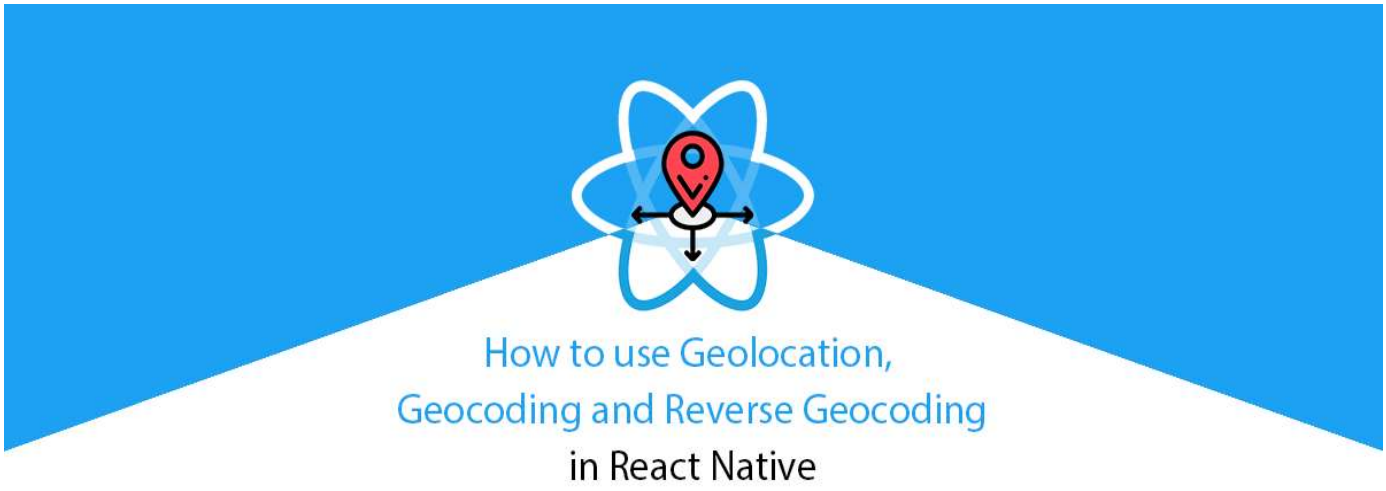

How to use Geolocation, Geocoding and Reverse Geocoding in React Native Apps

In this post, you will learn how to implement Geolocation React native apps. We will also learn how to Convert Geocode in Location address (Reverse Geocoding) and Location Address into Geocode(Geocoding) in a simple React native app.

React Native lets you build mobile apps using only JavaScript. It uses the same design as React, letting you compose a rich mobile UI using declarative components.

What Is React Native?

React Native is a JavaScript framework for writing real, natively rendering mobile applications for iOS and Android. It's based on React, Facebook's JavaScript library for building user interfaces, but instead of targeting the browser, it targets mobile platforms. In other words: web developers can now write mobile applications that look and feel truly "native," all from the comfort of a JavaScript library that we already know and love. Plus, because most of the code you write can be shared between platforms, React Native makes it easy to simultaneously develop for both Android and iOS.

Similar to React for the Web, React Native applications are written using a mixture of JavaScript and XML markup, known as JSX. Then, under the hood, the React Native "bridge" invokes the native rendering APIs in Objective-C (for iOS) or Java (for Android). Thus, your application will render using real mobile UI components, not webviews, and will look and feel like any other mobile application. React Native also exposes JavaScript interfaces for platform APIs, so your React Native apps can access platform features like the phone camera, or the user's location.

React Native currently supports both iOS and Android and has the potential to expand to future platforms as well. In this blog, we'll cover both iOS and Android. The vast majority of the code we write will be cross-platform. And yes: you can really use React Native to build production-ready mobile applications! Some example: Facebook, Palantir, and TaskRabbit are already using it in production for user-facing applications.

What is Geolocation?

The most famous and familiar location feature—Geolocation is the ability to track a device's whereabouts using GPS, cell phone towers, WiFi access points or a combination of these. Since devices are used by individuals, geolocation uses positioning systems to track an individual's whereabouts down to latitude and longitude coordinates, or more practically, a physical address. Both mobile and desktop devices can use geolocation. Geolocation can be used to determine time zone and exact positioning coordinates, such as for tracking wildlife or cargo shipments.

