Javascript ⇄ Shiny

Integrating Shiny and Javascript

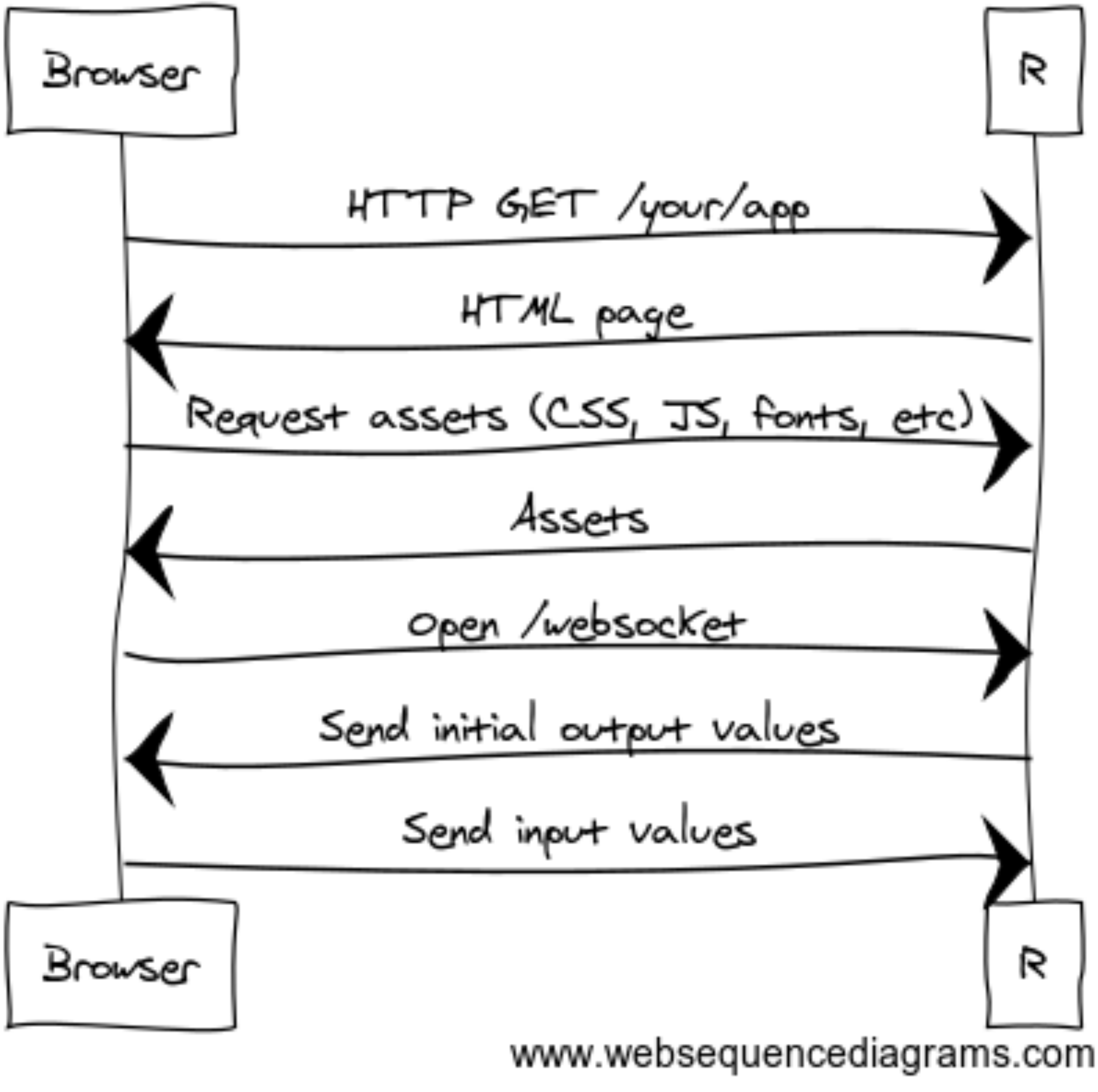
Alan Dipert

OCRUG, 11/27/2018

Topics

- How does Shiny work?
- Inputs
- Outputs
- htmlwidgets
- React.js
- Where to learn more

High level flow


Browser
R
HTTP GET /your/app
HTML page
Request assets (CSS, JS, fonts, etc)
Assets
Open /websocket
Send initial output values
Send input values
Browser
R
www.websequencediagrams.com

What’s a WebSocket?

