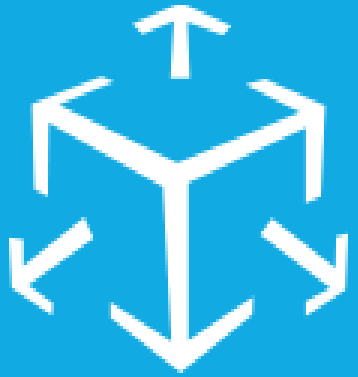


TOPS
Technologies

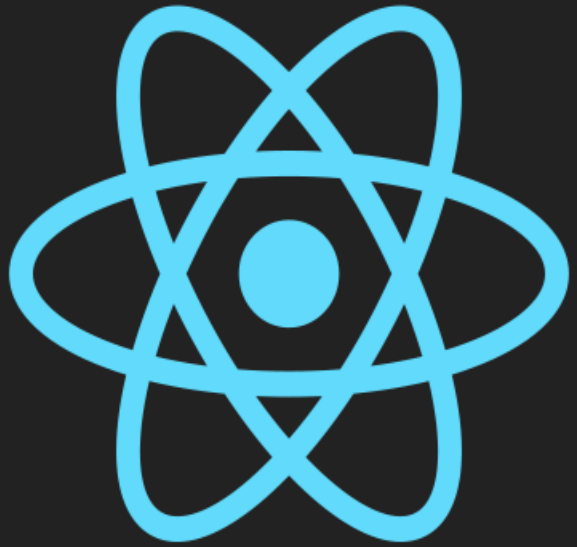
Introduction to student

- Career in Android
- Understanding Student Login of TOPS ERP
- Exam Process
- Working on Project and Assignment
- Using Lab
- Assign Project

TOPS
Technologies

React Native

TOPS Technologies



Module -1

TOPS Technologies

React Native Fundamentals

Introduction to React Native

- Based on Java script library React
- Build Cross platform application using Java script
- Java script framework for building IOS and android mobile applications
- Created by Facebook, first released iOS version on March 2015 and Android version on September 2015
- Writing the app by using JSX

React Native Features

Following are the features of React Native −

- React − This is a Framework for building web and mobile apps using JavaScript.
- Native − You can use native components controlled by JavaScript.
- Platforms − React Native supports IOS and Android platform.

React Native Advantages

Follow are the advantages of React Native −

- JavaScript − You can use the existing JavaScript knowledge to build native mobile apps.
- Code sharing − You can share most of your code on different platforms.
- Community − The community around React and React Native is large, and you will be able to find any answer you need.

React Native Limitation

Following are the limitations of React Native −

- Native Components − If you want to create native functionality which is not created yet, you will need to write some platform specific code.

Hybrid framework vs Native

Hybrid framework vs Native

Hybrid App	Native App
Developed using HTML, CSS, and Javascript	Developed in platform specific language, Objective-C or Swift for iOS, Java for Android, etc.
Write Once, Run Anywhere	Separate code for each platform
Medium performance comparable to Native apps	Fastest and most responsive experience to users
Save Time and Money	Higher investment of time, talent and resources
Faster development cycle	Higher costs and development time
Eg. Baskin Robbin, Sworkit, Untappd	Eg. PayPal, Gmail

Hybrid Framework vs Native

- Hybrid apps are built using web technologies like HTML, CSS and JavaScript whereas Native apps built with specific technology and language for specific platform like Java for Android, Swift for iOS.
- Hybrid app can be built for any platform from single code base.

Why Hybrids are better than natives?

- Unlike hybrid apps, native apps are built especially for the platform they're to be used on (iOS, Android etc).
- React Native allows a proportion of the code to be shared between platforms and empowers developers to create apps which feel less clunky and perform better than hybrid apps.

HTML 5

- HTML5 is the next major revision of the HTML standard superseding HTML 4.01, XHTML 1.0, and XHTML 1.1. HTML5 is a standard for structuring and presenting content on the World Wide Web.
- HTML5 is a cooperation between the World Wide Web Consortium (W3C) and the Web Hypertext Application Technology Working Group (WHATWG).
- The new standard incorporates features like video playback and drag-and-drop that have been previously dependent on third-party browser plugins such as Adobe Flash, Microsoft Silverlight, and Google Gears.

Example: HTML(5)

CSS3

- CSS is used to control the style of a web document in a simple and easy way.
- CSS is the acronym for "Cascading Style Sheet".

Example: CSS(SE)

Example: CSS3(SE)

Java Script

- JavaScript is a lightweight, interpreted programming language.
- It is designed for creating network-centric applications. It is complementary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML.
- It is open and cross-platform.