Some famous apps using Geolocation are

- Uber / Lyft—Cab booking
- Google Maps (of course)—Map services
- Swiggy / Zomato—Food delivery
- Fitbit—Fitness app
- Instagram / Facebook—For tagging photos

What is Geocoding and Reverse geocoding?

Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate.
Reverse geocoding is the process of transforming a (latitude, longitude) coordinate into a (partial) address. The amount of detail in a reverse geocoded location description may vary, for example, one might contain the full street address of the closest building, while another might contain only a city name and postal code.

Post structure

We will go in a step-by-step manner to explore the anonymous login feature of Firebase. This is my break-down of the blog

STEPS

1. Create a simple React Native app
2. Install Plugins for Geocoding and Geolocation and get User Location
3. Get User Current Location (Geocoding)
4. Convert User Geolocation into an address (Reverse Geocoding)
5. Convert User Entered Address into Geocode (Geocoding)

We have three major objectives

1. Get User Current Location which we will get in latitude and longitude (Geolocation)
2. Convert that latitude and longitude in Street Address (Reverse Geocoding)
3. And again convert Street address entered by the user into latitude and longitude (Geocoding)

Let's dive right in!

Step 1:—Create a simple React Native app

If you are coming from a web background, the easiest way to get started with React Native is with Expo tools because they allow you to start a project without installing and configuring Xcode or Android Studio. Expo CLI sets up a development environment on your local machine and you can begin writing a React Native app within minutes. For instant development, you can use Snack to try React Native out directly in your web browser.

If you are familiar with native development, you will likely want to use React Native CLI. It requires Xcode or Android Studio to get started. If you already have one of these tools installed, you should be able to get up and running within a few minutes. If they are not installed, you should expect to spend about an hour installing and configuring them.

Expo CLI Quickstart

Assuming that you have Node 10+ installed, you can use npm to install the Expo CLI command-line utility:

```
npm install -g expo-cli
```

Then run the following commands to create a new React Native project called "AwesomeProject":

```
expo init AwesomeProject
```

```
cd AwesomeProject
npm start # you can also use: expo start
```

This will start a development server for you.

React Native CLI Quickstart

Assuming that you have Node 10+ installed, you can use npm to install the React Native CLI command-line utility:

```
npm install -g react-native-cli
```

Then run the following commands to create a new React Native project called "AwesomeProject":

```
react-native init AwesomeProject
```

> *For details you can check this link.*

Step 2 :— Install Plugins for Geocoding and Geolocation and get User Location

> *In Bare React Native Apps*

react-native-geolocation-service

This library is created in an attempt to fix the location timeout issue on android with the react-native's current implementation of Geolocation API. This library tries to solve the issue by using Google Play Service's new `FusedLocationProviderClient` API, which Google strongly recommends over android's default framework location API. It automatically decides which provider to use based on your request configuration and also prompts you to change the location mode if it doesn't satisfy your current request configuration.

> NOTE: Location request can still timeout since many android devices have GPS issue in the hardware level or in the system software level. Check the FAQ for more details.

Installation

yarn

```
yarn add react-native-geolocation-service
```

npm

```
npm install react-native-geolocation-service
```

Setup

Android

No additional setup is required for Android

iOS

You need to include the `NSLocationWhenInUseUsageDescription` key in Info.plist to enable geolocation when using the app. In order to enable geolocation in the background, you need to include the `NSLocationAlwaysUsageDescription` key in Info.plist and add location as a background mode in the 'Capabilities' tab in Xcode.

- Update your `Podfile`

```
pod 'react-native-geolocation', path: '../node_modules/@react-native-community/geolocation'
```

- Then run `pod install` from ios directory

> *In EXPO managed apps,*

you'll need to run

```
expo install expo-location
```

Step 3:— Get User Current Location (Geocoding)

> *In Bare React Native Apps,*

Since this library was meant to be a drop-in replacement for the RN's Geolocation API, the usage is pretty straight forward, with some extra error cases to handle.

