Package: chains (via r-universe)

June 4, 2026

Title Complex Reactive Values for 'shiny'

Version 0.0.0.9017

Description Create hybrid reactive objects that can be set imperatively but are validated reactively, ensuring their state is always consistent with their dependencies.

License MIT + file LICENSE

URL https://chains.shinyworks.org, https://github.com/shinyworks/chains

BugReports https://github.com/shinyworks/chains/issues

Imports rlang, shiny, stbl (>= 0.3.0)

Suggests astgrepr, covr, knitr, pkgload, rmarkdown, shinytest2, testthat (>= 3.0.0)

Config/Needs/website rmarkdown, quarto, shinylive, fs

Config/testthat/edition 3

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository https://shinyworks.r-universe.dev

Date/Publication 2026-04-05 13:23:26 UTC

RemoteUrl https://github.com/shinyworks/chains

RemoteRef HEAD

RemoteSha be472b9f9179446170c28aa7df03272f4e8d69d3

Contents

.vrv 2
extract_error 3
is_default 4
validated_reactive_val 5
vrv_chr 7
vrv_dbl 8
vrv_fct 10
vrv_int 11
vrv_lgl 13

Index **15**

`.vrv` *Access the validated reactive value pronoun*

Description

When used inside the `validation_expr` of a `validated_reactive_val()`, this function acts as a pronoun to access the current value (before validation) of that `validated_reactive_val`. This allows the validation expression to reconcile the object's current state with its reactive dependencies.

Usage

```
.vrv(env = rlang::caller_env())
```

Arguments

`env`	(`environment`) The environment in which to find the `.vrv` pronoun. This should generally not be changed from the default of `rlang::caller_env()`.

Details

To reference `.vrv()` in a package and avoid `R CMD check` notes, you can either import this function with `#' @importFrom chains .vrv` or call it with the `chains::.vrv()` namespace.

Value

The current value of the `validated_reactive_val()`.

`extract_error` *Extract an error message from an object*

Description

A convenience function to extract the error message attached to an object. A method is implemented for `validated_reactive_val()` objects.

Usage

```
extract_error(x, ..., capture = TRUE)
```

Arguments

`x`	(any) The object from which to extract the error message.
`...`	Additional arguments passed to methods.
`capture`	(length-1 `logical`) If `TRUE`, the error is captured and returned with class `captured-error` (as well as `captured-` prepended on any error subclasses). If `FALSE`, the error is thrown.

Value

If the object contains an error message, an object with class `captured-error` if `capture` is `TRUE` or the error condition if `capture` is `FALSE`. Depending on your purpose, you may need to `signalCondition()` or `rlang::cnd_signal()` to actually signal the error. If the object does not contain an error, `NULL`.

Examples

```
vrv <- validated_reactive_val(
  value = "good",
  default = "default",
  validation_expr = {
    if (.vrv() == "bad") {
      rlang::abort("is bad", class = "special_error")
    }
    .vrv()
  }
)
shiny::isolate(extract_error(vrv))
vrv("bad")
shiny::isolate(vrv())
captured_error <- shiny::isolate(extract_error(vrv))
class(captured_error)
captured_error$message
raw_error <- shiny::isolate(extract_error(vrv, capture = FALSE))
try(rlang::cnd_signal(raw_error))
```

`is_default` *Determine whether a reactive is using its default value*

Description

A convenience function to check whether an object is currently using its default value. A method is implemented for `validated_reactive_val()` objects.

Usage

```
is_default(x, ...)
```

Arguments

`x`	(any) The object to test.
`...`	Additional arguments passed to methods.

Value

A `logical` value indicating whether the object is currently using its default value.

Examples

```
vrv <- validated_reactive_val(
  value = "good",
  default = "default",
  validation_expr = {
    if (.vrv() == "bad") {
      rlang::abort("is bad")
    }
    .vrv()
  }
)
shiny::isolate(vrv())
shiny::isolate(is_default(vrv))
vrv("bad")
shiny::isolate(vrv())
shiny::isolate(is_default(vrv))
```

validated_reactive_val *Create a validated reactive value*

Description

Create a reactive value that can be updated imperatively (like a `shiny::reactiveVal()`) but whose state is ultimately governed by a reactive validation expression. This ensures that its value is always consistent with its reactive dependencies before any downstream dependents are re-evaluated.

