CitiusTech

CPD #1268

Introduction to React Native Framework

Description: This CPD helps in understanding fundamentals of React native, a framework for building native apps.

Authors: Vidhya Jain
Reviewed by: Harshad Sawant, Vinil Menon

Keywords: React Native framework, Mobile Native applications, iOS, Android

About the CPD

- The CPD introduces the concept of “React Native Framework” used for developing native applications
- Further, it explains its architecture and offerings
- Finally, it concludes with current challenges and thoughts on developing with React Native Framework

Agenda

- **Introduction**
- React Native's Runtime & Flux Architecture
- React Native Offerings
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- Challenges
- Application Areas of React Native
- References

What is React Native?

- “React native” is a Framework for building native apps for iOS and Android
- Developed by Facebook and Instagram, React Native is an open source library released at GitHub on Mar 26, 2015
- Facebook launched this library with the focus on “learn once write anywhere”
- React native is written in web languages like JavaScript(JS), JSX, a subset of CSS etc.
- React native was born to overcome performance issues which Facebook faced during development of “Facebook Ad Org” web application

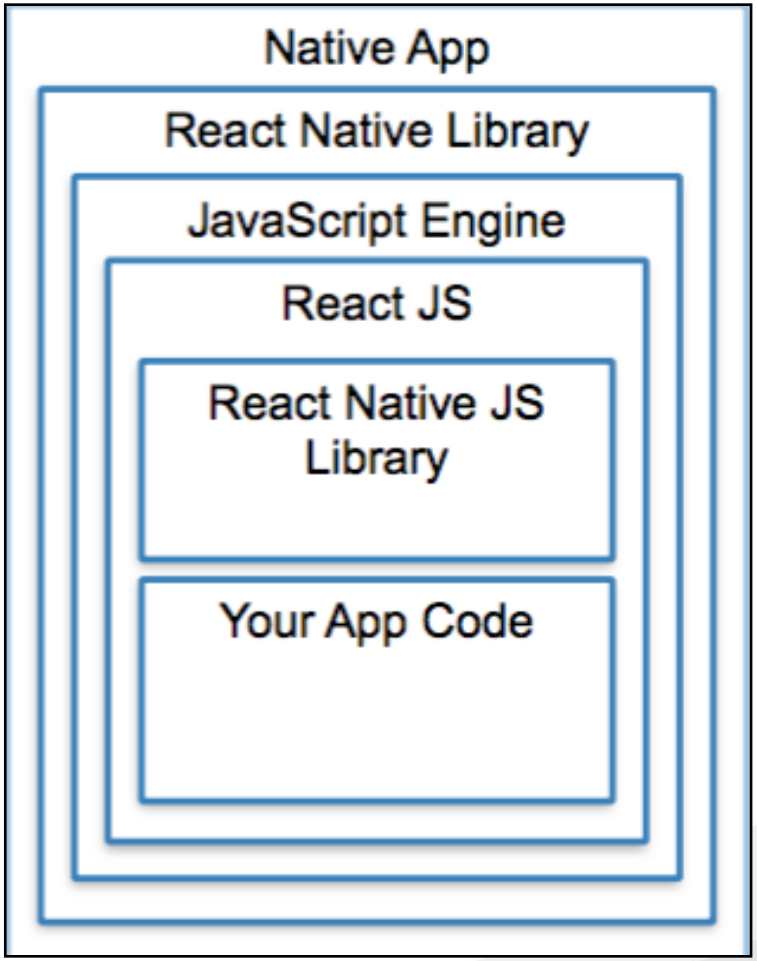


