Lecture 7: Data

Jordan Hayashi

Previous Lecture

- `react-navigation`
- `SwitchNavigator`
- `navigation` prop
- `StackNavigator`
- Configuring navigators
- `TabNavigator`
- Composing navigators

Data

- Not all apps are self-contained
- Any app that wants to rely on information not computed within the app needs to get it from somewhere
  - Communicate with other resources using an API

API

- “Application Programming Interface”
- A defined set of ways with which a resource can be interacted
  - React components have APIs; you interact by passing props
  - A class has an API; you interact by invoking methods
  - A web service has an API; you interact by making network requests
- Providers often get to decide on the API, but sometimes it’s decided for them
- Consumers have to read docs to know how to use an API

https://randomuser.me/documentation

Making Network Requests

- `fetch()` is polyfilled
  - It's not natively part of JavaScript, but it is implemented to match the usage of the browser `fetch()`
- `fetch()` expects an URL and optionally some config
- `fetch()` returns a `Promise`, which is fulfilled with a `Response` object

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

https://developer.mozilla.org/en-US/docs/Web/API/Response

Promises

- Allows writing asynchronous, non-blocking code
- Allows chaining callbacks and/or error handlers
  - `.then()` - executed after the previous Promise block returns
  - `.catch()` - executed if the previous Promise block errors

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

Async/Await

- Allows writing async code as if it were synchronous
  - Still non-blocking
- A function can be marked as `async`, and it will return a `Promise`
- Within an `async` function, you can `await` the value of another async function or `Promise`
- Use `try/catch` to handle errors

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await

Transforming Data

- Sometimes the shape of the data returned by an API isn’t ideal
  - Where should we do this “transformation?”
- Doing it early gives us an abstraction barrier and is more efficient

Authentication

- A process to determine if a user is who they say they are
- Generally done using a name and password
- But how do we send the name and password in the request?

HTTP Methods

- GET
  - The default in browsers and in `fetch()`
  - Add parameters in the url by appending a `?` and chaining `key=value` pairs separated by &
- POST
  - Submit data (e.g. a form) to an endpoint
  - Parameters are included in the request body
  - If POSTing JSON, must have `content-type: application/json` header and body must be JSON string

HTTP Response Codes

- Every network response has a “code” associated with it
  - 200: OK
  - 400: Bad Request
  - 403: Forbidden
  - 404: Not Found
  - 500: Internal Server Error
  - 418: I’m a teapot

https://developer.mozilla.org/en-US/docs/Web/HTTP/Status