Usage

```
validated_reactive_val(
  validation_expr,
  value = NULL,
  default = NULL,
  label = NULL,
  env = rlang::caller_env()
)

## S3 method for class 'vrv'
x$name
```

Arguments

validation_expr	(`expression`) A reactive expression that defines the authoritative state of this `validated_reactive_value`, This expression can access the `validated_reactive_value`'s own current value via the `.vrv()` pronoun to reconcile it with upstream dependencies.
value	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
default	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.
label	(length-1 `character` or `NULL`) An optional label for the `validated_reactive_value`, used for debugging.
env	(`environment`) The environment in which to evaluate the `validation_expr`.
x	(`vrv`) A `vrv` object.
name	(length-1 `character`) The name of the helper function to access. Expects one of `value`, `error`, or `is_default`. Other values return `NULL`.

Value

A `vrv` object, which is a function with a custom class. Call it with no arguments to (reactively) read the validated value. Call it with a single argument to imperatively set the value; it will be automatically validated on the next read. You can also access the most recent validation error with `my_vrv$error()` and check if the current value is the default with `my_vrv$is_default()`.

Examples

```
library(shiny)

ui <- fluidPage( selectInput("level", "Level", choices = c("A", "B")),
uiOutput("group_ui"), textOutput("current_group") )

server <- function(input, output, session) {
  # `group_val` depends on `input$level` and can also be set by `input$group`.
  group_val <- validated_reactive_val(
    value = "A1",
    validation_expr = {
      # If the current value is not valid for the new level, reset it.
      valid_groups <- if (input$level == "A") {
        c("A1", "A2")
      } else {
        c("B1", "B2")
      }
      if (.vrv() %in% valid_groups) {
        .vrv()
      } else {
        valid_groups[[1]]
      }
    }
  )

  # When the user changes the group input, imperatively update the vrv.
  observeEvent(input$group, {
    group_val(input$group)
  })

  # The UI for the group dropdown is dynamic.
  output$group_ui <- renderUI({
    choices <- if (input$level == "A") c("A1", "A2") else c("B1", "B2")
    selectInput("group", "Group", choices = choices, selected = group_val())
  })

  output$current_group <- renderText({
    paste("Current Validated Group:", group_val())
  })
}

shinyApp(ui, server)
```

vrv_chr *Create a validated reactive character vector*

Description

A wrapper around `validated_reactive_val()` that uses `stbl::stabilize_chr()` to validate and coerce its value. This is a convenience function that constructs the `validation_expr` for you.

Usage

```
vrv_chr(
  value = NULL,
  default = NULL,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  regex = NULL,
  label = NULL,
  env = rlang::caller_env()
)

vrv_chr_scalar(
  value = NULL,
  default = NULL,
  label = NULL,
  allow_null = FALSE,
  allow_zero_length = FALSE,
  allow_na = TRUE,
  regex = NULL,
  env = rlang::caller_env()
)
```

Arguments

`value`	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
`default`	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.
`allow_null`	(length-1 `logical`, `expression`, or `reactive expression`) Is `NULL` an acceptable value?
`allow_na`	(length-1 `logical`, `expression`, or `reactive expression`) Are `NA` values okay?
`min_size`	(length-1 `integer`, `expression`, or `reactive expression`) The minimum size of the vector.

max_size	(length-1 integer, expression, or reactive expression) The maximum size of the vector.
regex	(character, expression, or reactive expression) One or more regular expressions to test against the values. See stbl::stabilize_chr() for details.
label	(length-1 character or NULL) An optional label for the validated_reactive_value, used for debugging.
env	(environment) The environment in which to evaluate the validation_expr.
allow_zero_length	(length-1 logical, expression, or reactive expression) Is a zero-length vector acceptable?

Value

A vrv object which returns a validated character vector.

---

vrv_dbl *Create a validated reactive double vector*

Description

A wrapper around validated_reactive_val() that uses stbl::stabilize_dbl() to validate and coerce its value. This is a convenience function that constructs the validation_expr for you.