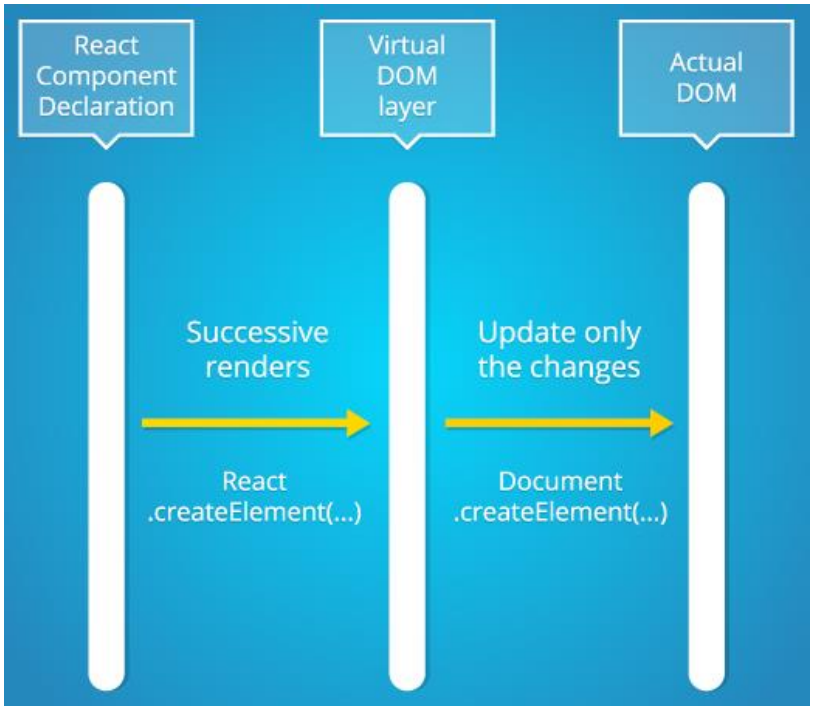
Image Source - http://moduscreate.com/

Why React Native?

- Facebook Ads Org is an application developed to Manage and create Facebook Ads on the go for small or medium business
- Frequent updates caused difficulties in managing high data and DOM structure. Small changes in the modal re-rendered complete view which made the application slow and affected user performance drastically
- Facebook launched “React Native” to solve this performance issues. It uses JS library named “React”, described as the fastest DOM rendering library. “React” is developed based on the concept of “Virtual DOM”

Concept of Virtual DOM

- Virtual DOM keeps track of data updates
- Virtual DOM plays an important role in
  - Comparing the new data with the old data versions
  - Calculate changes and updates actual DOM (UI), only with changes
  - Works asynchronously so main UI thread is not affected ,hence improving user experience and solving performance issues



Source - flipboard.com

Agenda

- Introduction
- **React Native's Runtime & Flux Architecture**
- React Native Offerings
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- Challenges
- Application Areas of React Native
- References

React Native's Runtime & Flux Architecture (1/4)

- Following architecture explains how React Native components are converted into native applications

