiny.appobj() is.singleton() isolate() isRunning() isTruthy() key_missing() loadSupport() mainPanel() makeReactiveBinding() markdown() markRenderFunction() maskReactiveContext() memoryCache() MockShinySession() modalButton() modalDialog() moduleServer() navbarMenu() navbarPage() navlistPanel() nearPoints() need() NS() ns.sep() numericInput() observe() observeEvent() onBookmark() onBookmarked() onFlush() onFlushed() onReactiveDomainEnded() onRestore() onRestored() onSessionEnded() onStop() onUnhandledError() outputOptions() p() pageWithSidebar() paneViewer() parseQueryString() passwordInput() plotOutput() plotPNG() pre() prependTab() printError() printStackTrace() Progress() quoToFunction() radioButtons() reactive() reactiveConsole() reactiveFileReader() reactivePoll() reactiveTimer() reactiveVal() reactiveValues() reactiveValuesToList() reactlog() reactlogAddMark() reactlogReset() reactlogShow() register_devmode_option() registerInputHandler() registerThemeDependency() removeInputHandler() removeModal() removeNotification() removeResourcePath() removeTab() removeUI() renderCachedPlot() renderDataTable() renderImage() renderPlot() renderPrint() renderTable() renderText() renderUI() repeatable() req() resourcePaths() restoreInput() runApp() runExample() runGadget() runGist() runGitHub() runTests() runUrl() safeError() selectInput() selectizeInput() serverInfo() setBookmarkExclude() setProgress() setSerializer() shinyApp() shinyAppDir() shinyAppFile() shinyAppTemplate() shinyOptions() shinyServer() shinyUI() showBookmarkUrlModal() showModal() showNotification() showTab() sidebarLayout() sidebarPanel() singleton() sizeGrowthRatio() sliderInput() snapshotExclude() snapshotPreprocessInput() snapshotPreprocessOutput() span() splitLayout() stopApp() strong() submitButton() suppressDependencies() tableOutput() tabPanel() tabPanelBody() tabsetPanel() tag() tagAppendAttributes() tagAppendChild() tagAppendChildren() tagGetAttribute() tagHasAttribute() tagList() tags() tagSetChildren() testServer() textAreaInput() textInput() textOutput() throttle() titlePanel() uiOutput() updateActionButton() updateActionLink() updateCheckboxGroupInput() updateCheckboxInput() updateDateInput() updateDateRangeInput() updateNavbarPage() updateNavlistPanel() updateNumericInput() updateQueryString() updateRadioButtons() updateSelectInput() updateSelectizeInput() updateSliderInput() updateTabsetPanel() updateTextAreaInput() updateTextInput() updateVarSelectInput() updateVarSelectizeInput() urlModal() useBusyIndicators() validate() validateCssUnit() varSelectInput() varSelectizeInput() verbatimTextOutput() verticalLayout() wellPanel() with_devmode() withLogErrors() withMathJax() withProgress() withReactiveDomain() withTags()

shiny@1.0.9

# Reactivity

actionButton() actionLink()

bindCache() bindEvent()

checkboxGroupInput() checkboxInput()

dataTableOutput() dateInput() dateRangeInput() debounce()

downloadButton() downloadHandler() downloadLink()

eventReactive() fileInput()

htmlOutput()

imageOutput()

invalidateLater()

isolate()

numericInput() observe() observeEvent()

passwordInput() plotOutput()

radioButtons() reactive()

reactiveFileReader() reactivePoll() reactiveTimer() reactiveVal() reactiveValues()

renderCachedPlot() renderDataTable() renderImage() renderPlot() renderPrint() renderTable() renderText() renderUI()

req()

selectInput() selectizeInput()

tableOutput()

textAreaInput() textInput() textOutput() throttle()

uiOutput()

validate()

varSelectInput() varSelectizeInput() verbatimTextOutput()

shiny@1.0.9

# Reactivity

actionButton()
actionLink()
checkboxGroupInput()
checkboxInput()
dateInput()
dateRangeInput()
downloadButton()
downloadLink()
fileInput()
numericInput()
passwordInput()
radioButtons()
selectInput()
selectizeInput()
textAreaInput()
textInput()
varSelectizeInput()
reactiveVal()
reactiveValues()

renderCachedPlot()
renderDataTable()
renderImage()
renderPlot()
renderPrint()
renderTable()
renderText()
renderUI()

bindCache()
bindEvent()
debounce()
invalidateLater()
isolate()
throttle()

eventReactive()
reactive()
reactiveFileReader()
reactivePoll()
reactiveTimer()

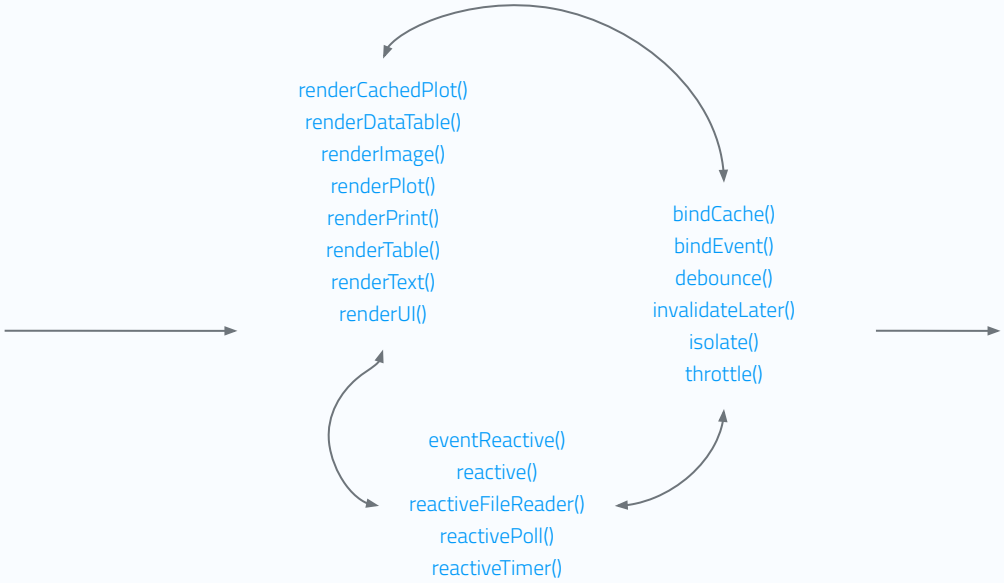
dataTableOutput()
downloadHandler()
htmlOutput()
imageOutput()
plotOutput()
tableOutput()
textOutput()
uiOutput()
verbatimTextOutput()

req()
validate()

observe()
observeEvent()

# Reactivity

actionButton()
actionLink()
checkboxGroupInput()
checkboxInput()
dateInput()
dateRangeInput()
downloadButton()
downloadLink()
fileInput()
numericInput()
passwordInput()
radioButtons()
selectInput()
selectizeInput()
textAreaInput()
textInput()
varSelectizeInput()
reactiveVal()
reactiveValues()

