Reactive References

Janko Thyson

Saturday, October 11, 2014

You currently have three different options in order to specify the function that references other reactive objects and that defines the actual reactive relationship to them.

Option 1: via YAML markup (recommended)

The easiest and most compact way of making references recognizable is by specifying them in either in special YAML markup string or wrapping the YAML markup inside a comment:

```
## Via string //
"object-ref: {id: {id}, where: {where}, as {as}]"

## Via comment //
## object-ref: {id} in {where} as {ref-id}]
```

Explanation of markup components

- `{id}`: name/ID that the visible value of the referenced object has been assigned to
- `{where}` (optional): environment that the referenced object has been assigned to Default: `parent.frame()`
- '{as}' (optional): alternative name/ID that is used in the remainder of the binding function

Note

When using comments, the leading `##` are optional in the sense that the entire line must simply be a valid comment in R scripts (e.g., could also be `#` or `###` etc.)

Examples of generic structure

Here are the possible markup constellations

```
object-ref: {id: x_1}
object-ref: {id: x_1, where: where_1}
object-ref: {id: x_1, where: where_1, as: ref_1}
object-ref: {id: x_1, as: ref_1}
```

Actual examples

```
setReactiveS3(id = "x_1", value = 10)

## With curly brackets //
setReactiveS3(id = "x_2", value = function() {
    "object-ref: {id: x_1}"
    x_1 * 2
  }
)

## W/o curly brackets //
setReactiveS3(id = "x_3", value = function()
  ## object-ref: {id: x_1}
  x_1 * 2
)

## With '{as}' //
setReactiveS3(id = "x_4", value = function() {
  "object-ref: {id: x_1, as: ref_1}"
  ref_1 * 2
})

## Multiple //
setReactiveS3(id = "x_5", value = function() {
  "object-ref: {id: x_1, where: where,as: ref_1}"
  "object-ref: {id: x_2, as: ref_2}"
  ref_1 + ref_2
})
```

Note that the order **does not** matter!

```
setReactiveS3(id = "x_6", value = function() {
  "object-ref: {id: x_1, as: ref_1, where: where}"
  "object-ref: {id: x_2, as: ref_2}"
  ref_1 + ref_2
})
```

Clean up

```
rmReactive("x_1")
```

```
## [1] TRUE
```

```
rmReactive("x_2")
```

```
## [1] TRUE
```

```
rmReactive("x_3")
```

```
## [1] TRUE
```

```
rmReactive("x_4")
```

```
## [1] TRUE
```

Option 2: via argument `refs`

You can also specify the references by using the special argument `refs`:

```
refs = list(ref_1 = list(id = {id}, where = {where}))
```

Examples

```r
resetRegistry()
```

```
## [1] TRUE
```

```r
where_1 <- new.env()

setReactiveS3(id = "x_1", value = 10, where = where_1)

## With curly brackets //
setReactiveS3(id = "x_2", value = function(
    refs = list(ref_1 = list(id = "x_1", where = where_1))
  ) {
    x_1 * 2
  })

## W/o curly brackets //
setReactiveS3(id = "x_3", value = function(
    refs = list(ref_1 = list(id = "x_1", where = where_1))
  ) x_1 * 2
)

## Without explicit 'where' //
setReactiveS3(id = "x_4", value = function(
  refs = list(ref_1 = list(id = "x_1")))
  x_1 * 2
)
```

Clean up

```r
rmReactive("x_1")
```

```
## [1] FALSE
```

```r
rmReactive("x_2")
```

```
## [1] TRUE
```

```r
rmReactive("x_3")
```

```
## [1] TRUE
```

```
rmReactive("x_4")
```

```
## [1] TRUE
```

Option 3: via explicit code

You can also specify the references by using lines that start with

```
.ref_*
```

followd by `<-` and a call to `get()` of the form:

```
get({id}, {where})
```

Examples of generic structure

```
.ref_1 <- get(x = "x_1", envir = where_1)
.ref_2 <- get("x_1", where_1)
```

NOTE

1. The recognition mechanism relies on names/IDs starting with `ref_` to properly identify references
2. To be absolutely sure you retrieve the correct object, it is recommended to use `inherits = FALSE`
3. All environment objects that are used inside the binding functions should be passed along as additional arguments to either `setReactiveS3()` or `setShinyReactive()`

Actual examples

```
setReactiveS3(id = "x_1", value = 10)

## With curly brackets //
setReactiveS3(id = "x_2", value = function() {
  .ref_1 <- get(x = "x_1", inherits = FALSE)
  .ref_1 * 2
})

## W/o curly brackets //
setReactiveS3(id = "x_3", value = function()
  .ref_1 <- get(x = "x_1", inherits = FALSE)
  ## For '* 2' you would need to use curly brackets
  ## and a new line
)

## W/o argument names //
setReactiveS3(id = "x_4", value = function()
  .ref_1 <- get("x_1", inherits = FALSE)
  ## For '* 2' you would need to use curly brackets
  ## and a new line
)

## Explicit environments //
where_1 <- new.env()

setReactiveS3(id = "x_1", value = 10, where = where_1)
setReactiveS3(id = "x_2", value = function() {
  .ref_1 <- get(x = "x_1", where_1, inherits = FALSE)
  .ref_1 * 2
}, where_1 = where_1)

where_1$x_1
```

```
## [1] 10
```

```
## --> `x_1` is in `where_1`
x_2
```

```
## [1] 20
```

```
## --> `x_2` is in .GlobalEnv but references `x_1` from `where_1`
```

Clean up

```
rmReactive("x_1")
```

```
## [1] TRUE
```

```
rmReactive("x_2")
```

```
## [1] TRUE
```

```
rmReactive("x_3")
```

```
## [1] TRUE
```

```
rmReactive("x_4")
```

```
## [1] TRUE
```

```
rm(where_1)
```