React

Excelsior!

By Stephen Lampa & Jesse Williamson

Today's Topics

- Requests
  - Promises
- Common libraries
- Live coding?

Requests

Promises

Promises

- Definition: The Promise object represents the eventual completion (or failure) of an asynchronous operation, and its resulting value. (MDN)
- In one of three states:
    - Pending
    - Fulfilled
    - Rejected

Promises

- Scenario: When your friend assures you that you will both go to the board game meetup together.
    - Pending: Riding the bus there
    - Fulfilled: If you and your friend go to the meetup together
    - Rejected: Friend doesn’t show up and you’re alone for the rest of your life

Promises

- Scenario: When your friends assure you that they will go to the board game meetup with you.
    - Pending: Riding the bus there
    - Fulfilled: If a friend goes to the meetup
    - Rejected: Friends don’t show up and you’re alone for the rest of your life

Promises

Two main methods to that return a Promise:

- resolve($someValue)
- reject($someValue)

Note: $someValue can be whatever object you want to pass along

Promises

- To actually handle the resolve we use:
  - .then(someSuccessFunc, someFailFunc);
  - .catch(someFailFunc);
- .then() => chainable! Each then returns something wrapped in a Promise

Network Requests
(No, not LinkedIn)
Or, Web API Requests
Or, HTTP Requests

Requests

Three main components:

- Line: Specifies the resource you want to get
- Headers: Metadata around the request
- Body: Additional items that the API may be expecting

Requests - Line

Generally follows:

GET https://auth.test.rflx.com/box

- Has a verb (GET, POST, PATCH, PUT, DELETE)
- Has a URI
- Can contain other resources
- There's more to this, but this is what you'll generally encounter with libraries

Requests - HTTP Verbs

- GET: Read (retrieve) data. Sometimes adds query strings (API dependent)
  - /box?value=true&token=12sAcvXC35
- POST: Create data. Generally uses the body part of the request (will get into that)
- PUT/PATCH: Update data. Difference between updating whole or part
- DELETE: Deletes data.

Requests - Headers

- As mentioned, kind of the metadata surrounding the request
- Responsible for things like how the client is sending data and/or how the client wants to receive data
- There are lots but here some of the most commonly interacted with ones...

Requests - Common Headers

- Content-Type: Specifies the type of content the client is sending (ex Content-Type: application/json)
- Accept: Specifies the type of content the client expects back (ex Accept: application/json)
- Authorization: Generally some token that says the client is authorized to get a resource
  - Or a specific security one specified by the API

Requests - Body

- Is the actual data sent to the API
- Specified by the Content-Type
- The API specifies what it is actually looking for
- Good APIs should ignore the bad stuff...

Requests - Response

- Important things to look for when getting a response...

Requests - Response Status

- Status Code: A code that the API sends back telling the client whether the action was successful or something else
- 200s - mean success
- 300s - redirects
- 400s - bad request
- 500s - server error

Requests - Common 200

- 200 OK - Associated with a good GET call
- 201 Created - Associated with a good POST call. Means a resource was… created

Requests - Common 300s

- **301 Moved Permanently - Says that the requested URL has moved to a new URL PERMANENTLY (can’t stress this enough)**

Requests - Common 400s

- 400 Bad Request - You messed up (catch all). More like the something is generally wrong with body of the request
- 401 Unauthorized - Not authenticated
- 403 Forbidden - Not authorized
- 404 Not Found - Doesn't look like anything to me
- 409 Conflict - Server was expecting something else (like content-type)
- 418 I'm a teapot - Teapot
- 422 Unprocessable Entity - The data and format looks good, but fails validations

Requests - Common 500s

- 500 Internal Server Error - Where you walk over to the Cloud team and talk to Stephen
- 503 Service Unavailable - Either the client or server is offline
- 504 Gateway Timeout - Basically telling you what the name implies. Requested never succeeded because it waited its specified time

Requests - Return Body

- Obviously the most important part…
- Nothing much to this as long as your Accept header specifies what you expect

How git brought fetch back (it did, but git is irrelevant to this current topic...)

Fetch

- Comes standard with JavaScript (not Node)
- Returns a Promise
- Able to specify the request line, header, body, etc
- Fairly straightforward
- Will be using a simple API - https://jsonplaceholder.typicode.com/

Common Libraries

Routing

- Routing: How you generally navigate through pages based on the URL
- Useful for users to bookmark pages or make use of the backward/forward buttons

react-router

https://github.com/ReactTraining/react-router

API Calls

- The new hotness: Axios
- Better return object to work with! Don't have to do `` `response.json()` ``
- Better error handling

Axios
https://github.com/axios/axios

State Management

- Things can get complicated with many more views in containers
- Global way to get/store state of components
- State changes predictably. Easy traceability

redux

https://github.com/reduxjs/redux

MobX

https://github.com/mobxjs/mobx

Handling side effects

- **Side effects: something that happens surrounding that is not part of the original functionality**

Redux-saga

https://github.com/redux-saga/redux-saga

Redux-thunk

https://github.com/reduxjs/redux-thunk

Testing

- Testing framework: Jest
- Component Helper: react-test-renderer
- Component Helper: enzyme

Jest

https://github.com/facebook/jest

Enzyme

https://github.com/airbnb/enzyme

Live Coding?

Questions?