React Native - Basics - Working with Maps

A brief outline of adding maps and geolocation to a React Native app.

This includes details for Apple Maps with iOS, and Google Maps with Android and iOS.

Contents

Intro

To add maps and geolocation to a React Native based application, we need to consider the following options,

- adding a map view
- defining an initial location for the map view
- adding geolocation option to use device GPS, network &c. location
- update the map view to render the location

We may also consider various options for how geolocation and a map view may work together in an application. For example,

- is the map view implicitly updated as the user's location is updated...
- can the user manually check and update their location and map view...

Create initial app

Create a basic React Native app using the usual command,

```
```

```
react-native init TestMaps
```

install React Native Maps

Then, we need to install the community package, `react-native-maps`, for the map view in the app.

In the CWD, use NPM to install the required package

```
npm install react-native-maps --save
```

This package will install a library to support rendering and general usage with map views on both Android and iOS.

For iOS, the default map view supports Apple Maps. For Android, we use Google Maps.

It is also possible to configure iOS support for Google Maps API.

n.b. this install option currently works OK with Apple Maps. For Google Maps, please install this package from the community Git repository, as detailed in the below *iOS with Google Maps* section.

setup default maps for iOS

In addition to the above package install, we also need to configure native support for our chosen Maps API.

This will link the *React Native Maps* library with the required native platform code. This permits the required native code to be compiled along with the React Native app.

The simplest option for iOS is to simply use the default Apple Maps.

We can run the following command to link the required native code dependencies with React Native for compiling the app,

```
react-native link react-native-maps
```

If you encounter the following error message,

```
...
Error: Cannot find module 'asap/raw'
...
```

you'll need to run

```
npm install
```

to ensure all of the required dependencies are satisfied and saved to the required directory. Then, re-run the above `link` command.

App - Apple Maps

We can now start to build our app with our first maps option, iOS's default **Apple Maps**.

app - add some initial data

We'll start by adding some default data for testing the map view, and marker rendering.

In a new `data.js` file, we might add the following location data

```
export const sites = [
  {
    name: 'Karnak Temple',
    image: '',
    coordinate: [ 25.719595, 32.655807 ],
    visible: true,
  },
  {
    name: 'Luxor Temple',
    image: '',
    coordinate: [ 25.6989, 32.6421 ],
        visible: true,
  },
        {
    name: 'Valley of the Kings',
    image: '',
    coordinate: [ 25.746424, 32.605309 ],
    visible: false,
  },
];
```

This data includes the latitude and longitude coordinates for three locations with additional properties, which we'll use later in the app.

app - add imports

In the `App.js` file, we may define our initial imports for the app

```
// import default React
import React, { Component } from 'react';
// import various components from React Native
import {
  StyleSheet,         // styles
  Text,               // text
  TouchableOpacity,   // touchable container
```

```
  View                // view container component
} from 'react-native';
// import map view from package
import MapView from 'react-native-maps';
// import data from data.js json
import { sites } from './data';
```

app - add a map view with data

With the above data, we may then start to create our map view and render in the app.

We'll start by creating a component for the app's Map view,

```
export default class Map extends Component {
        ...
}
```

and then add the initial `state` for the component,

```
state = {
  // show visible or all sites
  showVisible: false,
}
```

We also need to add the required `render()` method for the component, which we'll use to add a view container,

```
render() {
    return (
      <View style={styles.container}>
        ...
      </View>
    );
  }
```

and a custom component for the map,

```
<MapView
  style={styles.map}
  // initial location for map rendering
  initialRegion={{
        latitude: 25.6989,
        longitude: 32.6421,
        latitudeDelta: 0.0491,
        longitudeDelta: 0.0375,
```

```
    }}
  >
    {sites.map((site, index) =>
      // check state visibility against data item visibility
      <MapView.Marker
        coordinate={{
          latitude: site.coordinate[0],
          longitude: site.coordinate[1],
        }}
        // variant colours for pins
        pinColor={site.visible ? '#6A9662' : '#000000'}
        key={index}
      />
    )}
  </MapView>
```

In this custom map view, we define props for `style` and the map's initial location. Then, we call the ES6 `map()` method with the `sites` object from data to check for visibility of a data item.