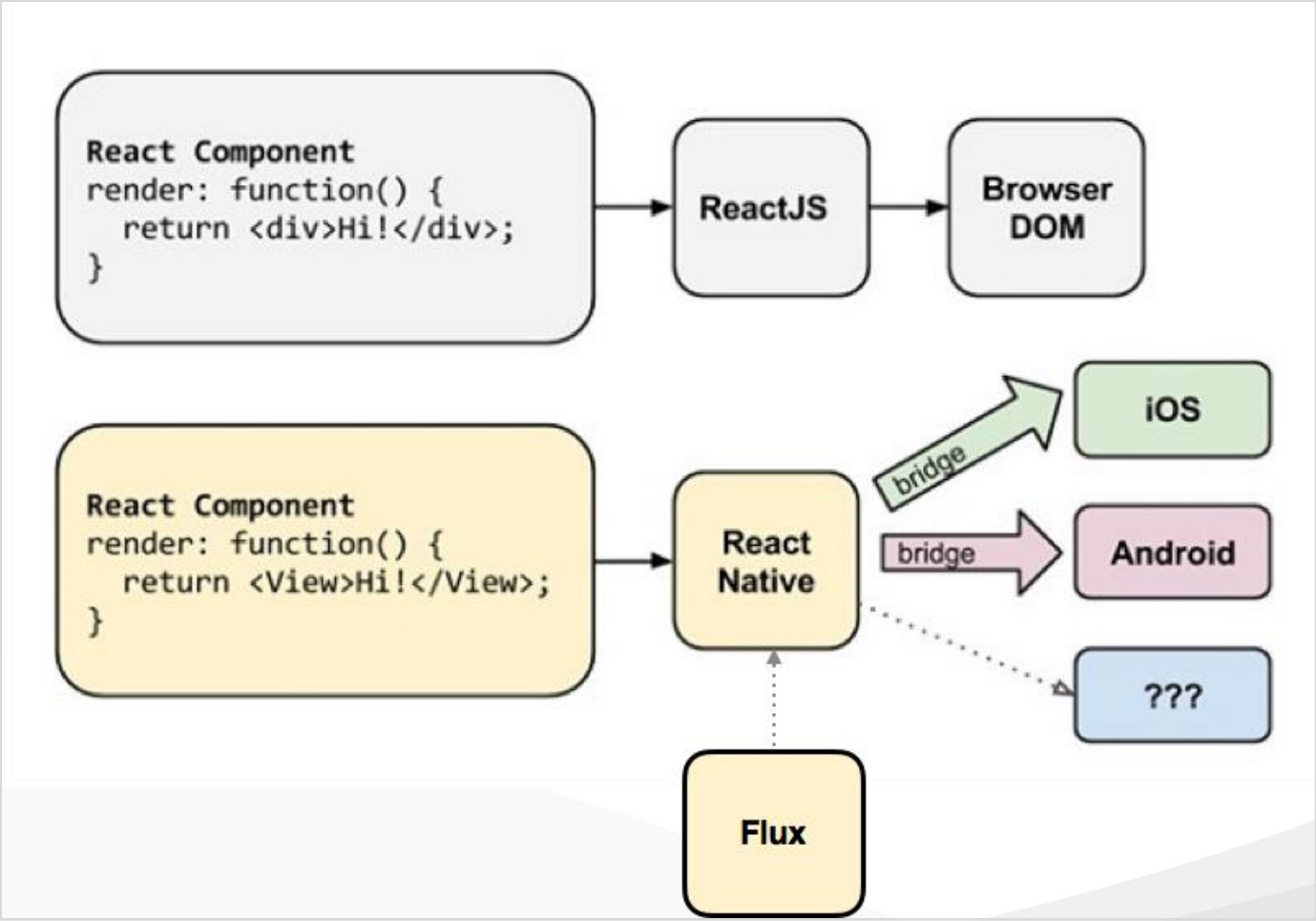


Image Source - https://www.infoq.com/articles/react-native-introduction

React Native’s Runtime & Flux Architecture (2/4)

- React Native uses custom asynchronous JS bridge running on a single thread, which asynchronously communicates with the native platform. This communication "converts" React Native components, to the native mobile OS components, for performing native operations
- JS bridge enables React Native to invoke the actual rendering APIs on iOS and Android
- Instead of compiling down to native code, React Native takes your application and runs it using the host platform's JavaScript engine, without blocking the main UI thread
- The JavaScript interpreter runs on a thread inside the native app on the device or emulator. However, during remote debugging, the JavaScript code can run in the developer's browser
- Frequent updates causes difficulties in managing high data using MVC pattern. MVC results in a multi-direction flow of data thus affecting user performance. Hence, React Native supports uni-directional architecture named “Flux”
- Flux avoids such weave of data flow with help of central dispatcher

React Native's Runtime & Flux Architecture (3/4)

- "Flux" - An architecture for creating one-way data layers in JavaScript applications
- React Native encourages one-way data flow using "Flux" architecture