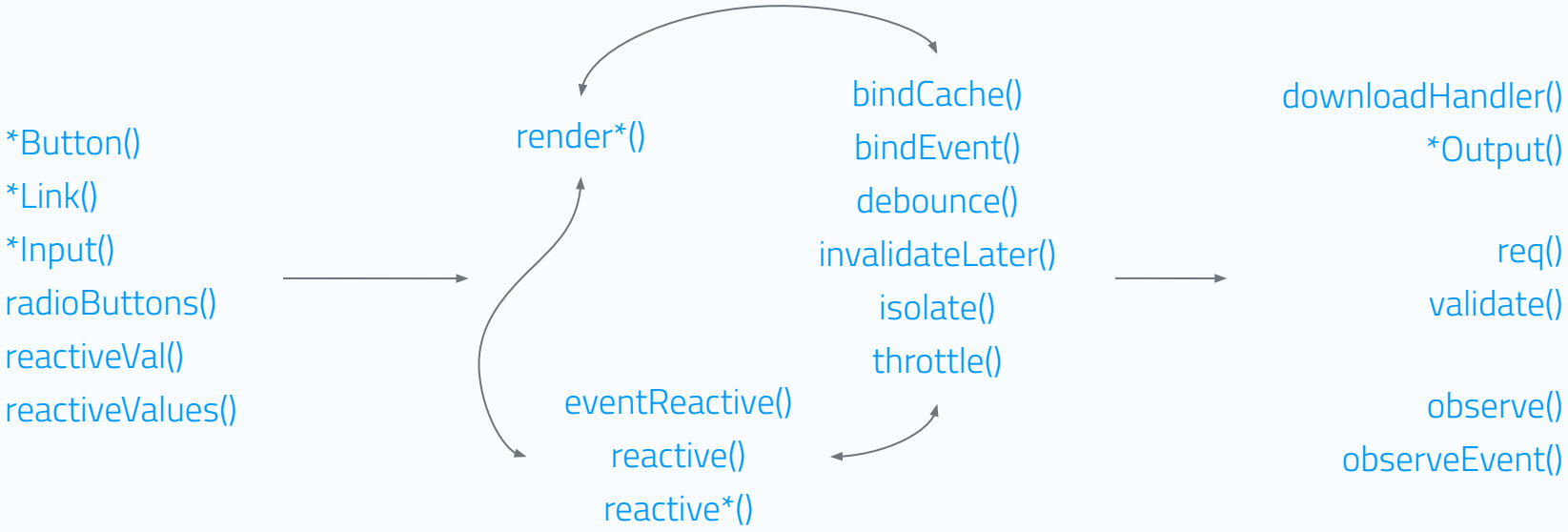



dataTableOutput()
downloadHandler()
htmlOutput()
imageOutput()
plotOutput()
tableOutput()
textOutput()
uiOutput()
verbatimTextOutput()

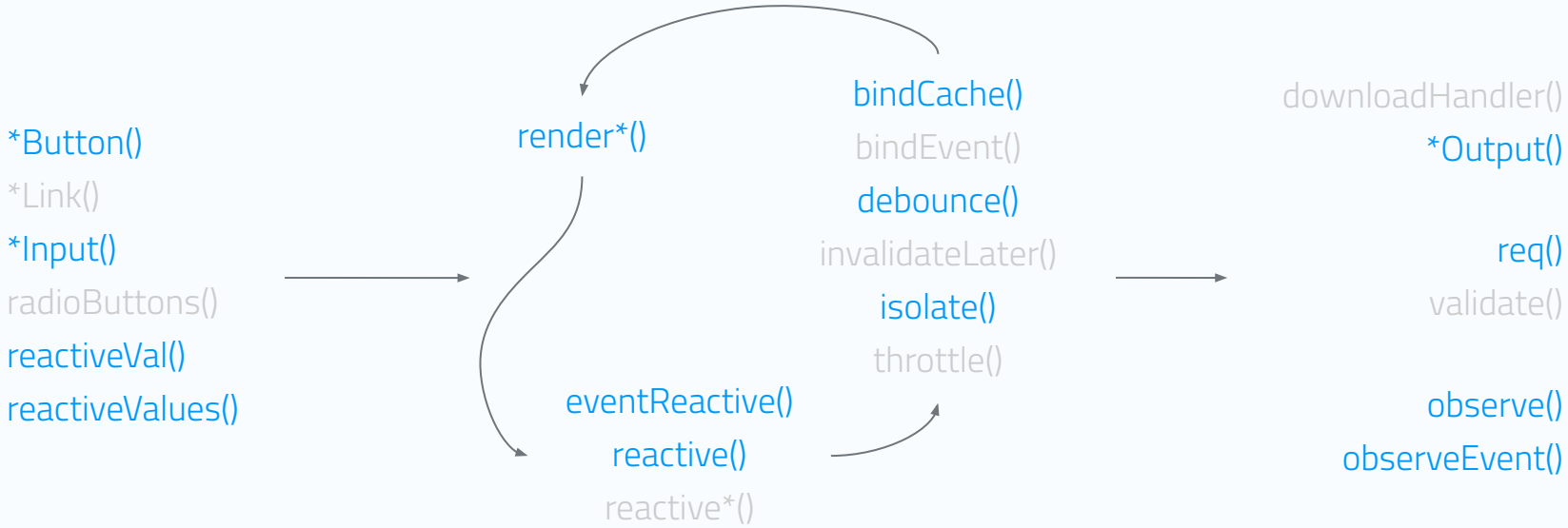
req()
validate()

observe()
observeEvent()

# Reactivity

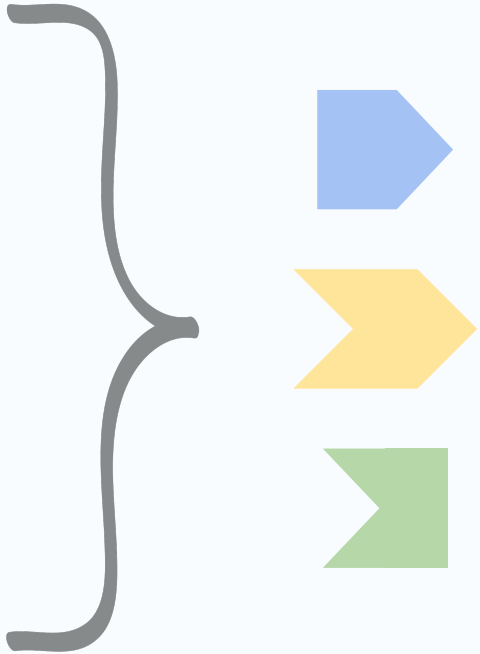

*Button()
*Link()
*Input()
radioButtons()
reactiveVal()
reactiveValues()
render*()
bindCache()
bindEvent()
debounce()
invalidateLater()
isolate()
throttle()
eventReactive()
reactive()
reactive*()
downloadHandler()
*Output()
req()
validate()
observe()
observeEvent()

# Reactivity


*Button()
*Link()
*Input()
radioButtons()
reactiveVal()
reactiveValues()
render*()
bindCache()
bindEvent()
debounce()
invalidateLater()
isolate()
throttle()
eventReactive()
reactive()
reactive*()
downloadHandler()
*Output()
req()
validate()
observe()
observeEvent()

# Reactivity

*Button
*Input()
render*()
*Output()
eventReactive()
reactive()
reactiveVal()
reactiveValues()
bindCache()
debounce()
isolate()
req()
observe()
observeEvent()



# Reactivity

*Button()
*Input()
reactiveVal()
reactiveValues()



render*()
eventReactive()
reactive()

---

bindCache()
debounce()
isolate()

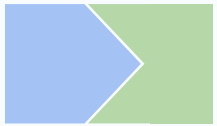


*Output()
observe()
observeEvent()

---

req()

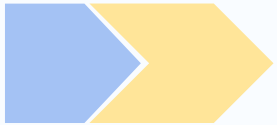
# Reactivity



Sources can affect Endpoints directly



Conductors can affect Endpoints directly

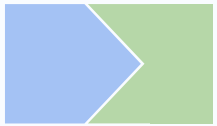


Sources can affect Conductors



Conductors can affect other Conductors

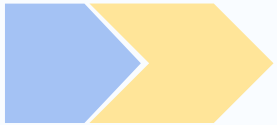
# Reactivity



Sources can affect Endpoints directly



Conductors can affect Endpoints directly



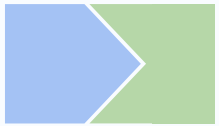
Sources can affect Conductors



Conductors can affect other Conductors

You can connect as many Reactive Components as you want

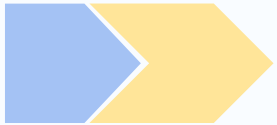
# Reactivity



Sources can affect Endpoints directly



Conductors can affect Endpoints directly

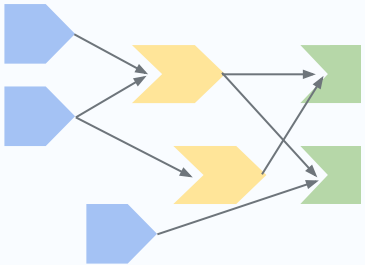


Sources can affect Conductors



Conductors can affect other Conductors

You can connect as many Reactive Components as you want



Reactive components can have many-to-many connections

# Client meeting


We would like a very simple application...

It will take age, weight, gender, activity, and activity duration as inputs and return the estimated number of calories. No worries - we will provide the formula! Also, we don't need anything fancy. Plain text is just fine!

