Environments, Reference Behavior, & Shared Fields

OBJECT-ORIENTED PROGRAMMING WITH S3 AND R6 IN R

Richie Cotton
Data Evangelist at DataCamp

list

list

environment

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

```
lst <- list(x = pi ^ (1:5), y = matrix(month.abb, 3))
```

```
env$x <- pi ^ (1:5)
env[["y"]] <- matrix(month.abb, 3)
```

```
lst
```

```
$x
3.141593   9.869604  31.006277  97.409091 306.019685


$y
     [,1]  [,2]  [,3]  [,4]
[1,] "Jan" "Apr" "Jul" "Oct"
[2,] "Feb" "May" "Aug" "Nov"
[3,] "Mar" "Jun" "Sep" "Dec"
```

```
env
```

```
<environment: 0x103f3dfc8>
```

```r
ls.str(lst)
```

```
x :  num [1:5] 3.14 9.87 31.01 97.41 306.02
y :  chr [1:3, 1:4] "Jan" "Feb" "Mar" "Apr" "May" ...
```

```r
ls.str(env)
```

```
x :  num [1:5] 3.14 9.87 31.01 97.41 306.02
y :  chr [1:3, 1:4] "Jan" "Feb" "Mar" "Apr" "May" ...
```

```r
lst2 <- lst
(lst$x <- exp(1:5))
```

```
2.718282   7.389056  20.085537  54.598150 148.413159
```

```r
lst2$x
```

```
3.141593   9.869604  31.006277  97.409091 306.019685
```

```r
identical(lst$x, lst2$x)
```

```
FALSE
```

```r
env2 <- env
(env$x <- exp(1:5))
```

```
2.718282   7.389056  20.085537  54.598150 148.413159
```

```r
env2$x
```

```
2.718282   7.389056  20.085537  54.598150 148.413159
```

```r
identical(env$x, env2$x)
```

```
TRUE
```

copy by value

copy by value

copy by reference

```r
thing_factory <- R6Class(
  "Thing",
  private = list(
    shared = {
      e <- new.env()
      e$a_shared_field = 123
      e
    }
  ),
  active = list(
    a_shared_field = function(value) {
      if(missing(value)) {
        private$shared$a_shared_field
      } else {
        private$shared$a_shared_field <- value
      }
    }
  )
)
```

```
a_thing <- thing_factory$new()
another_thing <- thing_factory$new()
```

```
a_thing$a_shared_field
```

```
123
```

```
another_thing$a_shared_field
```

```
123
```

```
a_thing$a_shared_field <- 456
another_thing$a_shared_field
```

```
456
```

Summary

- **Create** environments with `new.env()`
- **Manipulate** them using **list syntax**
- **Environments** copy **by reference**
- **Share R6 fields** using an **environment** field

Let's practice!

OBJECT-ORIENTED PROGRAMMING WITH S3 AND R6 IN R

Cloning R6 Objects

OBJECT-ORIENTED PROGRAMMING WITH S3 AND R6 IN R

Richie Cotton
Data Evangelist at DataCamp

- Environments use copy by reference
- So do **R6** objects

```r
thing_factory <- R6Class(
  "Thing",
  private = list(
    ..a_field = 123
  ),
  active = list(
    a_field = function(value) {
      if(missing(value)) {
        private$..a_field
      } else {
        private$..a_field <- value
      }
    }
  )
)
```

```r
a_thing <- thing_factory$new()
a_copy <- a_thing
a_thing$a_field <- 456
```

```r
a_copy$a_field
```

```
456
```

`clone()` **copies by value**

```r
a_clone <- a_thing$clone()
```

```r
a_thing$a_field <- 789
a_clone$a_field
```

```
456
```

```r
container_factory <- R6Class(
  "Container",
  private = list(
    ..thing = thing_factory$new()
  ),
  active = list(
    thing = function(value) {
      if(missing(value)) {
        private$..thing
      } else {
        private$..thing <- value
      }
    }
  )
)
```

```r
a_container <- container_factory$new()
a_clone <- a_container$clone()
```

```r
a_container$thing$a_field <- "a new value"
a_clone$thing$a_field
```

```
"a new value"
```

```r
a_deep_clone <- a_container$clone(deep = TRUE)
```

```r
a_container$thing$a_field <- "a different value"
a_deep_clone$thing$a_field
```

```
"a new value"
```