Difference between let and var and const

- var declarations are globally scoped or function scoped while let and const are block scoped.
- var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.
- While var and let can be declared without being initialized, const must be initialized during declaration

Example

JS Elements

- If - Example(Link)
- If-else - Example(Link)
- Nested If - Example(Link)
- Switch - Example(Link)
- For Loop - Example(Link)
- While - Example(Link)
- Do While - Example(Link)

JavaScript JSON

- JSON is a format for storing and transporting data. JSON is often used when data is sent from a server to a web page.

What is JSON?

- JSON stands for JavaScript Object Notation
- JSON is a lightweight data interchange format
- JSON is language independent *
- JSON is "self-describing" and easy to understand

ES6 JavaScript

New Features

- Let, var, const keyword
- Destructure
- Arrow Function
- High Order Function
- Class,
- Inheritance

Destructuring

- **Destructuring** is a feature introduced with ES6.
- A feature that enables you to break down a structure, like an array or object, and store its components in variables.

Example:(Link Array)

Example:(Link Object)

TOPS Technologies

Arrow Function

- An arrow function expression is a compact alternative to a traditional function expression, but is limited and can't be used in all situations.
- **Differences & Limitations:**
  - Does not have its own bindings to this or super, and should not be used as methods.
  - Does not have new.target keyword.
  - Not suitable for call, apply and bind methods, which generally rely on establishing a scope.
  - Can not be used as constructors.
  - Can not use yield, within its body.

Example:(Link)

TOPS Technologies

Higher Order Function

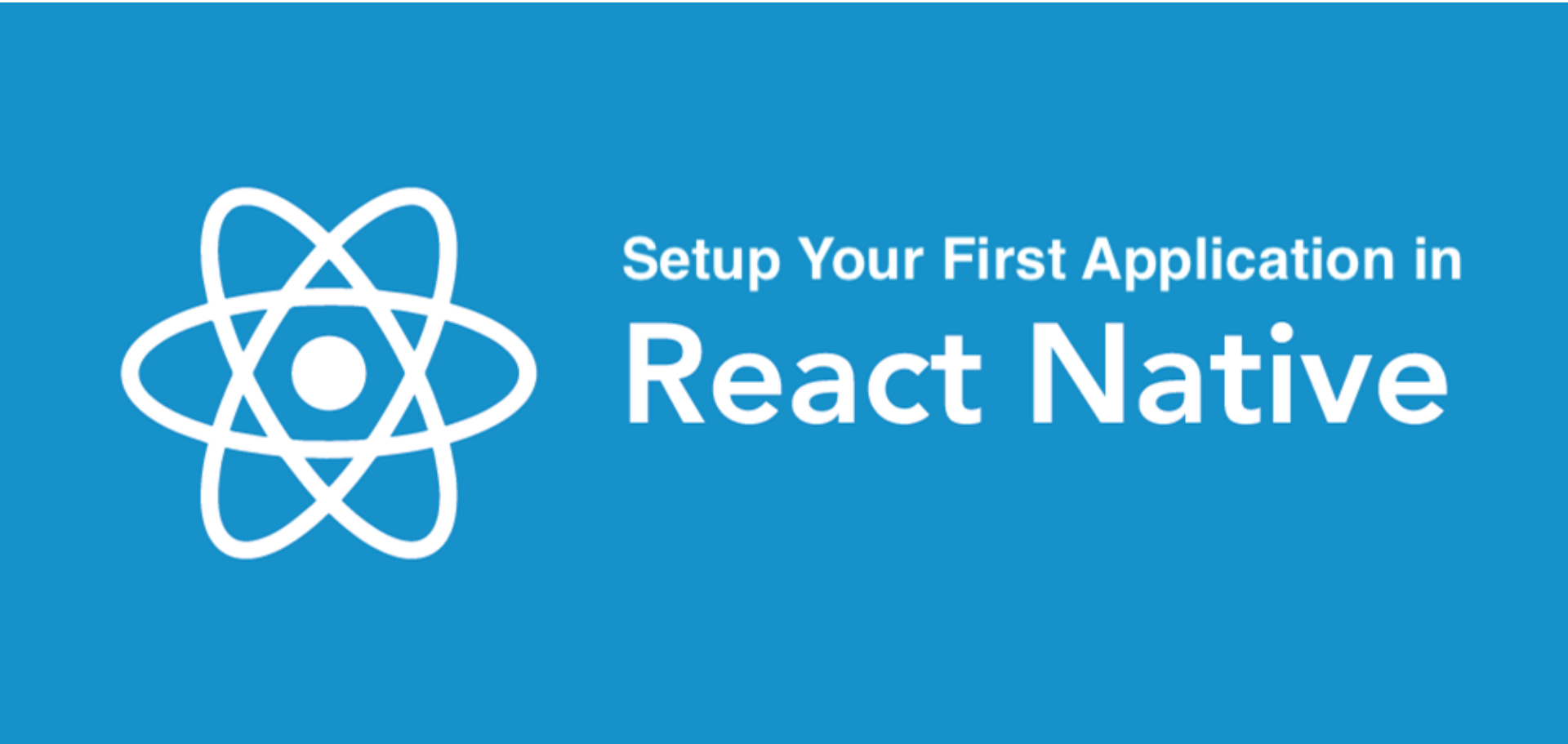
- Higher order functions can help you to step up your JavaScript game by making your code more declarative. That is, short, simple, and readable.