# Calorie burn

$$\text{Calories} = \frac{\text{Duration (min)} * (\text{MET} * \text{Age factor} * \text{Gender factor}) * \text{Weight (kg)} * 3.5}{200}$$

$$\text{Gender factor} = \begin{cases} \text{if men then } 1.1 \\ \text{if women then } 0.9 \end{cases}$$

$$\text{Age factor} = \begin{cases} \text{if age} \in [18, 30) \text{ then } 1.00 \\ \text{if age} \in [30, 40) \text{ then } 0.98 \\ \text{if age} \in [40, 50) \text{ then } 0.96 \\ \text{if age} \in [50, 60) \text{ then } 0.94 \\ \text{if age} \in [60, 70) \text{ then } 0.92 \\ \text{if age} \geq 70 \text{ then } 0.90 \end{cases}$$

$$\text{MET} = \begin{cases} \text{if Volleyball then } 3.0 \\ \text{if Walking then } 3.8 \\ \text{if Cycling then } 6.8 \\ \text{if Football then } 7.0 \\ \text{if Tennis then } 7.3 \\ \text{if Running then } 8.0 \\ \text{if Swimming then } 9.8 \end{cases}$$

The values used are for illustrative purposes only

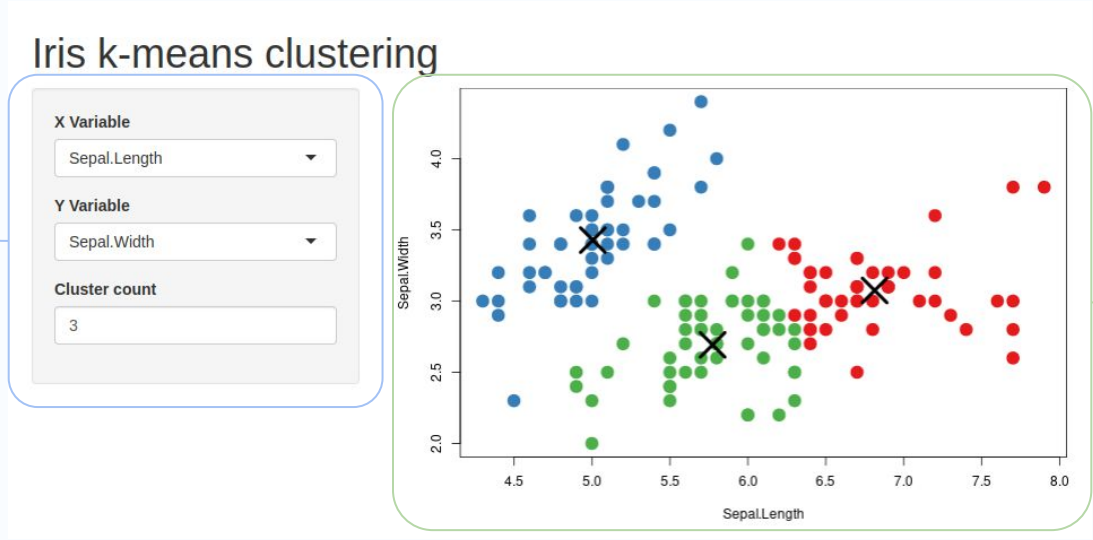
# Core Reactive Elements

## An overview of reactive values,observers, and render functions in Shiny.

# Inputs and Outputs

*Input()

*Output()



# reactive

*reactive()* is a worker that performs its task every time it is requested!

It is updated every time **any** reactive dependencies are triggered

It does not start any task unless requested

It does not have to complete the entire task on its own

It can deliver unfinished work if requested

# reactive

*eventReactive()* only updates when a specific event occurs!