Summary

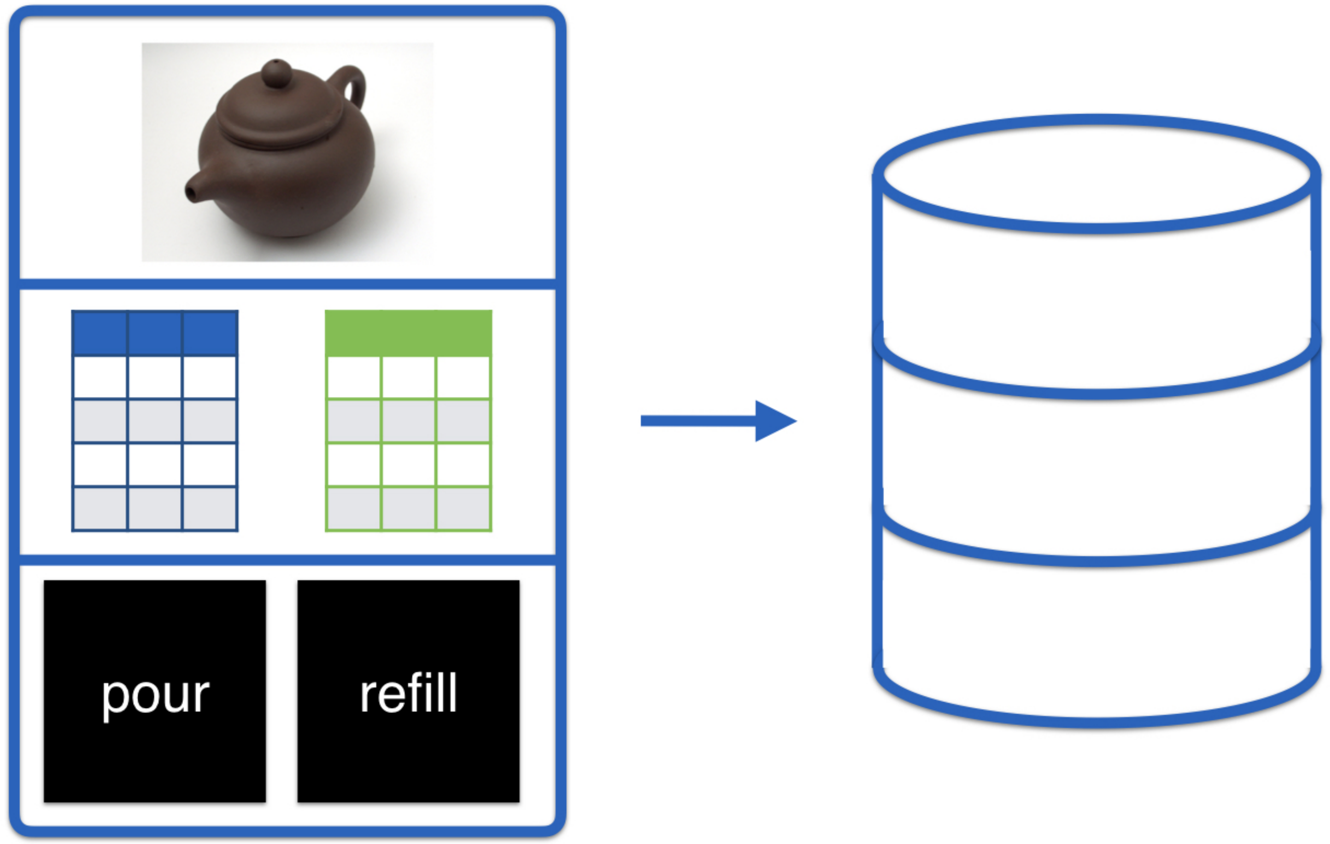
- **R6** objects copy **by reference**
- **Copy** them **by value** using `clone()`
- `clone()` is **autogenerated**
- `clone(deep = TRUE)` is for **R6** fields