- A Higher Order Function is any function that returns a function when executed, takes a function as one or more of its arguments, or both.

1. Map - Example
2. Filter - Example
3. Reduce - Example
4. Combining All - Example

TOPS Technologies

Environment Setup



Tools Installation (Required)

There are a couple of things you need to install to set up the environment for React Native.

VS Code,

Node Js

TOPS Technologies

Environment Setup

- install create-react-native-app globally

```
npm install -g create-react-native-app
```

- Create Project

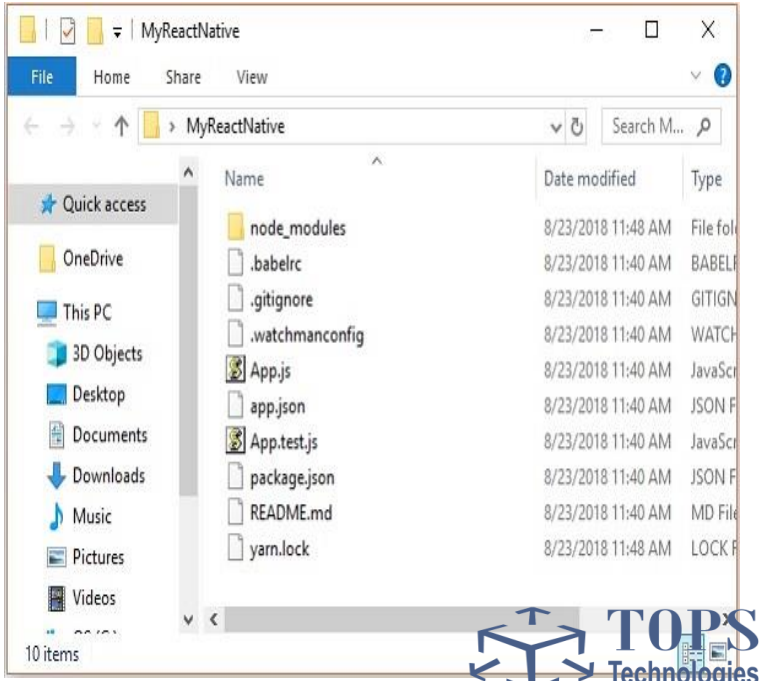
```
create-react-native-app MyReactNative
```

- Make sure Python Node Js and JDK 8 installed with your system
- Install React Native CLI

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

- Start react-native

```
npm start
```

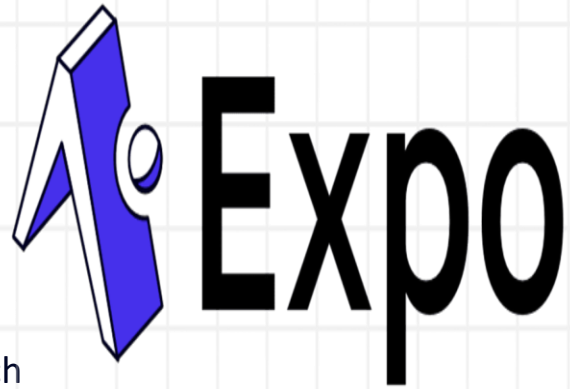


- Install Android Studio
- Configure AVD Manager
- Start Virtual Device then you can test your application with that device

- **Hot Reloading** : - modify App.js save the file it automatically updated in android device
- If not then select android virtual device then press ctrl + M then select enable hot reloading option

React Native CLI Structure

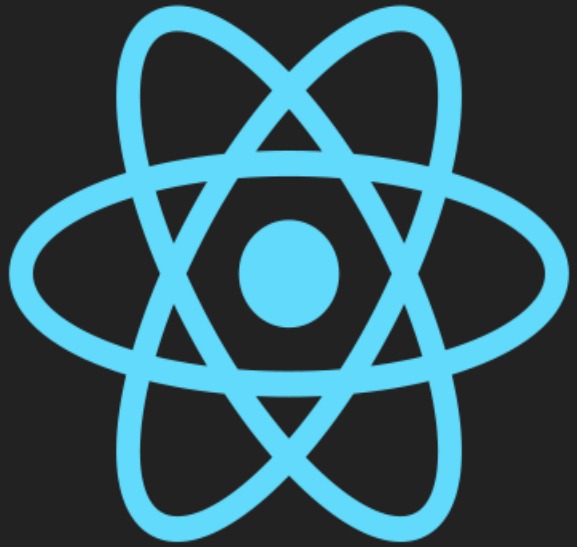
1. Step 1: Install Expo CLI or React Native CLI
2. Step 2: Install Xcode (for Mac OS)
3. Step 3: Install Android Studio
4. Step 4: Install IDE
5. Step 5: Create your first React Native app from scratch
6. Step 6: Set up React Native project on CI/CD



Enable

Example FirstApp

TOPS Technologies



Module - 2

Getting Started with React Basics

Getting started in React

- React is a JavaScript library for building user interfaces.