It is updated every time **specific** reactive dependencies are triggered

It does not start any task unless requested

It does not have to complete the entire task on its own

It can deliver unfinished work if requested

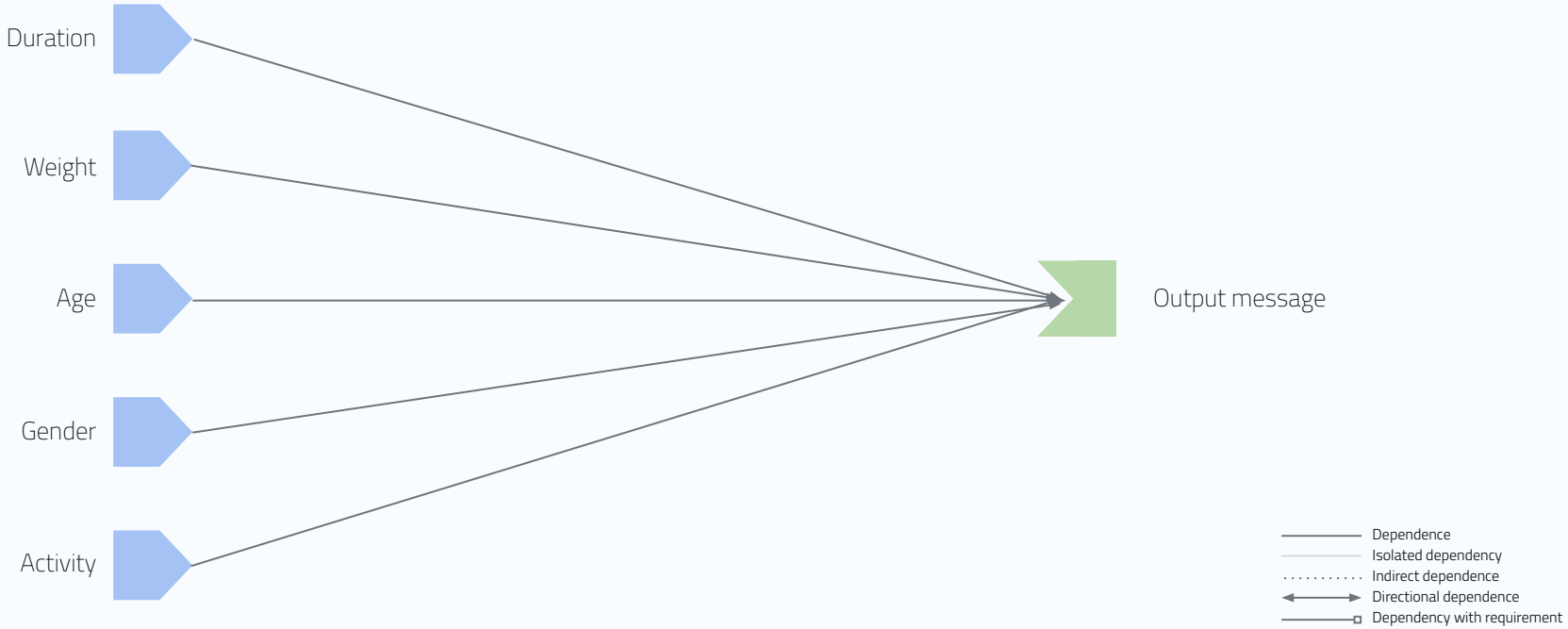
# reactive

## You can create a production line!

Each reactive/eventReactive perform a very specific task

They interact with each other sequentially

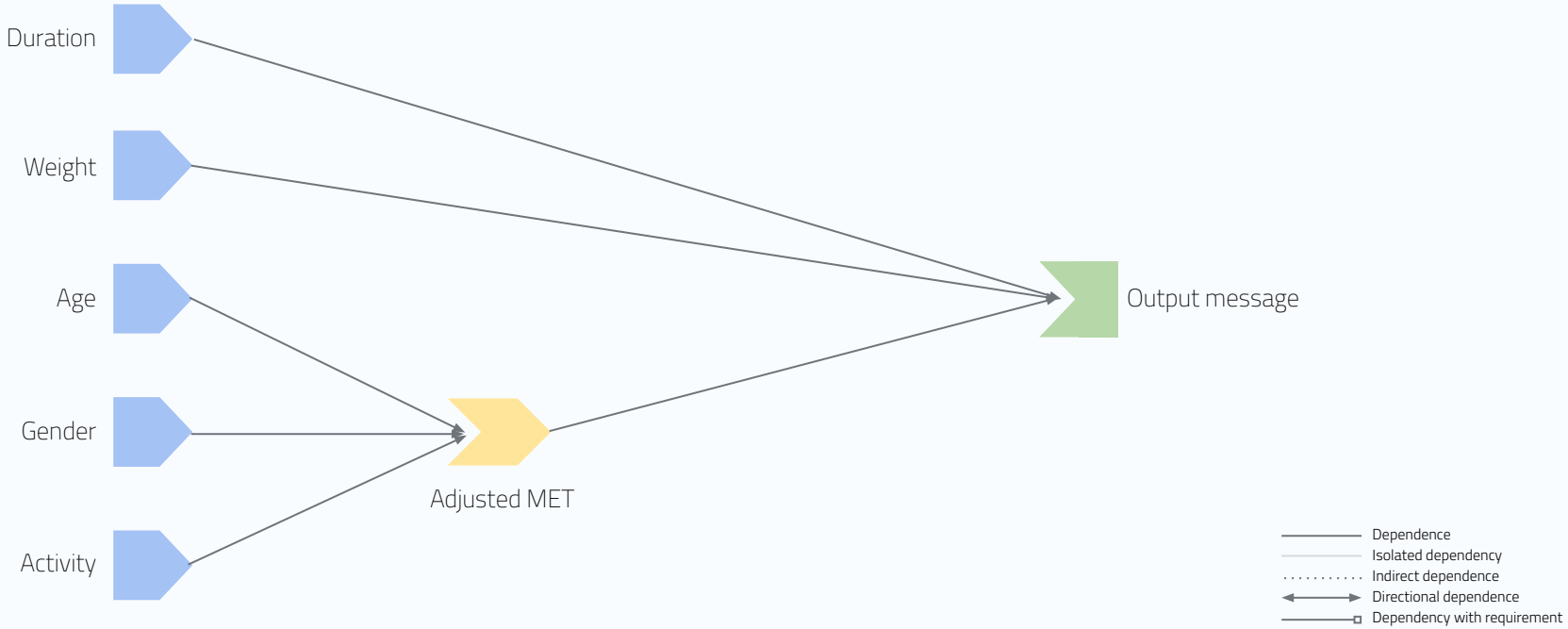
# Calorie burn - 1st version



# Calorie burn

examples/01/server.R & examples/01/ui.R

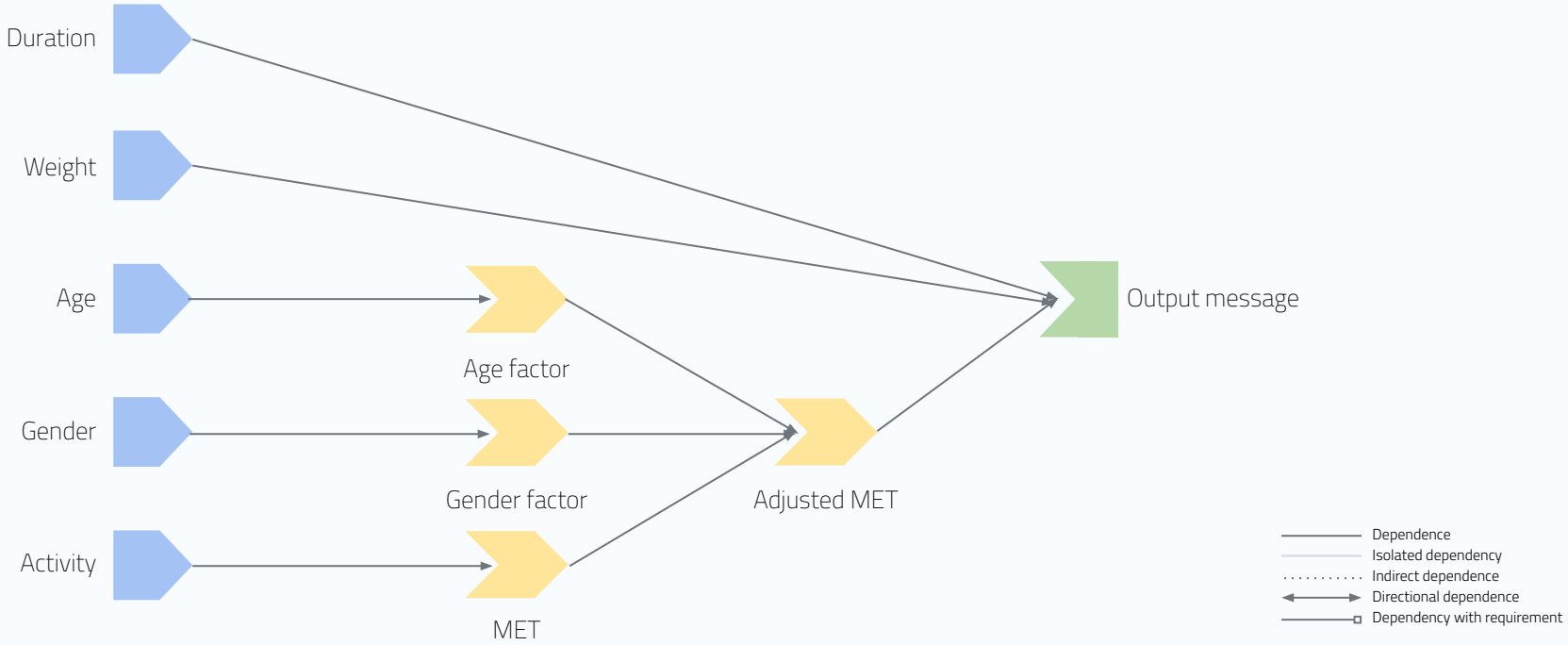
# Calorie burn - 2nd version



# Calorie burn

examples/02/server.R & examples/02/ui.R

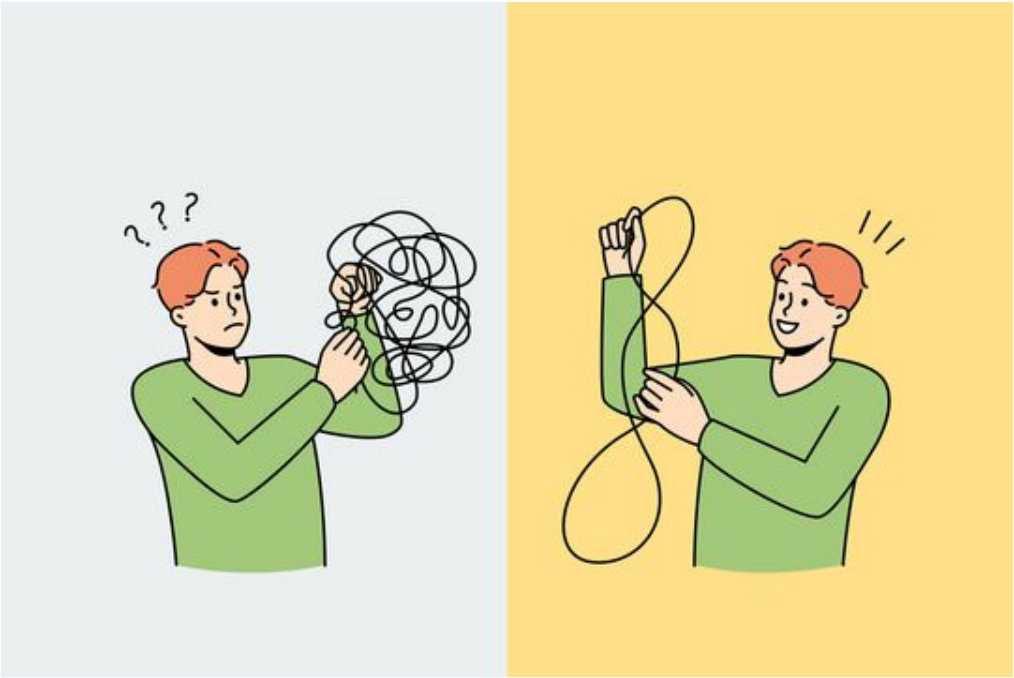
# Calorie burn - 3rd version



# Calorie burn

examples/03/server.R & examples/03/ui.R

# Calorie burn - In practice



# Why was that triggering twice?

A simpler case: we didn't carefully consider the app flow!

## Tips:

- Try to visualize the flow - f the app is large, focus only on the feature you're working on
- Try to create meaningful reactives with a single, well-defined purpose
- Be aware about the elements that are triggering your reactives

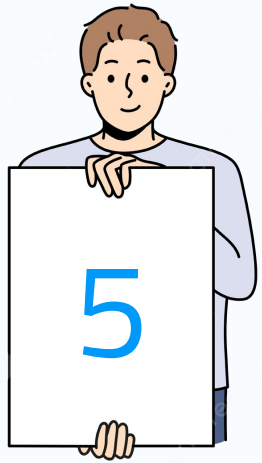
# Client meeting


It's working very well!

Do you see any other areas where we could further improve performance? Some users are a bit frustrated.

