```
server <- function(input, output, session) {

  downloads <- metaReactive({
    cranlogs::cran_downloads(
      ..(input$package),
      from = ..(format(Sys.Date() - 365)),
      to = Sys.Date()
    )
  })

  downloads_rolling <- metaReactive2({
    validate(need(sum(downloads()$count) > 0, "Input a valid package name"))

    metaExpr({
      ..(downloads()) %>%
        mutate(count = zoo::rollapply(count, 7, mean, fill = "extend"))
    })
  })

  output$plot <- metaRender(renderPlot, {
    ggplot(..(downloads_rolling()), aes(date, count)) + geom_line()
  })
}
```

Step 2: Mark reactive reads

Pro tip: use ..() to return the *value of* an expression

```
metaRender(renderPlot,
```

```
metaReactive2({
```

downloads_rolling

function(input,

```
zoo::rollapply(count,
```

```
cranlogs::cran_downloads(
```

```
validate(need(sum(downloads()$count)
```

```
metaReactive({
```

geom_line()

mutate(count

output$plot

downloads <-

"extend"))

metaExpr({

Sys.Date()

aes(date,

session)

output,

name"))

package

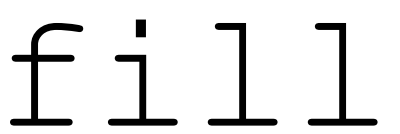
count))

mean,

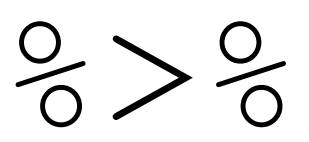
"Input

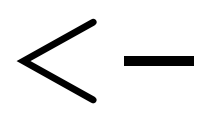
valid

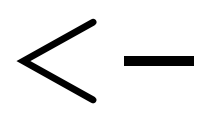
server

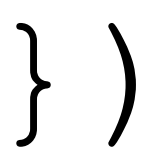

fill

from



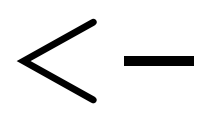






0,

7,

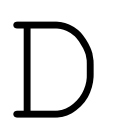


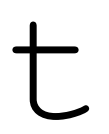
to

/

S

S

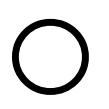


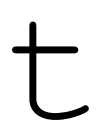


5

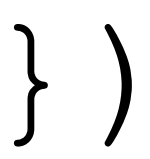
/

1





/



```
expandChain(output$plot)
```

```
output$code <- renderPrint({
```

Mark

2:

reads

reactive

Step

3:

Generate

with

expandChain()

code

```
metaRender(renderPlot,
```

```
metaReactive2({
```

downloads_rolling

function(input,

```
zoo::rollapply(count,
```

```
cranlogs::cran_downloads(
```

```
validate(need(sum(downloads()$count)
```

```
metaReactive({
```

geom_line()

mutate(count

output$plot

downloads <-

"extend"))

metaExpr({

Sys.Date()

aes(date,

session)

output,

name"))

package

mean,

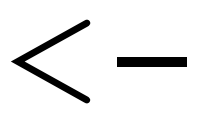
"Input

valid

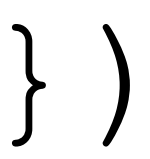
server

fill

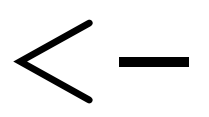
from



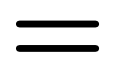
0,

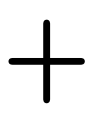


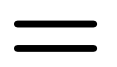
7,

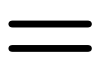


to



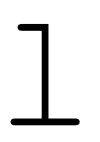






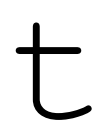
W

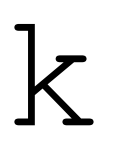
n

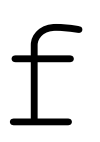


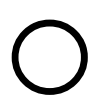
S

n



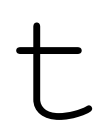




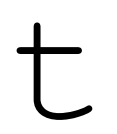


r

m



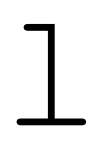
S

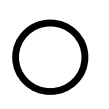


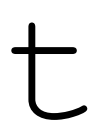
6

5

/

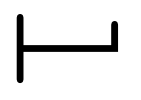




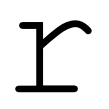


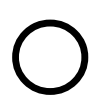
W

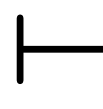
n

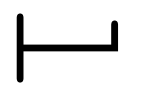


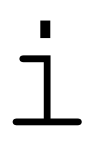
S











n

/