- React has been designed from the start for gradual adoption, and you can use as little or as much React as you need.

- Whether you want to get a taste of React, add some interactivity to a simple HTML page, or start a complex React-powered app.

What is JSX?

- JSX stands for JavaScript XML.
- JSX allows us to write HTML in React.
- JSX makes it easier to write and add HTML in React.
- JSX allows us to write HTML elements in JavaScript and place them in the DOM without any createElement() and/or appendChild() methods.
- JSX converts HTML tags into react elements.

Note : You are not required to use JSX, but JSX makes it easier to write React applications.

Why JSX?

- JSX makes it easier to write or add HTML in React.
- JSX can easily convert HTML tags to react elements.
- It is faster than regular JavaScript.
- As JSX is an expression, we can use it inside of if statements and for loops, assign it to variables, accept it as arguments, or return it from functions
- It is type-safe, and most of the errors can be found at compilation time.

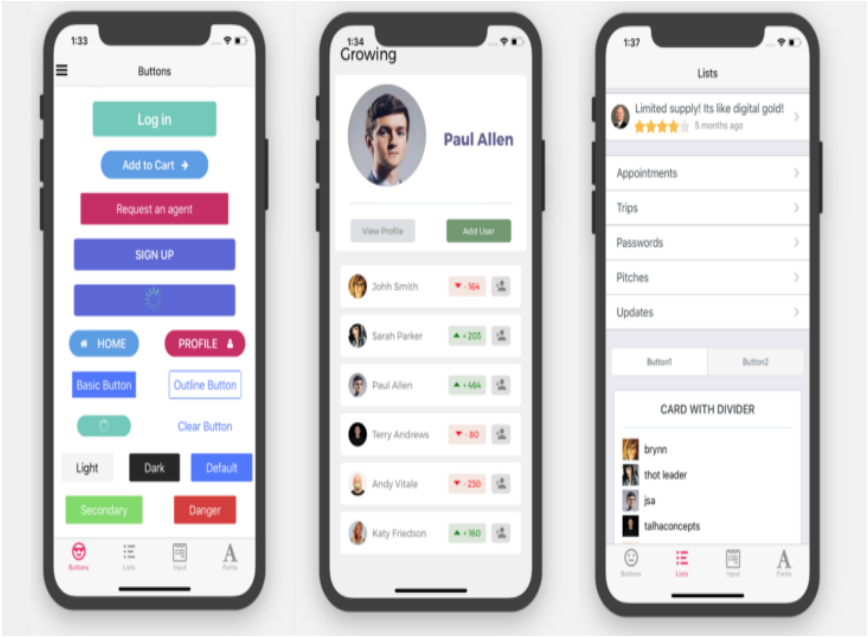
Const element = <h1>Hello World</h1>;

- It is called JSX, and it is a syntax extension to JavaScript. We recommend using it with React to describe what the UI should look like. JSX may remind you of a template language, but it comes with the full power of JavaScript.

JSX produces React “elements”

Example:(Link)

Components



TOPS Technologies

- Components are like functions that return HTML elements
- Components are independent and reusable bits of code
- They serve the same purpose as Javascript functions, but work in isolation and returns HTML via a render function
- Components come in two types, Class components and function components

Components Composition

- It's the ingredients and the arrangement of these ingredients to create something bigger out of it.
- It's the samples in a piece of music that make up a track.
- It's the fruits that are used for the perfect smoothie.
- It's the choreography of dancers in a musical.
- And it's the internals of a function in programming that need to be arranged in a way to get the desired output.

Function Components

- A Function component also returns HTML, and behaves pretty much the same way as a Class component, but Class components have some additions.

Why Function Component?

- No Class means no ‘this’
- Fewer lines
- Easier to read and understand

Syntax

```
const App = () =>{
    return (
        <Text>Hello World!</Text>
    )
}
export default App;
```

```
function App(){
    return (
        <Text>Hello World!</Text>
    )
}
export default App;
```

Example:(Link)