> *One thing to note, this library assumes that location permission is already granted by the user, so you have to use* `PermissionsAndroid` *to request for permission before making the location request.*

For getting user location you have to import `Geolocation` API from `react-native-geolocation-service` package like this

```
import Geolocation from 'react-native-geolocation-service';
```

And add this function into your code

```
if (hasLocationPermission) {
Geolocation.getCurrentPosition(
(position) => {
console.log(position);
},
(error) => {
// See error code charts below.
console.log(error.code, error.message);
},
{ enableHighAccuracy: true, timeout: 15000, maximumAge:          10000 }
);
}
```

So after adding all this code your file something look like this.

```
...
import Geolocation from 'react-native-geolocation-service';
...

componentDidMount() {
    // Instead of navigator.geolocation, just use Geolocation.
    if (hasLocationPermission) {
        Geolocation.getCurrentPosition(
            (position) => {
                console.log(position);
            },
            (error) => {
                // See error code charts below.
                console.log(error.code, error.message);
            },
            { enableHighAccuracy: true, timeout: 15000, maximumAge: 10000 }
        );
    }
}
```

getUserLocation.ts hosted with ❤ by GitHub view raw

> ***In EXPO [managed](#) apps,***

For getting user location you have to import `Location` API from `expo-location` package like this

```
import * as Location from 'expo-location';
```

And add this function into your code

```
_getLocationAsync = async () => {
let { status } = await Permissions.askAsync(Permissions.LOCATION);
if (status !== 'granted') {
this.setState({
errorMessage: 'Permission to access location was denied',
});
}
```

```
let location = await Location.getCurrentPositionAsync({});
this.setState({ location });
};
```

So after adding all this code your file something look like this.

> *One thing to note, this library assumes that location permission is already granted by the user, so you have to use* `Permissions from 'expo-permissions'` *to request for permission before making the location request.*

```
import React, { Component } from 'react';
import { Platform, Text, View, StyleSheet } from 'react-native';
import Constants from 'expo-constants';
import * as Location from 'expo-location';
import * as Permissions from 'expo-permissions';

export default class App extends Component {
  state = {
    location: null,
    errorMessage: null,
  };

  componentWillMount() {
    if (Platform.OS === 'android' && !Constants.isDevice) {
      this.setState({
        errorMessage: 'Oops, this will not work on Sketch in an Android emulator. Try it on your dev
      });
    } else {
      this._getLocationAsync();
    }
  }

  _getLocationAsync = async () => {
    let { status } = await Permissions.askAsync(Permissions.LOCATION);
    if (status !== 'granted') {
      this.setState({
        errorMessage: 'Permission to access location was denied',
      });
    }

    let location = await Location.getCurrentPositionAsync({});
    this.setState({ location });
  };

  render() {
    let text = 'Waiting..';
    if (this.state.errorMessage) {
      text = this.state.errorMessage;
    } else if (this.state.location) {
      text = JSON.stringify(this.state.location);
    }

    return (
      <View style={styles.container}>
        <Text style={styles.paragraph}>{text}</Text>
      </View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
    paddingTop: Constants.statusBarHeight,
    backgroundColor: '#ecf0f1',
  },
  paragraph: {
    margin: 24,
    fontSize: 18,
    textAlign: 'center',
  },
});
```

GeoLocationExpo.js hosted with ❤ by GitHub view raw

Step 4:—Convert User Geolocation into an address (Reverse Geocoding)

> *In Bare React Native Apps,*

In this app, we are using `react-native-geocoding` package for **Geocoding** and **Reverse Geocoding**

react-native-geocoding

A geocoding module for React Native to transform a description of a location (i.e. street address, town name, etc.) into geographic coordinates (i.e. latitude and longitude) and vice versa.

This module uses the Google Maps Geocoding API and requires an API key for purposes of quota management. Please check this link out to obtain your API key.

Install

```
npm install --save react-native-geocoding
```

For Geocoding and Reverse Geocoding in your app you import `Geocoder` API from `react-native-geocoding` package like this

```
import Geocoder from 'react-native-geocoding';
```

And Then you have to initialize the module in your app(needs to be done only once)

```
Geocoder.init("xxxxxxxxxxxxxxxxxxxxxxxx"); // use a valid API key
// With more options
// Geocoder.init("xxxxxxxxxxxxxxxxxxxxxxxx", {language : "en"}); // set the language
```

And use this code respectively for Geocoding and reverse Geocoding

```
Geocoder.from(41.89, 12.49)
                .then(json => {
                var addressComponent = json.results[0].address_components[0];
                        console.log(addressComponent);
                })
                .catch(error => console.warn(error));
```

```
// Works as well :
// ------------

// location object
Geocoder.from({
        latitude : 41.89,
        longitude : 12.49
});

// latlng object
Geocoder.from({
        lat : 41.89,
        lng : 12.49
});

// array
Geocoder.from([41.89, 12.49]);
```

Error Codes

Search this file...

