Package: jsplyr (via r-universe)

July 17, 2026

Title Manipulate 'JSON' Data in the Browser with a 'dplyr' Interface

Version 0.1.0

Description A 'dplyr' backend for 'shiny' applications that manipulates 'JSON' data in the web browser instead of on the server. Data manipulation verbs such as filter, select, mutate, summarise, arrange, and joins are evaluated lazily and translated into 'JavaScript' operations that run client-side, following the lazy evaluation approach of 'dbplyr' but generating 'JavaScript' rather than 'SQL'. Results are returned to R asynchronously as promises. This keeps data wrangling responsive for large data frames by offloading the work to the client.

License MIT + file LICENSE

URL https://github.com/r-world-devs/jsplyr, https://r-world-devs.github.io/jsplyr/

BugReports https://github.com/r-world-devs/jsplyr/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports cli, dplyr, glue, htmltools, jsonlite, promises, purrr, rlang, shiny, stringr

Suggests knitr, rmarkdown, shinytest2, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

Repository https://r-world-devs.r-universe.dev

Date/Publication 2026-07-17 07:56:53 UTC

RemoteUrl https://github.com/r-world-devs/jsplyr

RemoteRef HEAD

RemoteSha 9667bff598c4bed8afaceb299dbfedd8c98b8dce

Contents

`arrange` *Arrange rows of* `JSON` *data by column values.*

Description

Arrange rows of `JSON` data by column values.

Usage

```
## S3 method for class 'tbl_lazy_json'
arrange(.data, ..., .by_group = FALSE)
```

Arguments

`.data`	A `tbl_lazy_json` object.
`...`	Columns to sort by. Wrap a column in `dplyr::desc()` to sort it in descending order. Multiple columns break ties left to right. Bare column names and character strings are both accepted.
`.by_group`	Ignored. Accepted for consistency with the generic; grouped arrange ordering is not applied for `tbl_lazy_json`.

Details

Sorting happens in the browser and is stable across ties, so the original row order is preserved within equal keys.

Value

A `tbl_lazy_json` object with the sort appended as a lazy compute step. The ordering is applied in the browser when the pipeline is evaluated by `compute()`/`collect()`.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |>
    arrange(cyl, desc(mpg))
}
```

`collect` *Retrieve* `JSON` *data from the browser.*

Description

Retrieve `JSON` data from the browser.

Usage

```
## S3 method for class 'tbl_lazy_json'
collect(x, ..., raw = FALSE)
```

Arguments

`x`	A `tbl_lazy_json` object.
`...`	Unused. Provided for consistency with generic.
`raw`	A logical, if `TRUE` returns `JSON` as character.

Value

A `promises::promise()`. When `raw = FALSE` (the default) it resolves to a tibble::tibble built from the computed `JSON`; when `raw = TRUE` it resolves to the raw `JSON` string. The result is fetched asynchronously from the browser, so the value is a promise rather than a data frame.

compute *Compute JSON data in the browser.*

Description

Compute JSON data in the browser.

Usage

```
## S3 method for class 'tbl_lazy_json'
compute(x, ...)
```

Arguments

x	A tbl_lazy_json object.
...	Unused. Provided for consistency with generic.

Value

A tbl_lazy_json object with a pending computation attached. It carries a .promise field holding a promises::promise() that resolves when the browser posts back the computed JSON. The accumulated compute steps are reset so the object can be extended or collected further.

copy_to.ShinySession *Copy a local or remote data frame to the browser*

Description

Copy a local or remote data frame to the browser

Usage

```
## S3 method for class 'ShinySession'
copy_to(dest, df, ...)
```

Arguments

dest	A shiny session object.
df	A local data.frame or a name of the JSON data in the browser.
...	Unused. Provided for consistency with generic.

Value

Invisibly, a tbl_lazy_json object referencing the data now registered in the browser, ready to be piped into the data manipulation verbs. Called primarily for the side effect of sending the data to the client.

count *Count observations in* JSON *data.*

Description

count() groups the data by the given columns and counts the rows in each group. tally() counts rows for the existing grouping set by group_by() without adding new grouping columns. Both build on the group_by()/summarise() machinery and run in the browser.

Usage