# reactiveVal

*reactiveVal()* is a holder for a variable that you can update manually

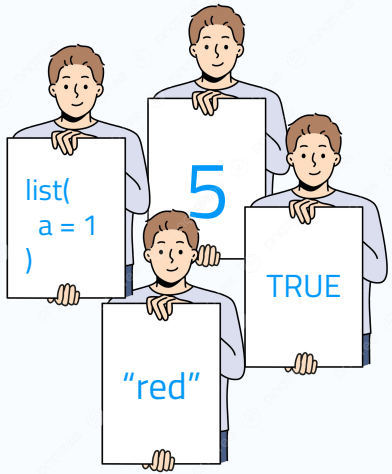



When the value is updated, it will trigger a new reactive chain

Only triggers reactive dependencies if the stored value changes

# reactiveValues

*reactiveValues()* is a set of holders that you can update manually




You can observe elements independently

When one of the value is updated, it will trigger a new reactive chain

Only triggers reactive dependencies if the stored value changes

# observe

*observe()* performs side effects when its reactive dependencies change

Runs when **any** of its reactive dependencies change

Used for side effects (e.g., updating variables, printing logs, ...)

Does not return a value

# observeEvent

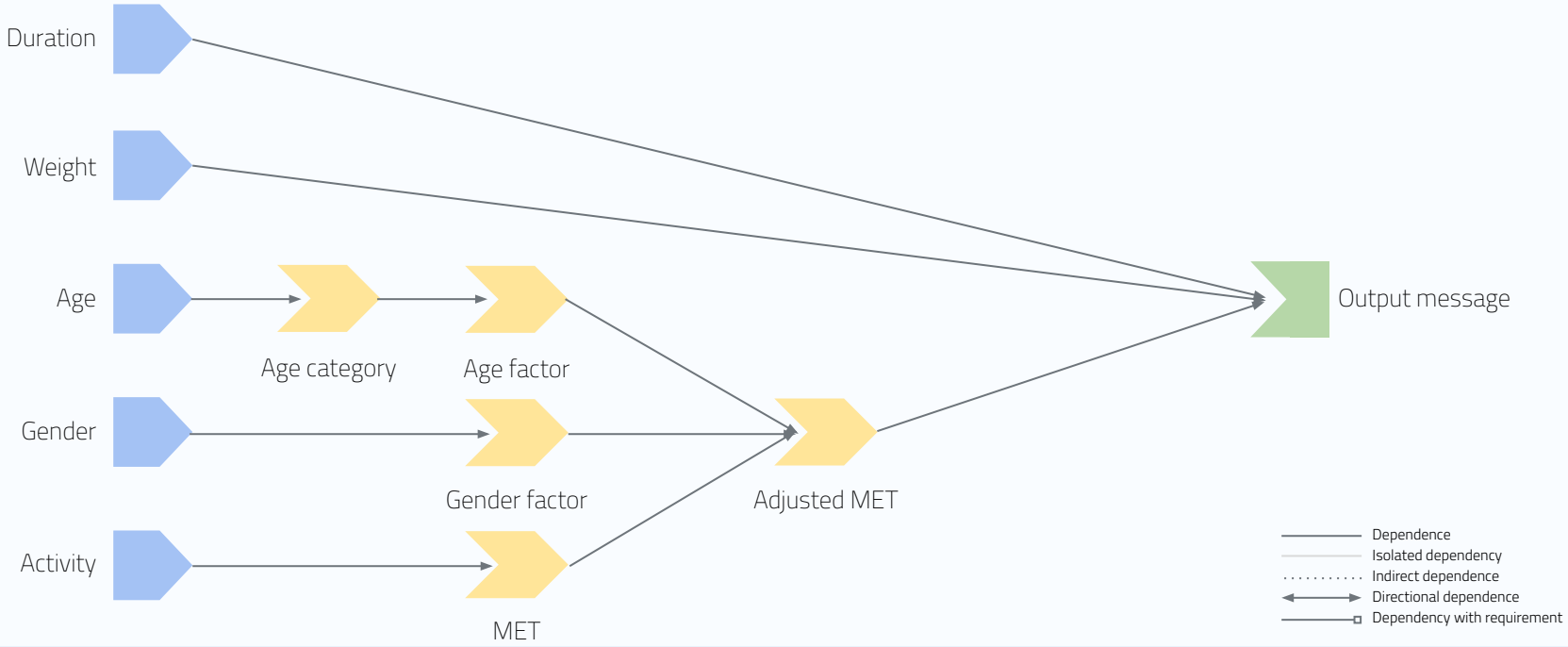
*observeEvent()* performs side effects only when specific event occurs

Runs when **specific** reactive dependencies change

Used for side effects (e.g., updating variables, printing logs, ...)

Does not return a value

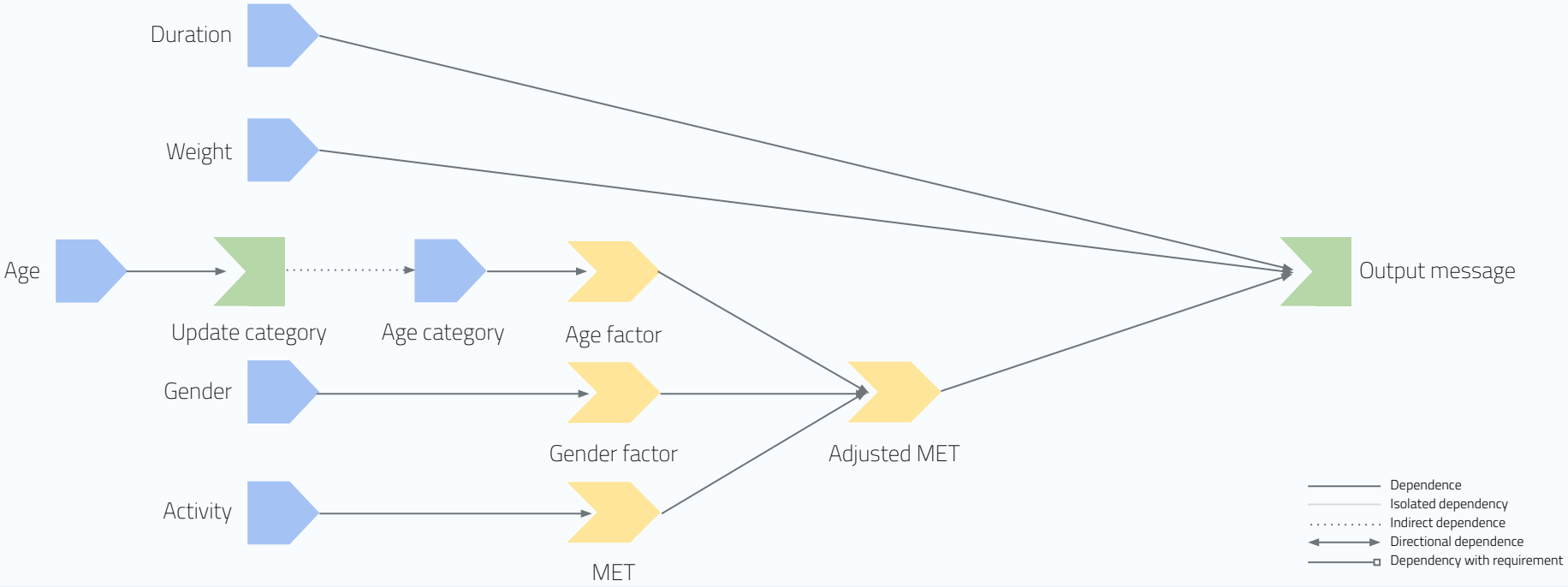
# Calorie burn - 4th version



# Calorie burn

examples/04/server.R & examples/04/ui.R

# Calorie burn - 5th version



# Calorie burn

examples/05/server.R & examples/05/ui.R

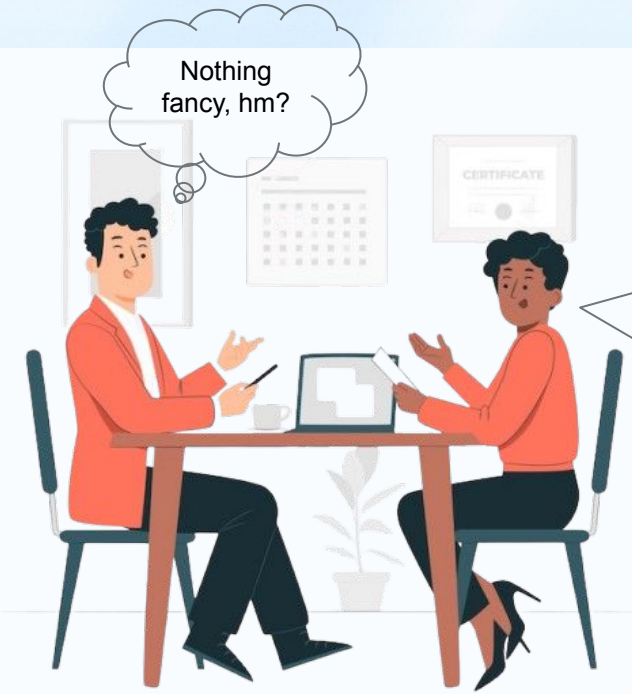
# Why was that triggering twice?