Name	Code	Description
NOT_INITIATED	0	Module hasn't been initiated. Call init function and pass it your app's api key as parameter.
INVALID_PARAMETERS	1	Parameters are invalid.
FETCHING	2	Error wile fetching to server. The error's 'origin' property contains the fetch error.
PARSING	3	Error while parsing server response. The error's 'origin' property contains the response.
SERVER	4	Error from the server. The error's 'origin' property contains the response's body.

ErrorCodes.csv hosted with ❤ by GitHub view raw

> ***In EXPO [managed](#) apps,***

For geocoding of Address we are using `react-native-geocoding` so the installation and Initialize Steps are the same as Geolocation.

we will use this code for Reverse Geocoding

```
_attemptReverseGeocodeAsync = async () => {
this.setState({ inProgress: true });
try {
let result = await Location.reverseGeocodeAsync(
this.state.selectedExample
);
this.setState({ result });
} catch (e) {
this.setState({ error: e });
} finally {
this.setState({ inProgress: false });
}
};
```

Step 5:—Convert User Entered Address into Geocode (Geocoding)

For Reverse geocoding of Address, we are using `expo-location` so the installation and Initialize Steps are the same as Reverse Geocoding.

we will use this code for Geocoding

```
Geocoder.from("Colosseum")
                .then(json => {
                        var location = json.results[0].geometry.location;
                        console.log(location);
                })
                .catch(error => console.warn(error));
```

> ***In EXPO [managed](#) apps,***

For geocoding of Address we are using `react-native-geocoding` so the installation and Initialize Steps are the same as Geolocation.

we will use this code for Geocoding

```
_attemptGeocodeAsync = async () => {
this.setState({ inProgress: true, error: null });
try {
let result = await Location.geocodeAsync(this.state.selectedExample);
this.setState({ result });
} catch (e) {
this.setState({ error: e.message });
} finally {
this.setState({ inProgress: false });
}
};
```