Usage

```
vrv_dbl(
  value = NULL,
  default = NULL,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_value = NULL,
  max_value = NULL,
  label = NULL,
  env = rlang::caller_env()
)

vrv_dbl_scalar(
  value = NULL,
  default = NULL,
  label = NULL,
  allow_null = FALSE,
  allow_zero_length = FALSE,
```

```
  allow_na = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_value = NULL,
  max_value = NULL,
  env = rlang::caller_env()
)
```

Arguments

value	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
default	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.
allow_null	(length-1 `logical`, `expression`, or `reactive expression`) Is `NULL` an acceptable value?
allow_na	(length-1 `logical`, `expression`, or `reactive expression`) Are `NA` values okay?
min_size	(length-1 `integer`, `expression`, or `reactive expression`) The minimum size of the vector.
max_size	(length-1 `integer`, `expression`, or `reactive expression`) The maximum size of the vector.
coerce_character	(length-1 `logical`, `expression`, or `reactive expression`) Should character vectors such as `"1"` and `"2.0"` be coerced to double?
coerce_factor	(length-1 `logical`, `expression`, or `reactive expression`) Should factors with values such as `"1"` and `"2.0"` be coerced to double? Note that this function uses the character value from the factor, while `as.double()` uses the integer index of the factor.
min_value	(length-1 `double`, `expression`, or `reactive expression`, or `NULL`) The lowest allowed value for `x`. If `NULL` (default) values are not checked.
max_value	(length-1 `double`, `expression`, or `reactive expression`, or `NULL`) The highest allowed value for `x`. If `NULL` (default) values are not checked.
label	(length-1 `character` or `NULL`) An optional label for the `validated_reactive_value`, used for debugging.
env	(`environment`) The environment in which to evaluate the `validation_expr`.
allow_zero_length	(length-1 `logical`, `expression`, or `reactive expression`) Is a zero-length vector acceptable?

Value

A `vrv` object which returns a validated double vector.

vrv_fct *Create a validated reactive factor-like character vector*

Description

A wrapper around `validated_reactive_val()` to help manage factor-like values, especially in a cascading / dependent manner. This is a convenience function that constructs the `validation_expr` for you by wrapping `stbl::stabilize_fct()`. Note that the objects returned by the resulting `vrv` are `character` vectors, not `factor`s, to allow values to remain unchanged when the allowed levels change but the value is still valid.

Usage

```
vrv_fct(
  levels,
  value = NULL,
  default = NULL,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  to_na = character(),
  label = NULL,
  env = rlang::caller_env()
)

vrv_fct_scalar(
  levels,
  value = NULL,
  default = NULL,
  allow_null = FALSE,
  allow_zero_length = FALSE,
  allow_na = TRUE,
  to_na = character(),
  label = NULL,
  env = rlang::caller_env()
)
```

Arguments

`levels`	(`expression`) An expression that returns a `character` vector of valid levels. Can be a reactive expression.
`value`	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
`default`	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.

allow_null	(length-1 logical, expression, or reactive expression) Is NULL an acceptable value?
allow_na	(length-1 logical, expression, or reactive expression) Are NA values okay?
min_size	(length-1 integer, expression, or reactive expression) The minimum size of the vector.
max_size	(length-1 integer, expression, or reactive expression) The maximum size of the vector.
to_na	(expression) Values to coerce to NA. Can be a reactive expression.
label	(length-1 character or NULL) An optional label for the validated_reactive_value, used for debugging.
env	(environment) The environment in which to evaluate the validation_expr.
allow_zero_length	(expression) Is a zero-length vector acceptable? Can be a reactive expression.

Value

A vrv object which returns a factor-like character vector.

---

vrv_int *Create a validated reactive integer vector*

Description

A wrapper around validated_reactive_val() that uses stbl::stabilize_int() to validate and coerce its value. This is a convenience function that constructs the validation_expr for you.