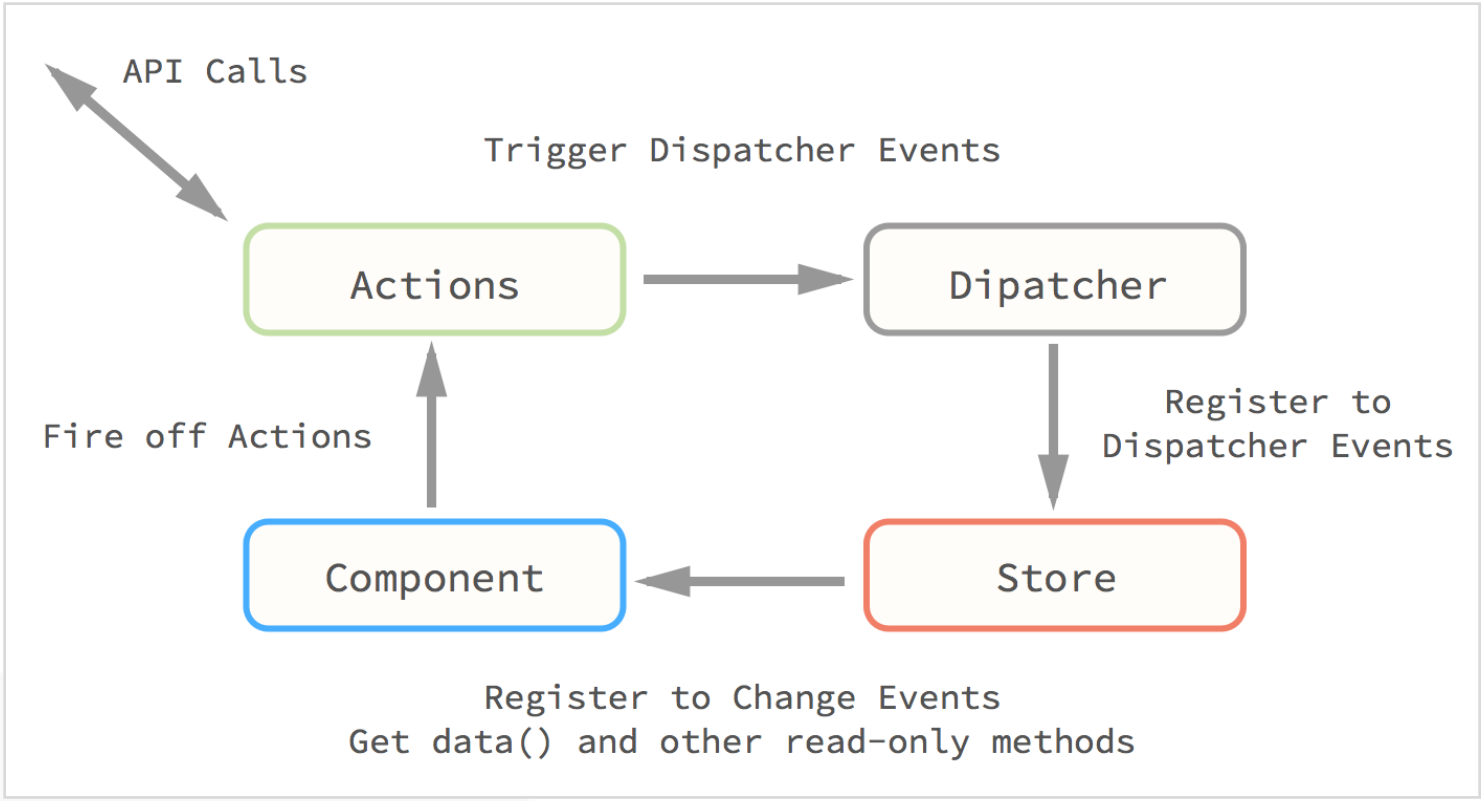


Image Source - http://engineering.kapost.com/wp-content/uploads/2015/05/figure4_3x13.png

React Native's Runtime & Flux Architecture (4/4)

- API Calls - "Fetch API" of React Native is used to perform network operations
- Actions - Helper methods which pass data to the Dispatcher
- Dispatcher - Dispatcher is a singleton object, operates as the central hub of data flow in an application. It is best used for performing asynchronous operations and acts as a controller for the application
- Store – Acts as a repository for the application
- Components - Used to define UI of the application. Components obtain the State as data from Stores and communicate them to the respective view in the hierarchy

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- **React Native Offerings**
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- Challenges
- Application Areas of React Native
- References

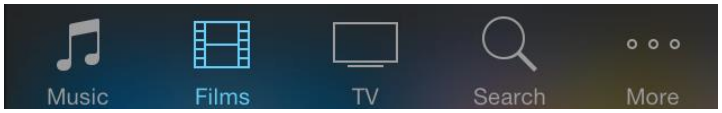
React Native Offerings - UI Components (1/3)