TOPS Technologies

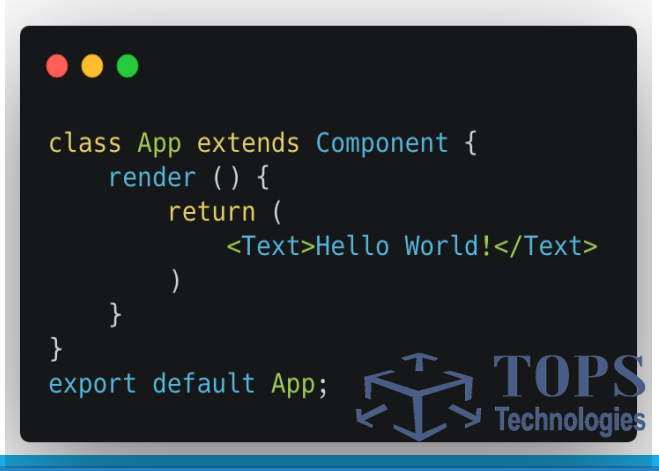
Class Components

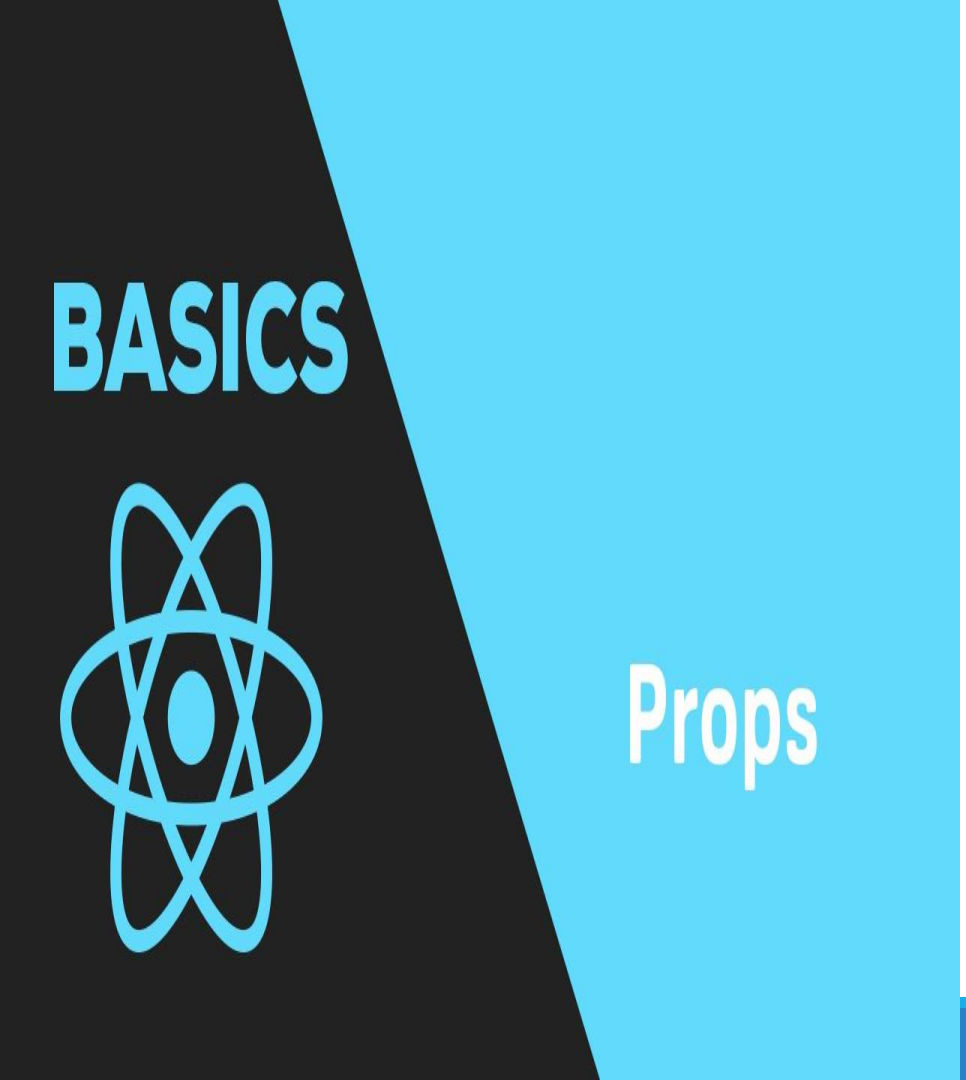
- The component's name must start with an uppercase letter.

- The component `extends React.Component` statement, this statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

- The component also requires a `render()` method, this method returns HTML.

Example:(Link)





Props

Props Basics

- Another way of handling component properties is by using props.
- Props are like function arguments, and you send them into the component as attributes.
- Props are arguments passed into React components.
- Props are passed to components via HTML attributes.
- React Props are like function arguments in JavaScript *and* attributes in HTML.

Example:(Pass Data using Props Link)

Props in the Constructor

- If your class component has a constructor function, the props should always be passed to the constructor and also to the React.Component via the `super()` method.

Example:(Link)

Note: React Props are read-only! You will get an error if you try to change their value.