Usage

```
vrv_int(
  value = NULL,
  default = NULL,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_value = NULL,
  max_value = NULL,
  label = NULL,
  env = rlang::caller_env()
)

vrv_int_scalar(
```

```
  value = NULL,
  default = NULL,
  label = NULL,
  allow_null = FALSE,
  allow_zero_length = FALSE,
  allow_na = TRUE,
  coerce_character = TRUE,
  coerce_factor = TRUE,
  min_value = NULL,
  max_value = NULL,
  env = rlang::caller_env()
)
```

Arguments

value	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
default	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.
allow_null	(length-1 `logical`, `expression`, or `reactive expression`) Is `NULL` an acceptable value?
allow_na	(length-1 `logical`, `expression`, or `reactive expression`) Are `NA` values okay?
min_size	(length-1 `integer`, `expression`, or `reactive expression`) The minimum size of the vector.
max_size	(length-1 `integer`, `expression`, or `reactive expression`) The maximum size of the vector.
coerce_character	(length-1 `logical`, `expression`, or `reactive expression`) Should character vectors such as `"1"` and `"2.0"` be coerced to integer?
coerce_factor	(length-1 `logical`, `expression`, or `reactive expression`) Should factors with values such as `"1"` and `"2.0"` be coerced to integer? Note that this function uses the character value from the factor, while `as.integer()` uses the integer index of the factor.
min_value	(length-1 `integer`, `expression`, or `reactive expression`, or `NULL`) The lowest allowed value for `x`. If `NULL` (default) values are not checked.
max_value	(length-1 `integer`, `expression`, or `reactive expression`, or `NULL`) The highest allowed value for `x`. If `NULL` (default) values are not checked.
label	(length-1 `character` or `NULL`) An optional label for the `validated_reactive_value`, used for debugging.
env	(`environment`) The environment in which to evaluate the `validation_expr`.
allow_zero_length	(length-1 `logical`, `expression`, or `reactive expression`) Is a zero-length vector acceptable?

Value

A `vrv` object which returns a validated integer vector.

`vrv_lgl` *Create a validated reactive logical vector*

Description

A wrapper around `validated_reactive_val()` that uses `stbl::stabilize_lgl()` to validate and coerce its value. This is a convenience function that constructs the `validation_expr` for you.

Usage

```
vrv_lgl(
  value = NULL,
  default = NULL,
  allow_null = TRUE,
  allow_na = TRUE,
  min_size = NULL,
  max_size = NULL,
  label = NULL,
  env = rlang::caller_env()
)

vrv_lgl_scalar(
  value = NULL,
  default = NULL,
  label = NULL,
  allow_null = FALSE,
  allow_zero_length = FALSE,
  allow_na = TRUE,
  env = rlang::caller_env()
)
```

Arguments

`value`	(various) The initial value. This value will be coerced via the validation expression when accessed. If this value is reactive, an observer with `priority = Inf` will be created to attempt to keep the validated value in sync with that reactive.
`default`	(various, including `expression` or `reactive expression`) A value to use when the current value is not valid according to the defined rules. Defaults to `NULL`.
`allow_null`	(length-1 `logical`, `expression`, or `reactive expression`) Is `NULL` an acceptable value?
`allow_na`	(length-1 `logical`, `expression`, or `reactive expression`) Are `NA` values okay?

min_size	(length-1 integer, expression, or reactive expression) The minimum size of the vector.
max_size	(length-1 integer, expression, or reactive expression) The maximum size of the vector.
label	(length-1 character or NULL) An optional label for the validated_reactive_value, used for debugging.
env	(environment) The environment in which to evaluate the validation_expr.
allow_zero_length	(length-1 logical, expression, or reactive expression) Is a zero-length vector acceptable?

Value

A vrv object which returns a validated logical vector.

Index

.vrv, 2
$.vrv *(validated_reactive_val)*, 5

extract_error, 3

is_default, 4

rlang::caller_env(), *2*
rlang::cnd_signal(), *3*

shiny::reactiveVal(), *5*
signalCondition(), *3*
stbl::stabilize_chr(), *7, 8*
stbl::stabilize_dbl(), *8*
stbl::stabilize_fct(), *10*
stbl::stabilize_int(), *11*
stbl::stabilize_lgl(), *13*

validated_reactive_val, 5
validated_reactive_val(), *2–4*, *7, 8*, *10, 11*, *13*
vrv_chr, 7
vrv_chr_scalar *(vrv_chr)*, 7
vrv_dbl, 8
vrv_dbl_scalar *(vrv_dbl)*, 8
vrv_fct, 10
vrv_fct_scalar *(vrv_fct)*, 10
vrv_int, 11
vrv_int_scalar *(vrv_int)*, 11
vrv_lgl, 13
vrv_lgl_scalar *(vrv_lgl)*, 13