- Prominent UI components offered by React Native are as follows

UI Component	Description	Platform Supported
TouchableHighlight	Corresponds to Button control	iOS & Android
View	Container where components can be placed	iOS & Android
Navigator	Helps in transition of different scenes/views in the app.	iOS & Android
NavigatorIOS	Allows user to add back-swipe functionality	iOS
MapView	Display pins, user location on map view	iOS & Android
TabBarIOS	Bottom Bar with different Tabs/Buttons.	iOS
WebView	Open an Web Page or an HTML page in the application	iOS & Android
ToolbarAndroid	Bar that can display a logo, navigation icon, title & subtitle and a list of actions.	Android
TouchableOpacity	wrapper for making views respond properly to touches.	iOS & Android

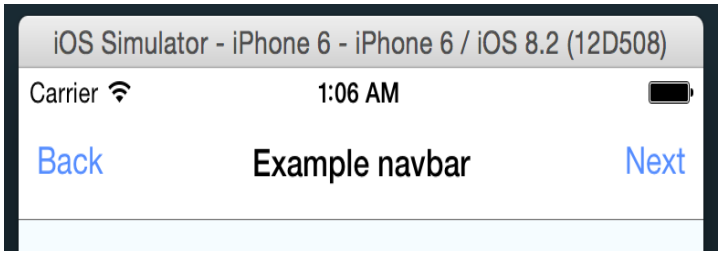
Complete list of offerings- https://facebook.github.io/react-native/docs/getting-started.html#content

React Native Offerings - UI Components (2/3)



TabBarIOS

TouchableHighlight

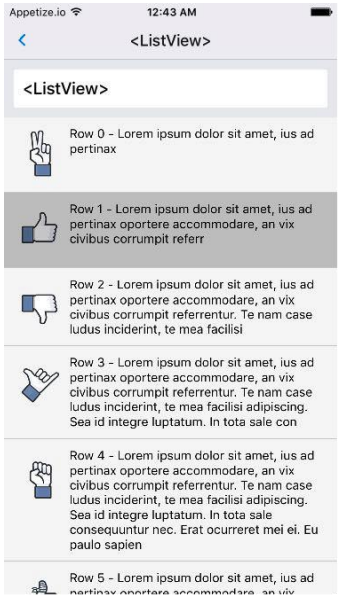


Navigator & NavigatorIOS

ToastAndroid

Source - https://facebook.github.io/react-native/docs/getting-started.html

React Native Offerings - UI Components (3/3)



ListView

PickerIOS

ProgressBarAndroid

Source - https://facebook.github.io/react-native/docs/getting-started.html

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- React Native Offerings
- **Programming Language of React Native**
- Comparison with Cross Platform dev. Frameworks
- Challenges
- Application Areas of React Native
- References

Programming Language of React Native

- React Native uses combination of web languages like Javascript, HTML and CSS (JSX)
- CSS is used for two main features of React Native - FlexBox and Styling
  - Flexbox makes it simple to build the most common UI layouts, such as stacked and nested boxes with margin and padding
- We still have to write separate code for Android and iOS
  - React Native provides ability to write view logic with a common strategy and common syntax across multiple platforms
  - However, iOS and Android will always have different features and hardware capabilities
  - React Native abstracts what is common between them giving access to platform-specific features in React-style JavaScript APIs

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- React Native Offerings
- Programming Language of React Native
- **Comparison with Cross Platform dev. Frameworks**
- Challenges
- Application Areas of React Native
- References

Comparison with Cross Platform Dev. Frameworks (1/2)

Parameters/Criteria	React Native	Xamarin	Ionic
Open standards-based development framework/Languages	HTML,CSS, Javascript, JSX	C# (Mono framework)	HTML, CSS and JS. (Angular might add learning curve and MVVM dependency)
Pricing and open source support	Open Source	Open Source	Open Source
Platform Support	iOS & Android	iOS, Android & Windows	iOS, Android & Windows
IDE Used	Xcode, Android Studio, Sublime Text	Xamarian Studio, visual Studio	Visual Studio Code, Atom, Sublime Text, WebStorm
Extensibility	Can refer existing native code library in React Native via JS bridge	Non - Interoperable. Cannot be referred in native & vice versa	Possible using Corodova capabilities. JS logic is embedded as Cordova component
Code Security	Not Possible. React Native's direct library to protect JS code not available.	Possible. Refer https://forums.xamarin.com/discussion/123/obfuscator-for-mono-for-android	Possible. Refer http://blog.ionic.io/minifying-your-source-code/