```
## S3 method for class 'tbl_lazy_json'
count(x, ..., wt = NULL, sort = FALSE, name = NULL)

## S3 method for class 'tbl_lazy_json'
tally(x, wt = NULL, sort = FALSE, name = NULL)
```

Arguments

x	A tbl_lazy_json object.
...	Columns to group by before counting (for count()). Accepts the same inputs as group_by().
wt	Not supported; weighted counts are not implemented for tbl_lazy_json. Supplying a non-NULL value raises an error.
sort	If TRUE, order the result by the count column descending.
name	Name of the count column in the output. NULL uses "n".

Value

A tbl_lazy_json object with the count appended as a lazy compute step. When evaluated it yields one row per group with the count column (named by name, default "n") added.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |> count(cyl)
  tbl(session, "mtcars") |> count(cyl, sort = TRUE)
  tbl(session, "mtcars") |> group_by(cyl) |> tally()
}
```

distinct *Keep distinct records of* JSON *data.*

Description

Keep distinct records of JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
distinct(.data, ..., .keep_all = FALSE)
```

Arguments

.data	A tbl_lazy_json object.
...	Column names to determine uniqueness. If empty, all columns are used.
.keep_all	If TRUE, keep all columns in the output. When FALSE (the default) and columns are supplied in ..., only those columns are returned, matching dplyr::distinct().

Value

A tbl_lazy_json object with the de-duplication appended as a lazy compute step, applied in the browser when the pipeline is evaluated.

filter *Add filter to* JSON *data.*

Description

Add filter to JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
filter(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Filtering expressions. Comparisons (==, >, <, etc.) combined with &/| are supported, as well as the dplyr helpers is.na(), between(), and across()/if_all()/if_any(). across()/if_all() and if_any() expand the predicate over the selected columns, combining them with & and | respectively. Column selections accept c(...), a bare column, a character vector, and all_of()/any_of(). Values referenced from input or the calling environment are resolved on the R side; column references are evaluated in the browser.

Value

A tbl_lazy_json object with the filter appended as a lazy compute step. Rows are subset in the browser when the pipeline is evaluated by compute()/collect().

---

group_by *Group JSON data by one or more columns.*

Description

Group JSON data by one or more columns.

Usage

```
## S3 method for class 'tbl_lazy_json'
group_by(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Columns to group by. Accepts bare column names as well as the tidyselect helpers c(...), all_of()/any_of(), and across(), which are resolved to plain column names on the R side.

Value

A tbl_lazy_json object carrying the grouping as browser-side state. The grouping is consumed by group-aware verbs such as summarise(), count(), and the slice() family when the pipeline is evaluated.

---

include_jsplyr *Link JS code.*

Description

Include the jsplyr JavaScript code in a Shiny app. To be put in the UI part of the app.

Usage

```
include_jsplyr()
```

Value

An htmlDependency object.

joins *Join two* JSON *tables.*

Description

Mutating joins (left_join, right_join, inner_join, full_join) combine columns from x and y, matching rows by key columns. Filtering joins (semi_join, anti_join) keep columns from x, using y only to determine which rows to keep.

Usage

```
## S3 method for class 'tbl_lazy_json'
left_join(x, y, by = NULL, ...)

## S3 method for class 'tbl_lazy_json'
right_join(x, y, by = NULL, ...)

## S3 method for class 'tbl_lazy_json'
inner_join(x, y, by = NULL, ...)

## S3 method for class 'tbl_lazy_json'
full_join(x, y, by = NULL, ...)

## S3 method for class 'tbl_lazy_json'
semi_join(x, y, by = NULL, ...)

## S3 method for class 'tbl_lazy_json'
anti_join(x, y, by = NULL, ...)
```

Arguments

x	A tbl_lazy_json object (the left table).
y	A tbl_lazy_json object (the right table). Must share the same browser session as x.
by	A character vector of columns to join by. Use a named vector (e.g. c("a" = "b")) to match columns with different names in x and y. If NULL (the default), a natural join is performed using all columns common to both tables.
...	Unused. Provided for consistency with the generics.

