rplaywright : Bridging Playwright to R for Reproducible Web Automation in Research

Erika Siregar

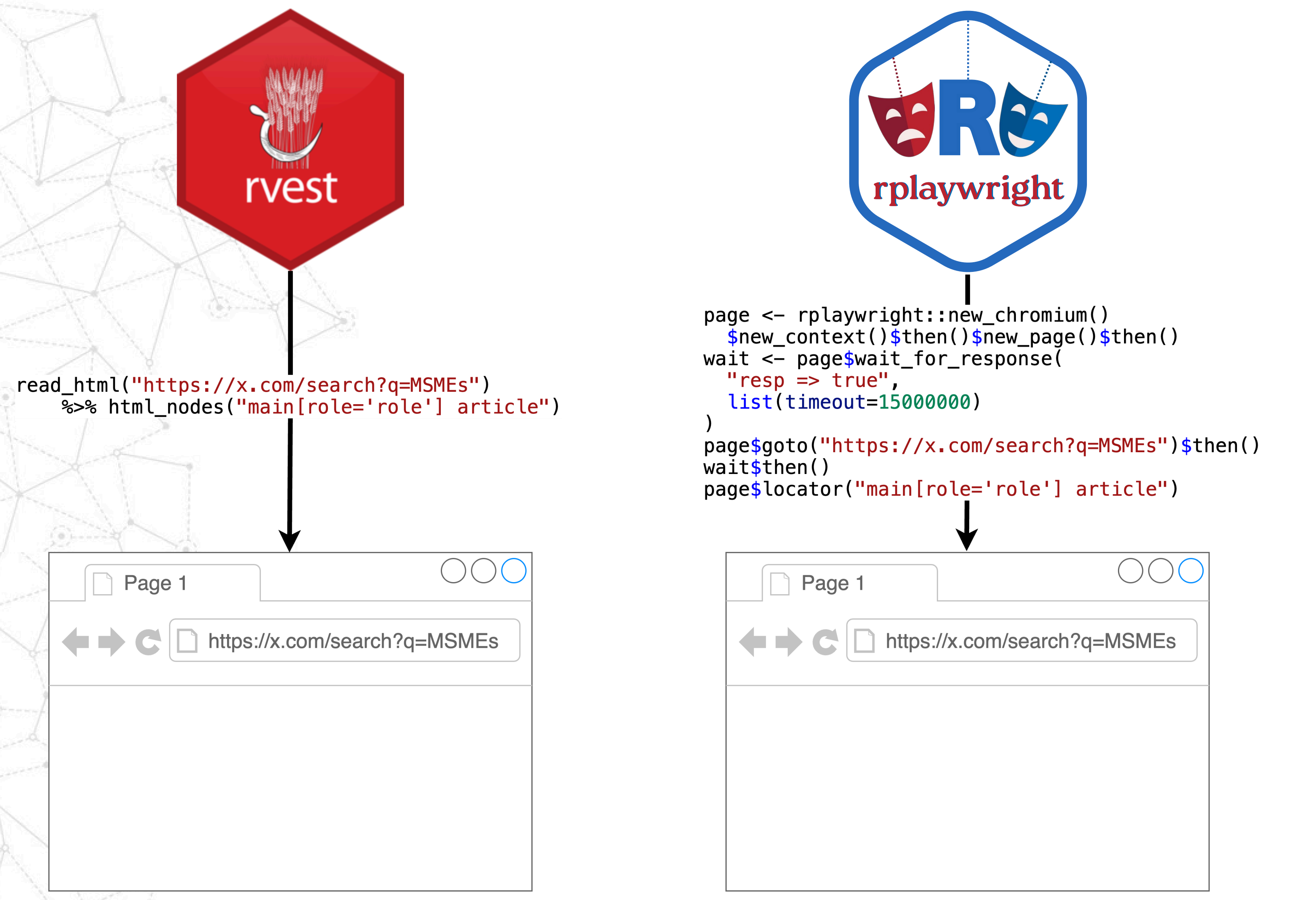
University of Sheffield
emsiregar1@sheffield.ac.uk

WHAT

rplaywright is an R package that bridges R with the Node.js **Playwright** library. It enables R users to perform simplified web testing and automation. The package was developed through the **rOpenSci Champions Program**.

WHY

Traditional scraping tools often struggle with dynamic, JavaScript-heavy websites. **rplaywright** solves this by waiting for page elements and events and interacting directly with the browser Document Object Model (DOM), bringing modern web automation to R.