Comparison with Cross Platform Dev. Frameworks (2/2)

Parameters/Criteria	React Native	Xamarin	Ionic
Technology Supported by	Facebook & Instagram	Microsoft	Cordova
Documentation & Resources	Complete API guidelines available on website with examples, Github open source codes	Guides, Recipes, Samples, Forums, Videos available	Framework and Platform Docs, Forum, Blogs available
Product Maturity / Users	New Framework so less customers. Primarily Facebook, Instagram, Myntra	Over 15,000 companies rely on Xamarin like Honeywell,Kelloys etc	Highest number of customers.
Apps Showcase	https://facebook.github.io/react-native/showcase.html	https://blog.xamarin.com/introducing-the-xamarin-app-showcase/	http://showcase.ionicframework.com/

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- React Native Offerings
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- **Challenges**
- Application Areas of React Native
- References

Challenges

- React native was originally launched only for iOS platform and support for Android was added recently. Hence, certain modules/views/properties are not supported for Android
- Memory and performance profiling tools are not available
- React Native is an evolving framework and significant API changes are observed between releases
- Restricts execution of multiple instances of React Native Project at the same time
- Code is not entirely reusable for making iOS and Android Applications

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- React Native Offerings
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- Challenges
- **Application Areas of React Native**
- References

Application Areas of React Native

- React Native can be used for applications like:
  - High Data Driven Applications
  - React native can be used for applications with continues updates and changing data like news feeds apps, Ads. applications etc.
  - Chat applications like Facebook Messenger, Group applications like Facebook Groups which requires high performance can go for React Native
- Code cannot be obfuscated in react native
- React Native usage from Developer‘s point of View:
  - Ease of Development and Debugging using Google Chrome Tools
  - Instantly "refresh" your application to see your code changes like web browser hence saves lot of development time
  - Learn JS only once and can be used for all platforms
  - Existing Native APIs can be used via React Native bridge
  - React Native is an evolving framework and significant API changes are observed between releases

Pre-requisites for React Native

Prerequisites

- Knowledge of HTML, CSS, JS required.
- Mac OS X is required for developing iOS application (can be used for android as well)
- Developer Tools
  - Xcode for iOS
  - Android Studio for Android
  - Sublime text for JS development
  - Google Chrome Tools are used for debugging
  - Terminal/Command line for installing react native tools
  - Homebrew - To install NodeJS
  - Watchman - A tool by Facebook for watching changes in the filesystem
  - Flow - For static typechecking of your React Native code
- Emulators
  - Genymotion is used to run Android Apps.
  - iPhone simulators

Agenda

- Introduction
- React Native's Runtime & Flux Architecture
- React Native Offerings
- Programming Language of React Native
- Comparison with Cross Platform dev. Frameworks
- Challenges
- Application Areas of React Native
- **References**

References

- https://facebook.github.io/react-native/
- React Native Installation & Framework Documentation - http://facebook.github.io/react-native/docs/getting-started.html
- Videos - http://facebook.github.io/react/docs/videos.html
- Applications using React Native: https://facebook.github.io/react-native/showcase.html
- React Component Lifecycle - https://facebook.github.io/react/docs/component-specs.html
- Sample App - https://github.com/facebook/react-native
- Designing an App for Multiple Platforms using React native: http://makeitopen.com/tutorials/building-the-f8-app/design/
- Read more about Flexbox- https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- Check Styling in React Native - https://facebook.github.io/react-native/docs/style.html
- Integrating Data with React Native - http://makeitopen.com/tutorials/building-the-f8-app/data/
- Testing a React Native App -http://makeitopen.com/tutorials/building-the-f8-app/testing/
- List View example using react native - https://www.raywenderlich.com/126063/react-native-tutorial
- Basic tutorial for Android App - http://code.tutsplus.com/tutorials/creating-a-dictionary-app-using-react-native-for-android--cms-24969

THANK YOU