For each map view marker, we set props for the data item coordinates, and assign a pin colour relative to the visibility property in the data. A `key` is then set using the `index` value from the `map()` method iterator.

app - basic stylesheet

We may then add an initial stylesheet for the views,

```
// styles for map &c.
const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'flex-end',
    alignItems: 'center',
  },
  map: {
    ...StyleSheet.absoluteFillObject,
  },
  },
});
```

app - build and test iOS app with Apple Maps

Build and test as usual,

```
react-native run-ios
```

App - iOS with Google Maps

We may also test using Google Maps with an iOS app, instead of the default Apple Maps.

We'll start by creating a new test app, `BasicAppGoogleMaps`, and then installing `react-native-maps`.

app - setup `react-native-maps`

As noted above, due to bugs with the current NPM install of `react-native-maps` with Google Maps framework on iOS, we may use the Git repository to install this module.

Update `package.json` with the following Git dependency,

```
...
"dependencies": {
    ...
    "react-native-maps": "https://github.com/react-community/react-native-maps.git"
  },
...
```

Then, update the app's modules for this app

```
npm install
```

We may then link the React Native app with the required native dependencies.

```
react-native link react-native-maps
```

app - update iOS code

Update `AppDelegate.m`, for example using Xcode, and add the following specifics for the Google Maps framework.

```
+ #import <GoogleMaps/GoogleMaps.h>

@implementation AppDelegate
...

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
+  [GMSServices provideAPIKey:@"_YOUR_API_KEY_"]; // add this line using the api key obtained from Google Console
...
```

To work with the Google Maps API, we need to add an API key from the Google Developer Console.

Then we can use the following instructions to add the Google Maps SDK to the iOS native project

- https://developers.google.com/maps/documentation/ios-sdk/start

After setting up the Google Maps framework in iOS, we need to update `package.json` to set a relative path to this framework,

```
"scripts": {
  ...
  "postinstall": "./node_modules/react-native-maps/enable-google-maps
./ios",
},
```

where `./ios` defines the local install directory for the app's Google Maps framework in the local React Native app.

We may then run `npm install` to ensure `postinstall` is completed.

app - update JS for Google Maps on iOS

After adding the above dependencies, we may update the app's JS for Google Maps

```
import MapView, { PROVIDER_GOOGLE } from 'react-native-maps';
  ...
<MapView
  provider={PROVIDER_GOOGLE}
  style={styles.map}
  ...
>
```

To test the app with Google Maps, we may use the JavaScript code for `App.js` and `data.js`, as detailed in the above Apple Maps example.

app - build and test iOS app with Google Maps

We may then build the app as usual to test the app works as expected,

```
react-native run-ios
```

App - Android with Google Maps

As Google Maps is the default maps application for Android, setup is considerably easier than iOS.

app - enable SDK for Android

On the Google Developer Console, enable the Maps SDK for Android,

- https://developers.google.com/maps/documentation/android-sdk/signup

This API key is needed to allow Google Maps to run and show in your custom app.

app - setup Google Maps for Android

We may then use the following outline for configuring Android,

- https://github.com/react-community/react-native-maps/blob/HEAD/docs/installation.md#build-configuration-on-android

This outline includes necessary updates for the Android build within the React Native app.

app - build and test Android app

To build and test our new maps app with a local Android emulator, it may be necessary to start the emulator before running the React Native app.

We may check available devices using the following command,

```
emulator -list-avds
```

and then run a specific emulator as follows,

```
emulator -avd Pixel_2_XL_API_27
```

specifying the name of the installed and configured emulator.

With the emulator now running, we may build the React Native app using the standard command,

```
react-native run-android
```

References

- React Native Maps - https://github.com/react-community/react-native-maps
- Google Maps iOS SDK - https://developers.google.com/maps/documentation/ios-sdk/start
- Google Maps Android SDK - https://developers.google.com/maps/documentation/android-sdk/signup