Introducing States

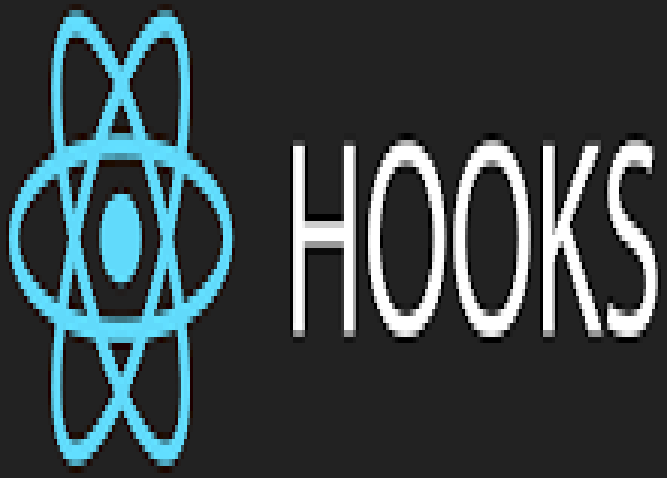
States introduction

Anything that changes over time is known as state.

1. React components has a built-in state object.
2. The state object is where you store property values that belongs to the component.
3. When the state object changes, the component re-renders.

Example:(Link Counter App) (Link state Object) (Link using setState) (Link previousState)

Note: Always use the setState() method to change the state object, it will ensure that the component knows its been updated and calls the render() method (and all the other lifecycle methods).



Introducing Hooks

Hooks introduction

- **React State Hook** that enables users to create state variables without classes.
- A Hook is a special function that lets you “hook into” React features. For example, useState is a Hook that lets you add React state to function components.
- If you write a function component and realize you need to add some state to it, previously you had to convert it to a class.

Types of Hooks:

1. **Basics**
2. **Additional**

TOPS Technologies

Types of Hooks

- Basics Hooks
  1. useState
  2. useEffect
  3. useContext
- Advanced Hooks
  1. useReducer
  2. useCallback
  3. useMemo
  4. useRef

Rules of Hooks

- Only Call Hooks at the Top Level
- Only Call Hooks from React Functions
- Don’t call Hooks from regular JavaScript functions. Instead, you can:
- Call Hooks from React function components.
- Call Hooks from custom Hooks.

Example(Link Custom Hooks)