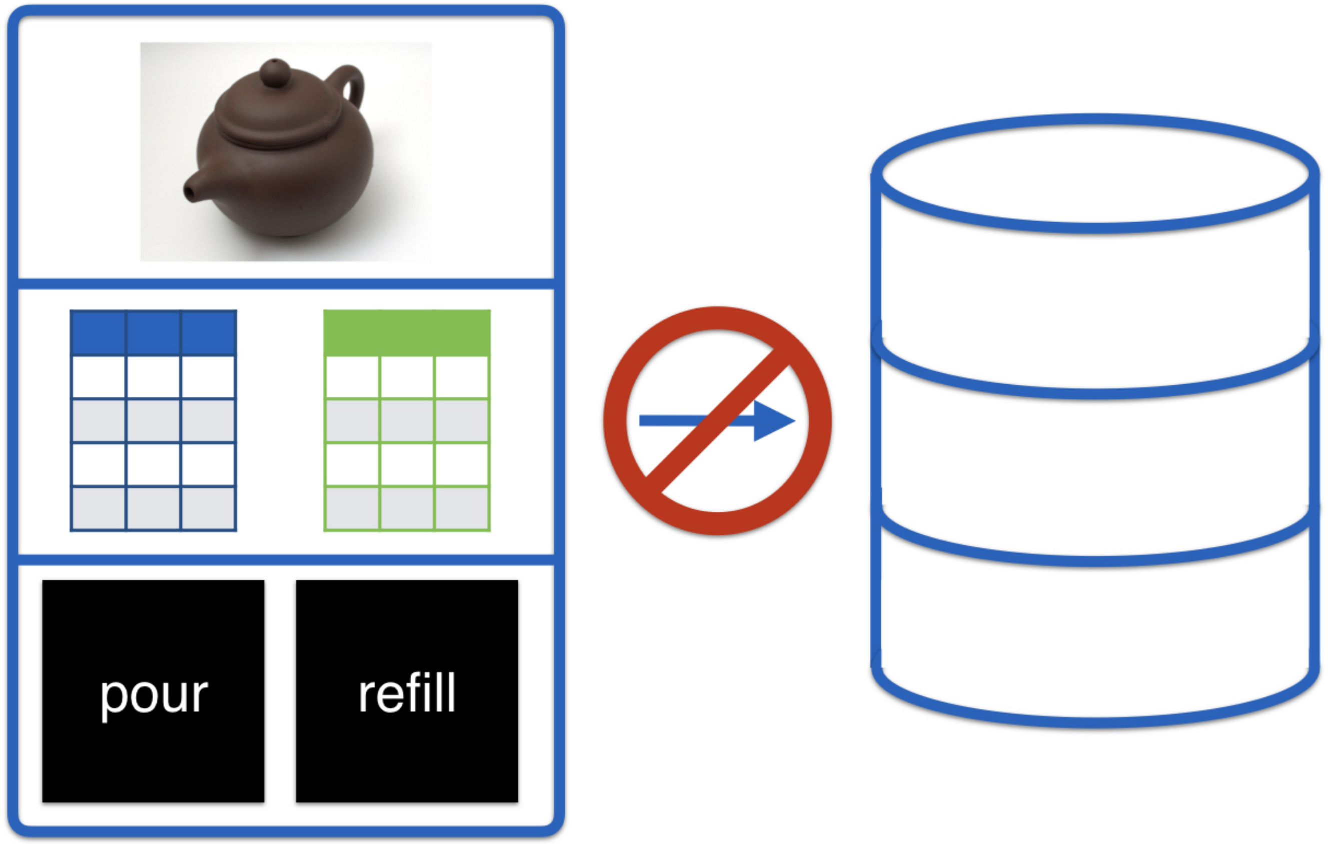
Let's practice!

Shut it Down

OBJECT-ORIENTED PROGRAMMING WITH S3 AND R6 IN R

Richie Cotton
Data Evangelist at DataCamp


pour
refill


pour
refill

`initialize()` customizes **startup**

`finalize()` customizes **cleanup**

```
thing_factory <- R6Class(
  "Thing",
  private = list(
    ..a_field = 123
  ),
  public = list(
    initialize = function(a_field) {
      if(!missing(a_field)) {
        private$a_field = a_field
      }
    },
    finalize = function() {
      message("Finalizing the Thing")
    }
  )
)
```

```r
a_thing <- thing_factory$new()
```

```r
rm(a_thing)
```

```r
gc()
```

```
Finalizing the Thing
         used (Mb) gc trigger (Mb) max used (Mb)
Ncells 443079 23.7     750400 40.1   592000 31.7
Vcells 718499  5.5    1308461 10.0  1092342  8.4
```

```
library(RSQLite)
database_manager_factory <- R6Class(
  "DatabaseManager",
  private = list(
   conn = NULL
  ),
  public = list(
    initialize = function(a_field) {
      private$conn <- dbConnect("some-database.sqlite")
    },
    finalize = function() {
      dbDisconnect(private$conn)
    }
  )
)
```

Summary

- `finalize()` **cleans up** after **R6** objects
- It is useful when **working with databases**
- It gets called during **garbage collection**

Let's practice!

OBJECT-ORIENTED PROGRAMMING WITH S3 AND R6 IN R