Even though the output doesn't change, a *reactive* expression will always be invalidated, whereas *reactiveVal* will not

Tips:

- Try to identify reactives with a limited number of outputs and replace them with *reactiveValues*
- Prefer using *observeEvent* and *eventReactive* for better control over reactive dependencies

# Client meeting




That's amazing!

We really love it and are using it more and more. We did receive some feedback from our users, though - they would like a way to store their choices for weight, age, gender, activity, and duration since these values don't change often.

# Unexpected Triggers

## How to Prevent Unnecessary Reactivity Updates

# isolate

*isolate()* prevents certain reactive elements from triggering an expression

*isolate()* can be used inside observers and reactives to prevent an expression from being invalidated by reactive changes

# observeEvent arguments

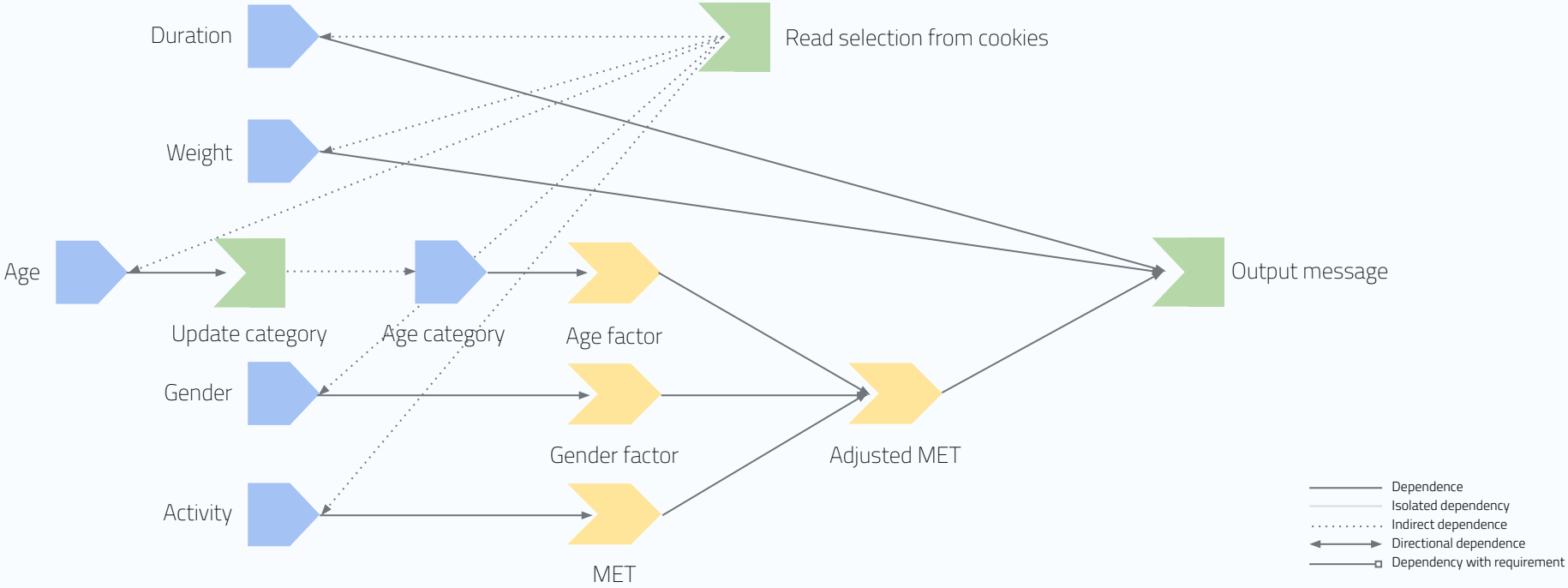
## *once*, *ignoreNULL* and *ignoreInit* are useful friends

*once*: Ensures that the expression is only evaluated once

*ignoreNULL*: Controls whether NULL values should be ignored or not

*ignoreInit*: Prevents the initial value of a reactive from triggering

# Calorie burn - 6th version



# Calorie burn

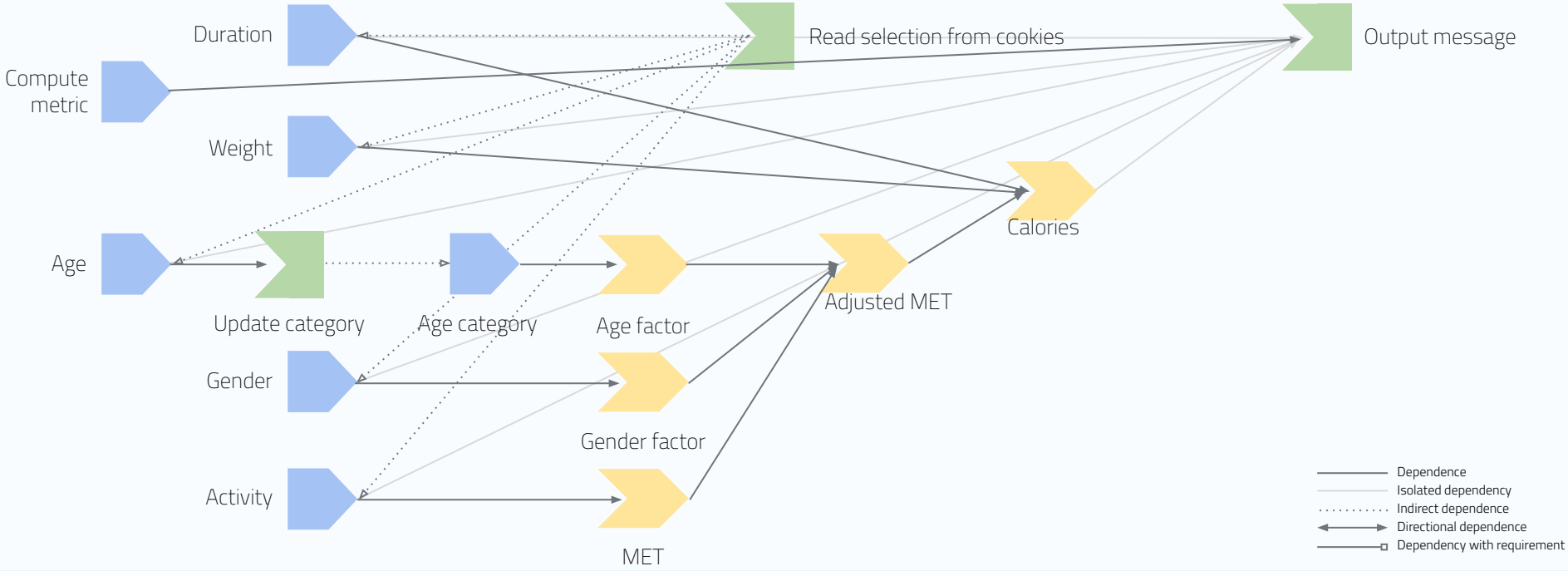
examples/06/server.R & examples/06/ui.R

# Client meeting


By the way...

Could we improve the output message? We'd like to have all the information used to generate the result presented in a single sentence.

# Calorie burn - 7th version



# Calorie burn

examples/07/server.R & examples/07/ui.R

# Why was that triggering twice?

We need to carefully manage the triggers of each reactive context and be aware that, unless we use specific options, observers will always be triggered.

Tips:

- Use *once* when you need to ensure an observer or reactive expression executes only once
- Use *ignoreInit* to prevent execution on initialization, ensuring the first value is ignored
- Use *ignoreNULL = FALSE* when you need to trigger a reaction even when the input is NULL
- Use *isolate()* to access reactive values without establishing a reactive dependency

# Client meeting


We are getting there!