- Originally (< 2000), HTTP clients (browsers) had to initiate activity on the server by sending requests
    - Server had no natural way to initiate activity on the client.
- In order for the server to initiate activity on the client, the client would have to periodically send a request (short-polling)
- Other workarounds were developed (Comet, long-polling)
- WebSockets are an addition to the HTTP protocol that standardize a persistent connection between client and server. WebSockets are widely supported and workarounds increasingly unnecessary.

Inputs and Outputs

- Shiny ships with many components, sometimes called widgets
    - Plots, sliders, text inputs, date pickers…
- Client: arbitrary JS, register with Shiny on page load. Receive data from the server and present it (output), or send data to the server (input)
- Server: manage options, produce initial HTML, send data to the client (output)

Inputs in detail

Exploration in RStudio

https://shiny.rstudio.com/articles/building-inputs.html

Input instantiation

1. `ui` function produces initial HTML, sends to browser
    1. Sets class or other attributes to make “placeholder” discoverable
2. Page load: Javascript finds every placeholder on page, attaches event handlers to them
3. Inputs registered
4. Shiny loads

Clicking the button

1. User clicks button

2. “click” handler fires, parses button text as integer, increments, triggers “change” event

3. “change.incrementBinding” handler fires, invoking callback()

4. callback() calls the input’s getValue function to obtain its value

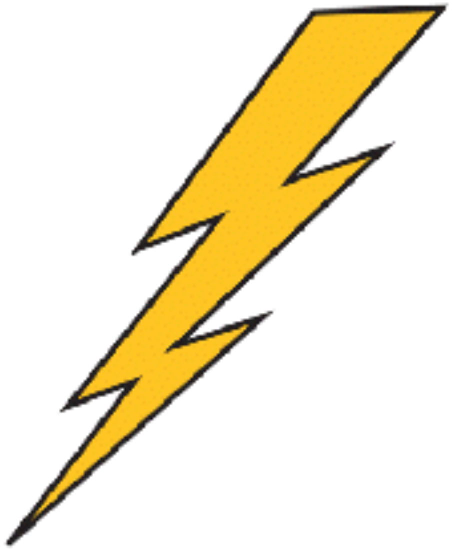
Outputs in detail

- Instantiation similar to inputs
- Two big differences in operation
    - Flow of data after instantiation is inverted (duh)
    - Server-Side Output Functions like `renderPlot`, `renderTable`

Outputs in detail

Exploration in RStudio

https://shiny.rstudio.com/articles/building-outputs.html



And Then There Were htmlwidgets

htmlwidgets

- You’ve used them: Leaflet, dygraphs, Plotly, DiagrammeR…
- Announced 2014 (Shiny announced in 2012)
- More capable than inputs or outputs
  - Share data without server (Crosstalk)
  - …and work offline
  - …and appear in RMarkdown
- *Probably the highest leverage way to expose Javascript functionality to R*

What does React.js do?

- Novel way to project data into the DOM
- Browser’s native DOM is a “retained mode” graphics interface
  - Graph of stateful objects
  - Efficient use requires management of DOM nodes
- React.js presents an “immediate mode” interface
  - Sits atop the DOM
  - Simulates re-drawing the whole page every time an incremental change occurs by intelligently “patching” DOM

Learning More

- *Javascript: The Good Parts* by Douglas Crockford
- Mozilla Javascript Guide: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction
- Shiny custom input, output docs
- htmlwidgets.org
- reactjs.org