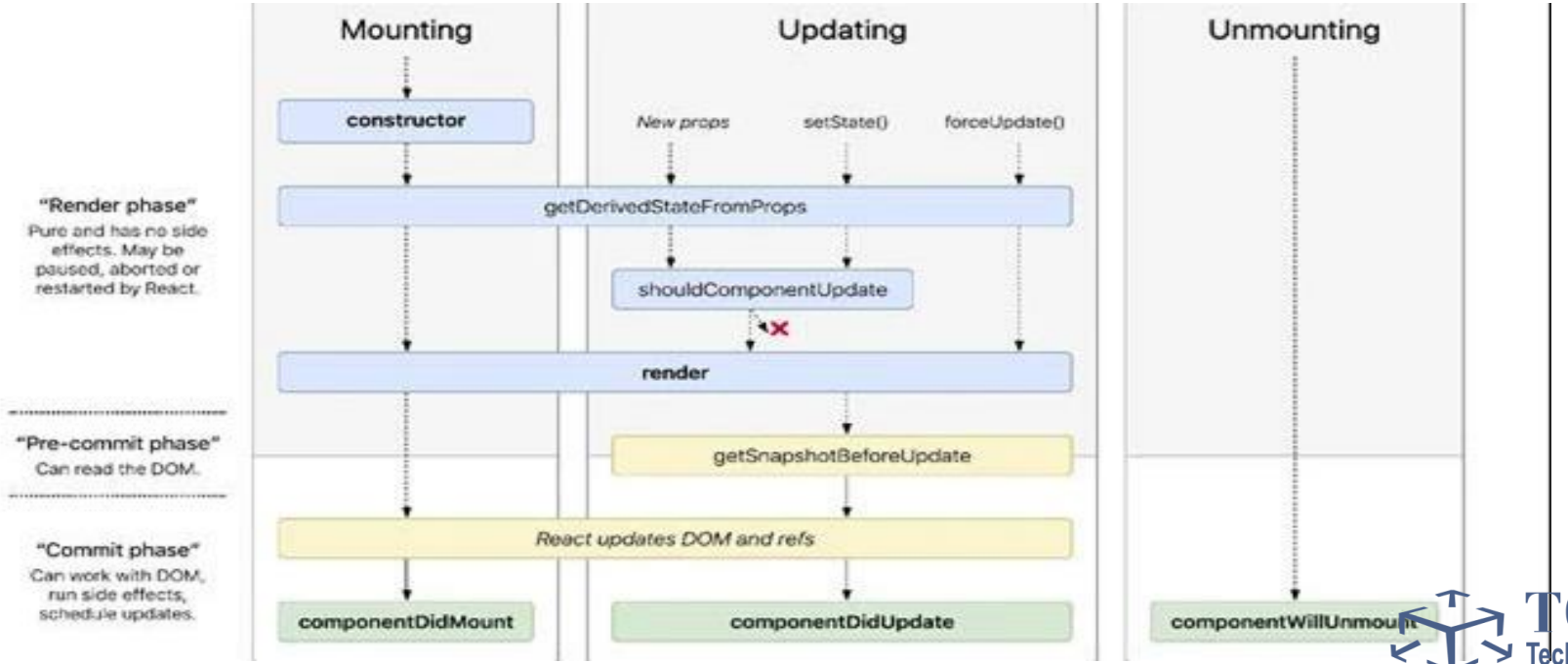
Life-Cycle of Component

Life Cycle of Component

A component's lifecycle can be divided into 4 parts:

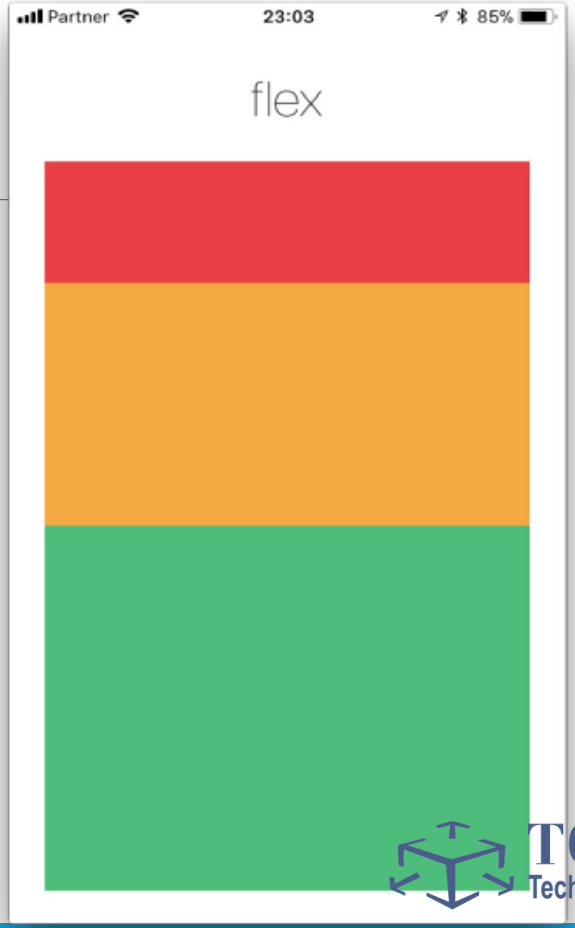
- **Mounting** — an instance of a component is being created and inserted into the DOM.
- **Updating** — when the React component is born in the browser and grows by receiving new updates.
- **Unmounting** — the component is not needed and gets unmounted.
- **Error handling** — called when there is an error during rendering, in a lifecycle method, or in the constructor of any child component.