This new button is nice, but we'd prefer a more reactive experience like before. Ideally, the calculations should update automatically as soon as the user changes the inputs.

# Controlling Execution

## Managing Dependencies and Avoiding Unwanted Triggers

# req

*req()* stops reactive execution when specified conditions are not met.

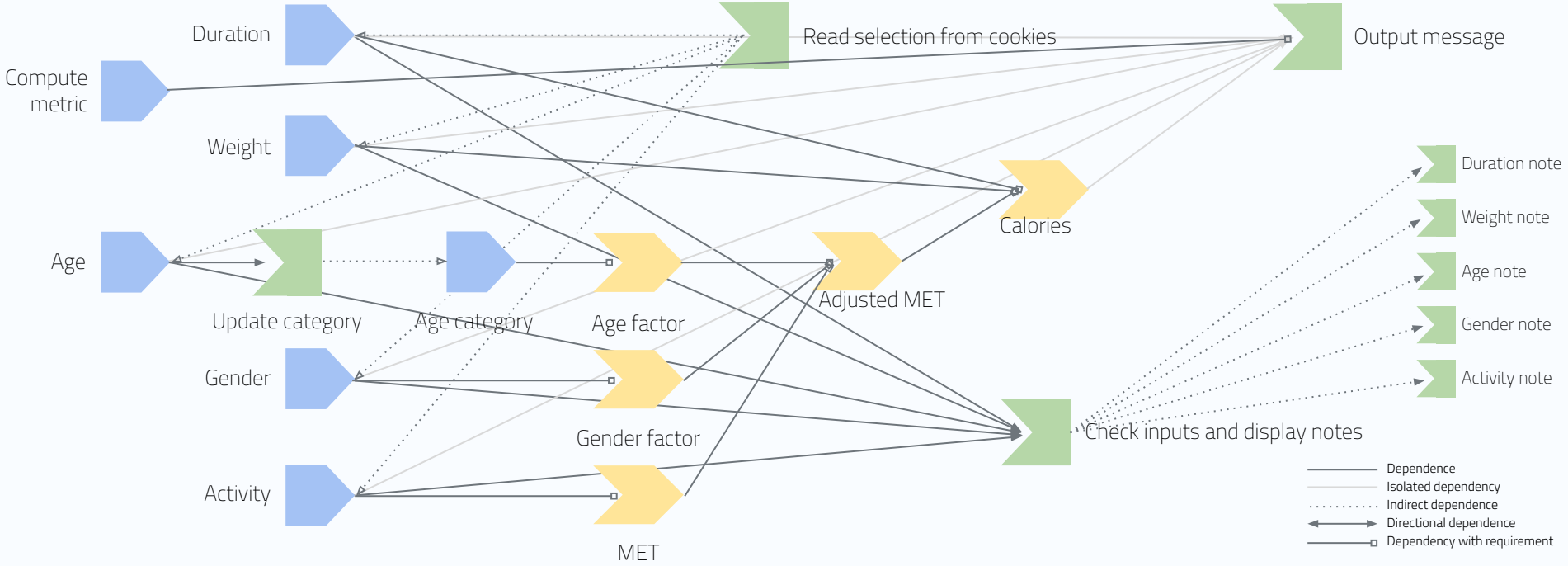

*req()* can be used inside observers and reactive expressions to prevent execution when certain conditions are not met.

# debounce

*debounce()* postpones the execution of a reactive context


You can debounce any reactive expression by specifying the duration to wait for new changes before executing the reactive context