Value

A tbl_lazy_json object with the join appended as a compute step.

mutate *Add or modify columns in JSON data.*

Description

Add or modify columns in JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
mutate(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Name-value pairs of expressions. The name gives the name of the column in the output. The value should be an expression using existing columns, e.g. new_col = col1 + col2. Conditional helpers are translated to JavaScript: ifelse() and if_else() become ternary operators, and case_when() becomes a chained set of ternary operators. case_when() clauses without a TRUE ~ ... catch-all yield null for unmatched rows, matching dplyr's NA default. across() is expanded on the R side into one column per selection. It accepts a single function (across(c(a, b), round)), a formula lambda (across(c(a, b), ~ .x * 2)), or a named list of functions (across(c(a, b), list(double = ~ .x * 2))). Column selections accept c(...), a bare column, a character vector, and all_of()/any_of(). Use .names with the {.col}/{.fn} glue placeholders to control the output column names; by default a single function reuses the input name and multiple functions produce {.col}_{.fn}.

Value

A tbl_lazy_json object with the column additions/modifications appended as a lazy compute step, evaluated in the browser when the pipeline is computed.

pull *Extract a single column from JSON data as a vector.*

Description

Like collect(), pull() retrieves data asynchronously from the browser, so it returns a promises::promise() that resolves to a vector rather than returning the vector directly.

Usage

```
## S3 method for class 'tbl_lazy_json'
pull(.data, var = -1, name = NULL, ...)
```

Arguments

.data	A tbl_lazy_json object.
var	The column to extract. A bare name, a string, or a position. Positive positions count from the left; negative positions count from the right (e.g. -1 is the last column), matching dplyr::pull().
name	Ignored. Accepted for consistency with the generic; named vectors are not produced for tbl_lazy_json.
...	Unused. Provided for consistency with the generic.

Value

A promise resolving to a vector with the column's values.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |>
    pull(mpg) %...>% print()
}
```

---

relocate *Change column order of* JSON *data.*

Description

Change column order of JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
relocate(.data, ..., .before = NULL, .after = NULL)
```

Arguments

.data	A tbl_lazy_json object.
...	Columns to move. Bare names and character strings are accepted.
.before, .after	Destination of the columns selected by ... Supply a single column (bare name or string) to place the moved columns before or after it. With neither, the selected columns move to the front.

Details

Reordering happens in the browser by rebuilding each row's keys in the new order. Columns not named in ... keep their relative order.

Value

A tbl_lazy_json object with the column reordering appended as a lazy compute step, applied in the browser when the pipeline is evaluated.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |> relocate(gear, carb)
  tbl(session, "mtcars") |> relocate(mpg, .after = cyl)
}
```

---

rename *Rename columns of* JSON *data.*

Description

Rename columns of JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
rename(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Use new_name = old_name to rename columns. Both bare names and character strings are accepted (e.g. rename(mpg_new = mpg) or rename("mpg_new" = "mpg")). Column order is preserved.

Value

A tbl_lazy_json object with the rename appended as a lazy compute step, applied in the browser when the pipeline is evaluated.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |>
    rename(miles_per_gallon = mpg, cylinders = cyl)
}
```

select *Select columns from* JSON *data.*

Description

Select columns from JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
select(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Column names.

Value

A tbl_lazy_json object with the column selection appended as a lazy compute step, applied in the browser when the pipeline is evaluated.

show_query *Present computation steps.*

Description

Present computation steps.

Usage

```
## S3 method for class 'tbl_lazy_json'
show_query(x, ...)
```

Arguments

x	A tbl_lazy_json object.
...	Filtering expressions.

Value

The tbl_lazy_json object x, invisibly. Called for the side effect of printing the accumulated compute steps to the console.

slice *Select rows of* JSON *data by position.*

Description

The slice() family picks rows by position or by ordering on a column. When the data has been grouped with group_by(), slicing is applied within each group.

Usage

```
## S3 method for class 'tbl_lazy_json'
slice(.data, ..., .by = NULL, .preserve = FALSE)

## S3 method for class 'tbl_lazy_json'
slice_head(.data, ..., n, prop, by = NULL)

## S3 method for class 'tbl_lazy_json'
slice_tail(.data, ..., n, prop, by = NULL)

## S3 method for class 'tbl_lazy_json'
slice_min(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)

## S3 method for class 'tbl_lazy_json'
slice_max(
  .data,
  order_by,
  ...,
  n,
  prop,
  by = NULL,
  with_ties = TRUE,
  na_rm = FALSE
)
```

Arguments

.data	A tbl_lazy_json object.