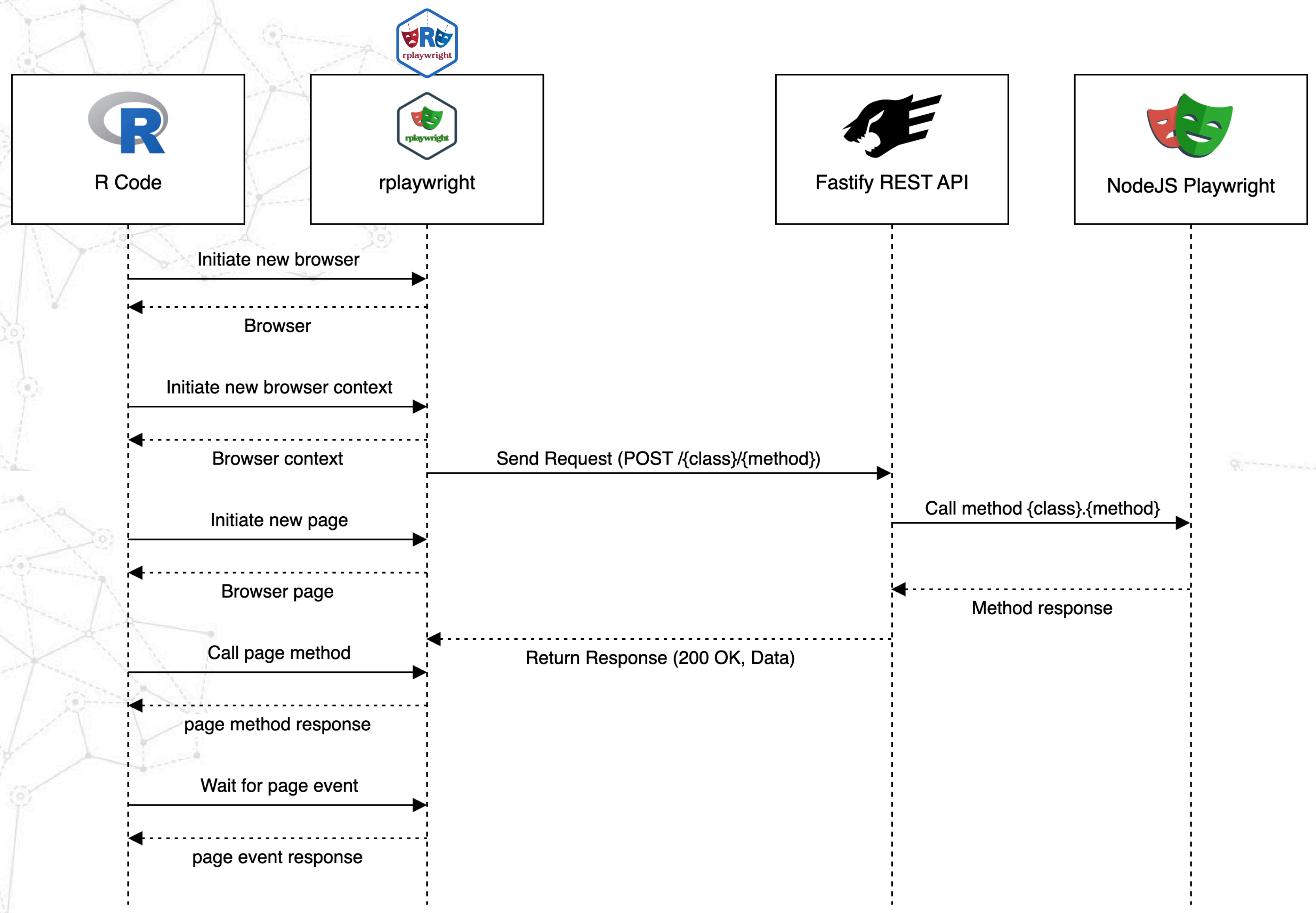


```
read_html("https://x.com/search?q=MSMEs")
    %>% html_nodes("main[role='role'] article")
```

```
page <- rplaywright::new_chromium()
  $new_context()$then()$new_page()$then()
wait <- page$wait_for_response(
  "resp => true",
  list(timeout=15000000)
)
page$goto("https://x.com/search?q=MSMEs")$then()
wait$then()
page$locator("main[role='role'] article")
```

RVest	RPlaywright
✓ Without Browser	✓ Cross browser: Chromium, Firefox, Webkit
✓ Static web page only	✓ Static, dynamic, lazily loaded web page
✓ No wait for mechanism	✓ Wait for request, response, locator, event
✓ No cookie injection	✓ Cookie injection
✓ No script injection	✓ Script injection
✓ No script evaluation	✓ Script evaluation, e.g. scroll to bottom to load more
✓ No browser context	✓ Multiple browser context
✓ Single page	✓ Multiple page

How rplaywright works:



R Code

rplaywright

Fastify REST API

NodeJS Playwright

Initiate new browser

Browser

Initiate new browser context

Browser context

Send Request (POST /{class}/{method})

Call method {class}.{method}

Initiate new page

Browser page

Method response

Return Response (200 OK, Data)

Call page method

page method response

Wait for page event

page event response

Coverage : **11** classes **186** methods

Contribute:

Use Cases:

*For demonstration purposes only; always respect robots.txt when web scraping.

Scrape Static Website

Static website

rplaywright can also scrape static websites using methods such as *locator, get_by_alt_text, get_by_label, get_by_placeholder, get_by_role, get_by_test_id, get_by_text,* and *get_by_title*. **Scan QR Code for example.**

Fill Form

Fill Form

rplaywright can automate form interactions, such as registration, login, or surveys. Methods include *locator, blur, check, clear, click, dblclick, fill, focus, hover, press, select_option*, and more. **Scan QR Code for example.**

Scrap X Posts

Scraping X post

rplaywright can handle complex dynamic websites, interacting with the DOM and network. In this example, a cookie is injected into the browser context, and posts are captured from network responses while scrolling to the bottom. **Scan QR Code for example.**

Limitations:

- ✓ **Node.js Dependency** – rplaywright relies on a Node.js-based API, which increases installation size.
- ✓ **Browser Engines** – rplaywright downloads dedicated builds (its own versions) of Chromium, Firefox, and WebKit. This ensures reproducibility but adds to the package footprint.
- ✓ **Workflow Integration** – rplaywright outputs are not yet dataframes, so a short conversion is needed before using **%>%** or **|>**. Once structured, they integrate smoothly with packages like dplyr, tidyr, and ggplot2.

Future Works:

- ✓ **Expanded API Coverage** – rplaywright does not yet implement all classes and methods of the Node.js Playwright API. Ongoing development and contributions will extend its functionality.
- ✓ **Enhanced R Integration** – Compatibility with both magrittr pipe (**%>%**) and native pipe (**|>**) is planned to improve workflow fluency. Additional work is also needed to handle methods with identical names by introducing clearer input differentiation.