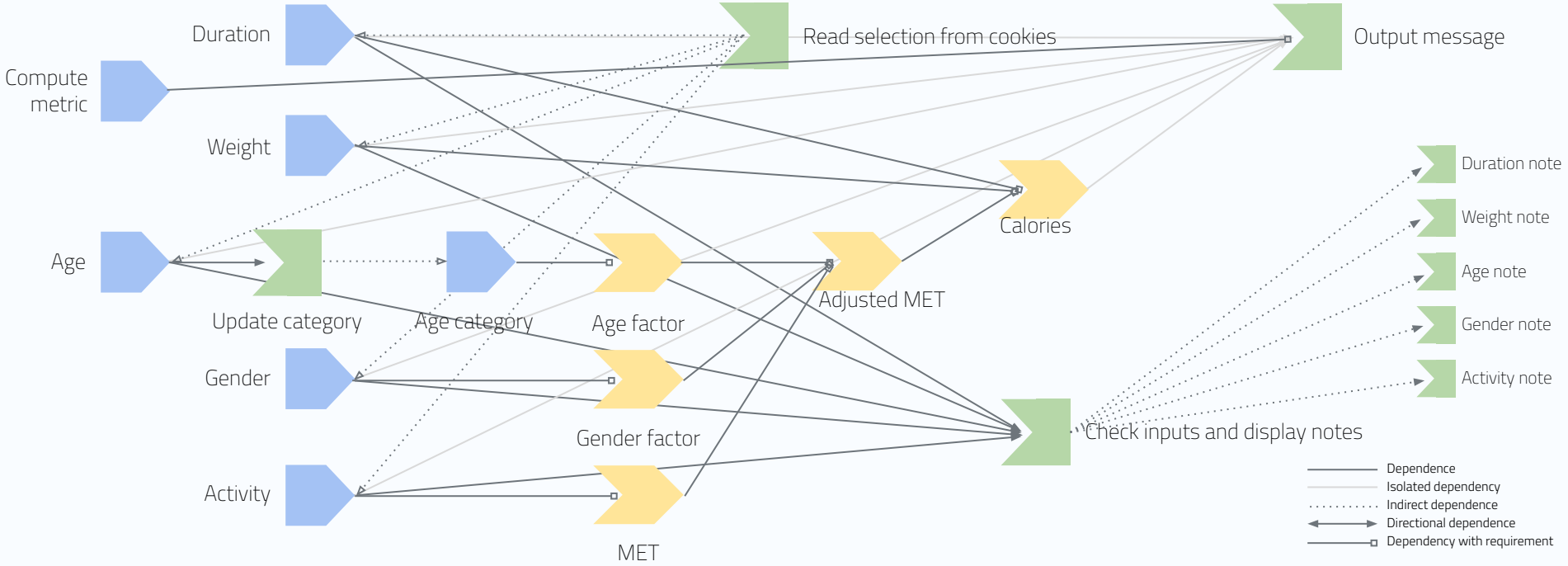
# Calorie burn - 8th version



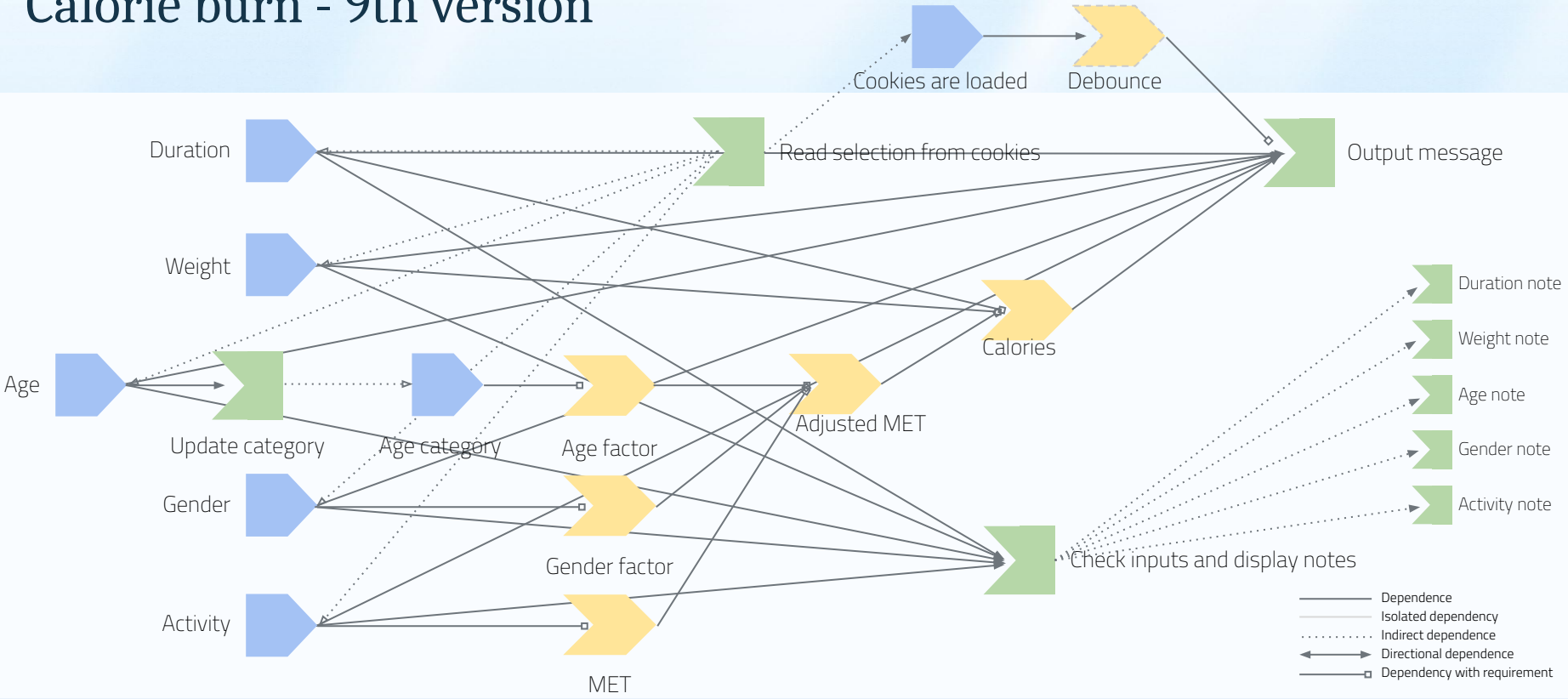
# Calorie burn

examples/08/server.R & examples/08/ui.R

# Calorie burn - 8th version (to compare)



# Calorie burn - 9th version



# Calorie burn

examples/09/server.R & examples/09/ui.R

# Why was that triggering twice?

Inputs in Shiny are updated after the current reactive chain completes, ensuring that UI changes are applied in a synchronized and consistent manner.

Tips:

- Pay attention to where you are using reactive elements (e.g., functions)
- Use meaningful initial values for your inputs
- Use *req()* to prevent premature reactivity
- Use *debounce()* to wait for the reactive chain to settle (or for the user to finish selecting values)

# Optimizing Reactivity

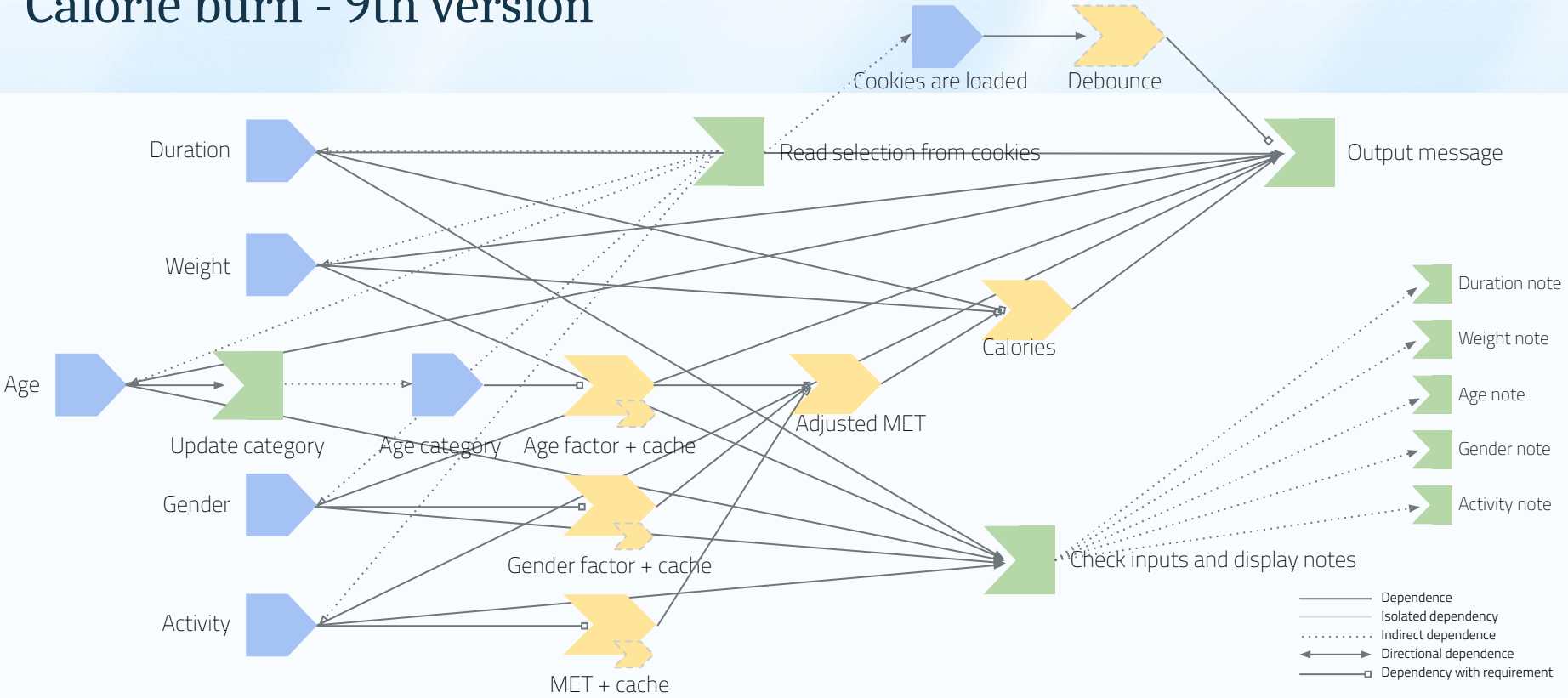
## Improving Performance and Reducing Unwanted Updates

# bindCache

*bindCache()* helps store previously computed results

It avoids redundant recalculations and improve app performance.

# Calorie burn - 9th version



# Calorie burn

examples/10/server.R & examples/10/ui.R

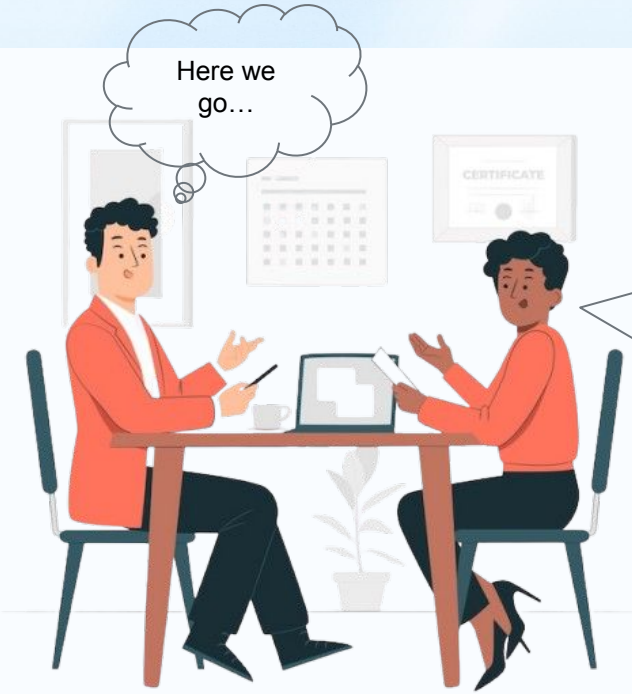
# Why was that triggering twice?

Reactives don’t remember previous values on their own; they only react to changes in input.

Tips:

- Use bindCache() when the results in a reactive don’t change often

# Client meeting




THANK YOU!

The app is performing better than ever. You did a great job! However, the UI doesn't look very good at all…

# Best Practices - Summary

## Key Tips for Improving Shiny App Performance

# Why was that triggering twice?

- Try to visualize the flow and create meaningful reactives with a clear, single purpose
- Be aware that *reactive()* and *reactiveVal()* behave differently
- Prefer using *eventReactive()* and *observeEvent()* when appropriate
- Use *once*, *ignoreInit*, and *ignoreNULL* to properly control the app flow
- Use *isolate()* to access a reactive object without adding a dependency
- Be mindful that using reactive elements inside functions can lead to unintended reactivity
- Use *req()* to prevent unnecessary reactivity
- Use *debounce()* to delay reactivity
- Use *bindCache()* to improve performance when the results of a reactive don't change frequently

# Thank you :)

Douglas Mesquita

shiny::conf

09/04/2025