`...`	For `slice()`, integer row positions to keep (1-based). Negative positions drop rows. Unused by the other variants.
`.preserve`	Ignored. Accepted for consistency with the generic.
`n`	Number of rows to keep. Used by `slice_head()`, `slice_tail()`, `slice_min()`, and `slice_max()`. Defaults to 1 where applicable.
`prop`	Proportion of rows to keep (0-1), an alternative to `n` for `slice_head()`/`slice_tail()`/`slice_min()`/`slic`
`by, .by`	Ignored. Accepted for consistency with the generics; per-call grouping is not applied (use `group_by()`, which slicing respects).
`order_by`	For `slice_min()`/`slice_max()`, the column to order by (bare name or string).
`with_ties, na_rm`	Ignored. Accepted for consistency with the `slice_min()`/`slice_max()` generics.

Details

All slicing is evaluated in the browser. `slice_min()`/`slice_max()` order rows by the given column (ascending for min, descending for max) and keep the first `n` (or `prop`).

Value

A `tbl_lazy_json` object with the row selection appended as a lazy compute step. When the data is grouped with `group_by()`, the selection is applied within each group in the browser at compute time.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |> slice(1, 3, 5)
  tbl(session, "mtcars") |> slice_head(n = 5)
  tbl(session, "mtcars") |> group_by(cyl) |> slice_max(mpg, n = 2)
}
```

---

summarise *Summarise* JSON *data.*

Description

Summarise JSON data.

Usage

```
## S3 method for class 'tbl_lazy_json'
summarise(.data, ...)
```

Arguments

.data	A tbl_lazy_json object.
...	Name-value pairs of summary functions. The name gives the name of the column in the output. The value should be a call to a summary function like mean(), sum(), min(), max(), n(). across() is expanded on the R side into one summary per selected column. It accepts a single function (across(c(a, b), mean)) or a named list of functions (across(c(a, b), list(mean = mean, sd = sd))). Column selections accept c(...), a bare column, a character vector, and all_of()/any_of(). Use .names with the {.col}/{.fn} glue placeholders to control the output column names; by default a single function reuses the input name and multiple functions produce {.col}_{.fn}.

Value

A tbl_lazy_json object with the aggregation appended as a lazy compute step. When evaluated it yields one row per group (or a single row when ungrouped) with the summary columns.

---

tbl_lazy_json *Create a lazy JSON tbl*

Description

Create a lazy JSON tbl

Usage

```
tbl_lazy_json(session, json_name, compute_steps = list())
```

Arguments

session	A shiny session object.
json_name	A character.
compute_steps	A list of compute steps to be triggered when compute() is called.

Value

An object of class tbl_lazy_json: a list holding the shiny session, a state_id keying the data in the browser, and the list of compute_steps to run when the pipeline is computed.

ungroup *Remove grouping from* JSON *data.*

Description

Drops grouping previously set with `group_by()`. With no arguments all grouping is removed; supplying column names removes only those columns from the grouping set (partial ungroup), matching `dplyr::ungroup()`.

Usage

```
## S3 method for class 'tbl_lazy_json'
ungroup(x, ...)
```

Arguments

x	A `tbl_lazy_json` object.
...	Columns to remove from the grouping. Accepts the same inputs as `group_by()` (bare names, strings, and the tidyselect helpers `c(...)`, `all_of()`/`any_of()`, `across()`). If empty, all grouping is removed.

Details

Grouping is tracked as browser-side state consumed by group-aware verbs such as `summarise()` and the `slice()` family. `ungroup()` appends a step that clears or trims that state at compute time.

Value

A `tbl_lazy_json` object with the grouping state cleared (or trimmed when columns are supplied), appended as a lazy compute step.

Examples

```
if (interactive()) {
  tbl(session, "mtcars") |>
    group_by(cyl) |>
    ungroup()
}
```

Index