`rserve-ts`: a `TypeScript` interface to `Rserve` for modern web apps

Tom Elliott

iNZight Analytics Ltd

Iwi affiliations: Ngāti Whakaue, Ngāti Raukawa

Abstract

1 Introduction

The R programming language is a powerful tool for statistical computing and data analysis. As web applications become increasingly prevalent, there is a growing need to integrate R's capabilities into modern web applications. The `Rserve` package (Urbanek, 2003) provides a way for programs to utilize R's facilities through a client-server architecture. While various clients exist for different programming languages, the JavaScript ecosystem has relied on `rserve-js`, which, despite its stability, has not been significantly updated since 2018 and uses outdated JavaScript patterns.

This paper describes the development of `rserve-ts`, a modern TypeScript interface to `Rserve` designed for contemporary web applications. The library wraps `rserve-js` while providing a more modern interface that leverages promises for asynchronous code and implements comprehensive type-checking for data returned from R. These improvements ensure runtime safety and enhance the development experience for front-end developers, making it easier to outsource front-end development to non-R developers.

2 Background & Motivation

2.1 The R-to-Web Landscape

2.2 TypeScript in Modern Web Development

TypeScript has become increasingly important in modern web development, with usage growing from 12% of developers in 2017 to 34% in 2023 (JetBrains, 2023). This growth reflects the language's ability to provide type safety for JavaScript applications, making development, extension, and debugging significantly easier. Type definitions not only catch potential runtime errors during development but also enable powerful development features like autocomplete.

2.3 Limitations of Existing Solutions

The `rserve-js` package, while stable and actively used in projects like RCloud, employs outdated JavaScript coding patterns including:

- Use of instantaneously invoked function expressions (IIFEs)
- Lack of type safety
- Callback-based asynchronous code instead of modern promises
- Limited developer tooling support

These limitations make it increasingly difficult to integrate with modern web development workflows and tools.

3 Core Design of rserve-ts

3.1 Architecture Overview

The `rserve-ts` library operates in two modes:

1. Standard evaluation: Users can send arbitrary R code for evaluation
2. Object capabilities (Ocaps): Results are automatically converted to JSON objects

The library's application programming interface (API) is based on the JSON representation of R objects, modifying the standard evaluation mode to consistently return JSON objects for a more uniform interface.

3.2 Modernizing Asynchronous Operations

[Content from section 2.1 about callbacks vs promises]

3.3 Type System Fundamentals

In R, all objects are vectors of a certain type. The translation of these types to JavaScript/TypeScript presents several challenges:

3.3.1 Scalar vs Vector handling

- R vectors of length 1 without attributes are converted to scalar values
- All other cases return objects with additional properties (`r_type` and optionally `r_attributes`)

3.3.2 Schema Validation

The library uses the `zod` library to define and validate object schemas, ensuring runtime type safety. For example:

```
const intSchema = z.custom((data) => {
  if (data instanceof Int32Array) {
    return data.hasOwnProperty('r_type') &&
           data.r_type === 'int_array';
  }
  return false;
});
```

[Continue with remaining sections...]

4 Type System Implementation

[Content from sections 4.1-4.3]

5 Developer Experience

6 Future Work

7 Conclusion

Acknowledgements

Supported by the Ngā Puanga Pūtaiao Fellowships from Government funding, administered by the Royal Society Te Apārangi.

Acronyms

API application programming interface. 1, 3, 5, 6

GUI graphical user interface. 4

IIFE instantaneously invoked function expression. 3

UX user experience. 11

References

JetBrains. (2023). *The state of developer ecosystem in 2023 infographic.* https://www.jetbrains.com/lp/devecosystem-2023/

Urbanek, S. (2003). Rserve – A Fast Way to Provide R Functionality to Applications. *Friedrich Leisch & Achim Zeileis.*