Working Example of Geocode and Reverse Geocing in Your App

```
import React from 'react';
import { ActivityIndicator, Text, Button, Platform, StyleSheet, View } from 'react-native';
import Touchable from 'react-native-platform-touchable';

import { Permissions, Location } from 'expo';

const EXAMPLES = [
  '1 Hacker Way',
  { latitude: 49.28, longitude: -123.12 },
  'Palo Alto Caltrain Station (this one will error)',
  'Rogers Arena, Vancouver',
  { latitude: 0, longitude: 0 },
];

export default class GeocodingScreen extends React.Component {
  static navigationOptions = {
    title: 'Geocoding',
  };

  state = {
    selectedExample: EXAMPLES[0],
    result: '',
    inProgress: false,
  };

  componentDidMount() {
    Permissions.askAsync(Permissions.LOCATION);
  }

  render() {
    let { selectedExample } = this.state;

    return (
      <View style={styles.container}>
        <View style={styles.headerContainer}>
          <Text style={styles.headerText}>Select a location</Text>
        </View>

        <View style={styles.examplesContainer}>
          {EXAMPLES.map(this._renderExample)}
        </View>

        <View style={styles.separator} />

        <View style={styles.actionContainer}>
          <Button
            onPress={this._attemptGeocodeAsync}
            title="Geocode"
            disabled={typeof selectedExample !== 'string'}
            style={styles.button}
          />
          <Button
            onPress={this._attemptReverseGeocodeAsync}
            title="Reverse Geocode"
            disabled={typeof selectedExample !== 'object'}
            style={styles.button}
          />
        </View>

        <View style={styles.separator} />

        {this._maybeRenderResult()}
      </View>
    );
  }
```

```
66
67    _attemptReverseGeocodeAsync = async () => {
68      this.setState({ inProgress: true });
69      try {
70        let result = await Location.reverseGeocodeAsync(
71          this.state.selectedExample
72        );
73        this.setState({ result });
74      } catch (e) {
75        this.setState({ error: e });
76      } finally {
77        this.setState({ inProgress: false });
78      }
79    };
80
81    _attemptGeocodeAsync = async () => {
82      this.setState({ inProgress: true, error: null });
83      try {
84        let result = await Location.geocodeAsync(this.state.selectedExample);
85        this.setState({ result });
86      } catch (e) {
87        this.setState({ error: e.message });
88      } finally {
89        this.setState({ inProgress: false });
90      }
91    };
92
93    _maybeRenderResult = () => {
94      let { selectedExample } = this.state;
95      let text = typeof selectedExample === 'string'
96        ? selectedExample
97        : JSON.stringify(selectedExample);
98
99      if (this.state.inProgress) {
100       return <ActivityIndicator style={{ marginTop: 10 }} />;
101     } else if (this.state.result) {
102       return (
103         <Text style={styles.resultText}>
104           {text} resolves to {JSON.stringify(this.state.result)}
105         </Text>
106       );
107     } else if (this.state.error) {
108       return (
109         <Text style={styles.errorResultText}>
110           {text} cannot resolve: {JSON.stringify(this.state.error)}
111         </Text>
112       );
113     }
114   };
115
116   _renderExample = (example, i) => {
117     let { selectedExample } = this.state;
118     let isSelected = selectedExample === example;
119     let text = typeof example === 'string' ? example : JSON.stringify(example);
120
121     return (
122       <Touchable
123         key={i}
124         hitSlop={{ top: 10, bottom: 10, left: 20, right: 20 }}
125         onPress={() => this._selectExample(example)}>
126         <Text
127           style={[
128             styles.exampleText,
129             isSelected && styles.selectedExampleText,
130           ]}>
```

```
          {text}
        </Text>
      </Touchable>
    );
  };

  _selectExample = example => {
    if (this.state.inProgress) {
      return;
    }

    this.setState({ selectedExample: example, result: '', error: '' });
  };
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
  },
  separator: {
    height: 1,
    backgroundColor: '#eee',
    marginTop: 10,
    marginBottom: 5,
  },
  headerText: {
    fontSize: 18,
    fontWeight: '600',
    marginBottom: 5,
  },
  headerContainer: {
    borderBottomWidth: 1,
    borderBottomColor: '#eee',
    marginHorizontal: 20,
    marginBottom: 0,
    marginTop: 20,
  },
  exampleText: {
    fontSize: 15,
    color: '#ccc',
    marginVertical: 10,
  },
  examplesContainer: {
    paddingTop: 15,
    paddingBottom: 5,
    paddingHorizontal: 20,
  },
  selectedExampleText: {
    color: 'black',
  },
  resultText: {
    padding: 20,
  },
  actionContainer: {
    flexDirection: 'row',
    alignItems: 'center',
    justifyContent: 'center',
    marginVertical: 10,
  },
  errorResultText: {
    padding: 20,
    color: 'red',
  },
  button: {
    ...Platform.select({
```

```
196         android: {
197           marginBottom: 10,
198         },
199       }),
200     },
201   });
```

GeoCodeExpo.js hosted with ❤ by GitHub view raw

Conclusion

In this blog, we learned how to implement Geolocation React Native app. We also learnt how to Convert Geocode in Location address(Reverse Geocoding) and Location Address into Geocode(Geocoding) in a simple React Native app.

. . .

Next Steps

If you liked this blog, you will also find the following React Native blogs interesting and helpful. Feel free to ask any questions in the comment section

- Firebase—Integrate Firebase | Analytics | Push notifications | Firebase CRUD
- How To in React Native—Geolocation | Life cycle hooks | Image Picker | Redux implementation | Make API calls | Navigation | Translation | Barcode & QR code scan | Send & Read SMS | Google Vision | Pull to Refresh
- Payments—Apple Pay | Stripe payments
- Authentication—Google Login | Facebook login | Phone Auth | Twitter login
- Create Instagram / Whatsapp Story Feature in React Native
- React Native life cycle hooks | Implement Redux | Async actions with Redux
- Create Awesome Apps in React Native using Full App

If you need a base to start your next React Native app, you can make your next awesome app using React Native Full App

React Native Full App by Enappd

Xceed Imagination • 2 years ago

Hello Umang,

Nice article.

Can you please help us on how to get background location tracking working in React Native?

Thanks,

Reply • Share ›

Ionic 5 Full App - Angular Cordova Ionic 5 Full App - Angular Capacitor Ionic 5 Full App - React Capacitor Ionic 4 Taxi Booking Complete Platform Ionic 4 Grocery Shopping Complete Platform Ionic 4 Spotify Clone

Other Links

Privacy Policy Terms and Conditions Licensing Options