Component Life Cycle

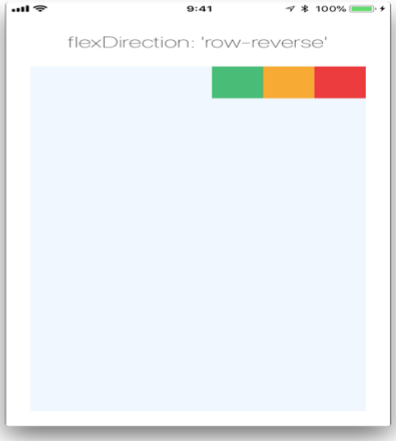


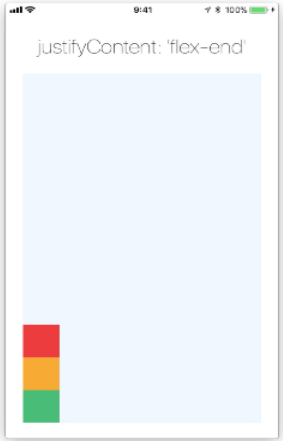
TOPS Technologies

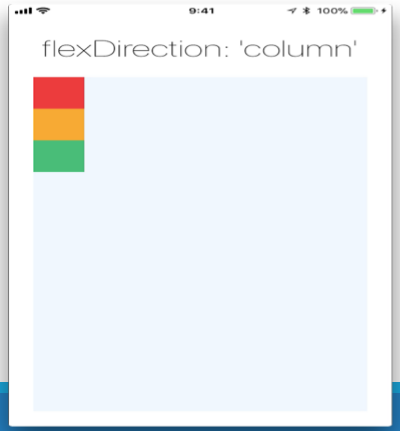
Layout with Flexbox

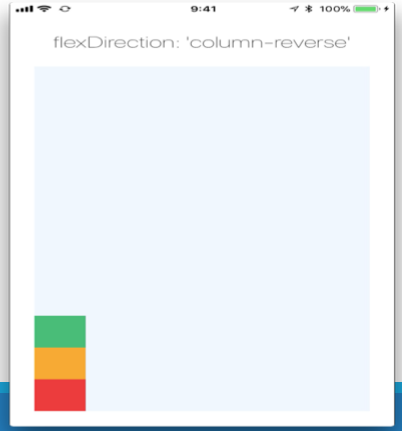


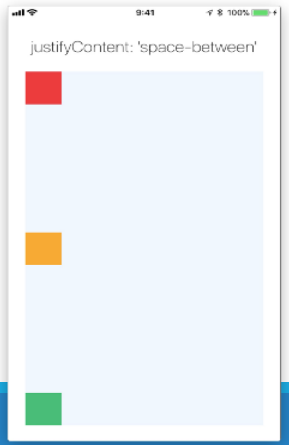
Layout with Flexbox

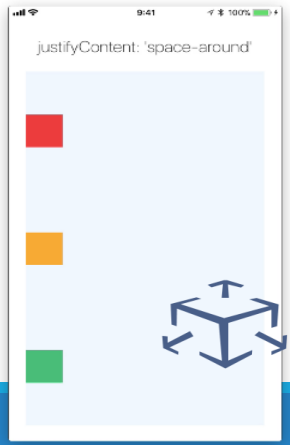






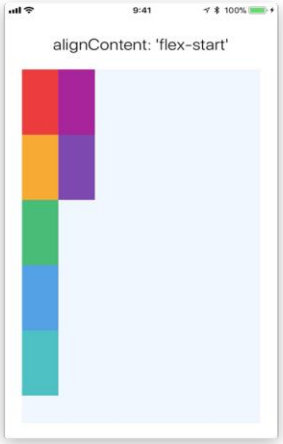


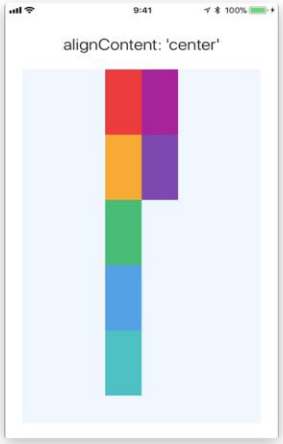


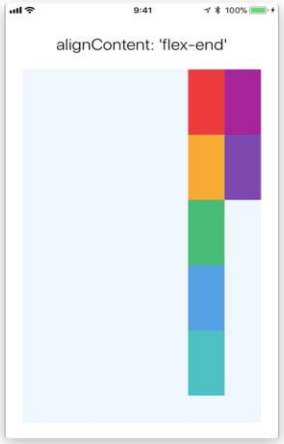


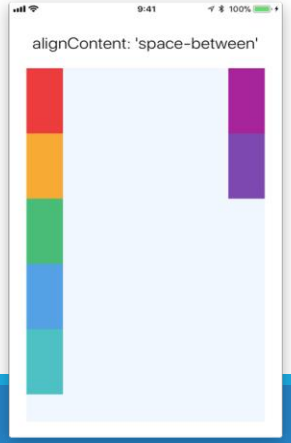
TOPS Technologies

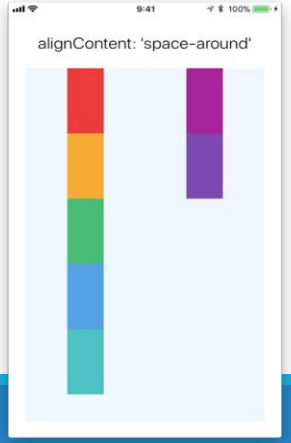
Layout with Flexbox

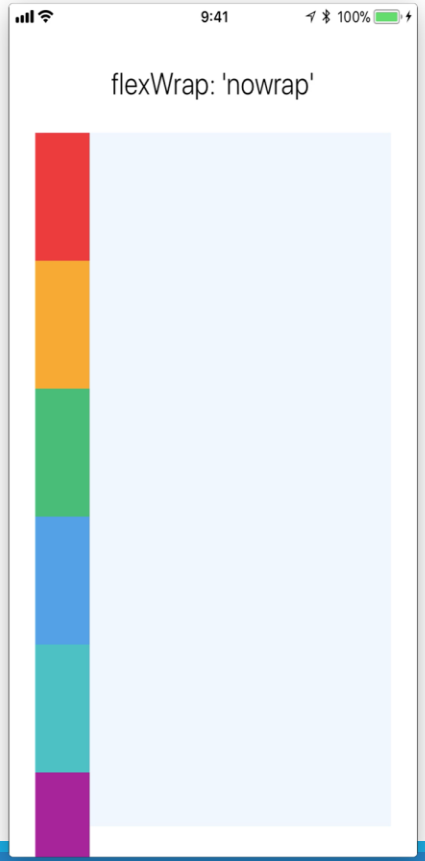


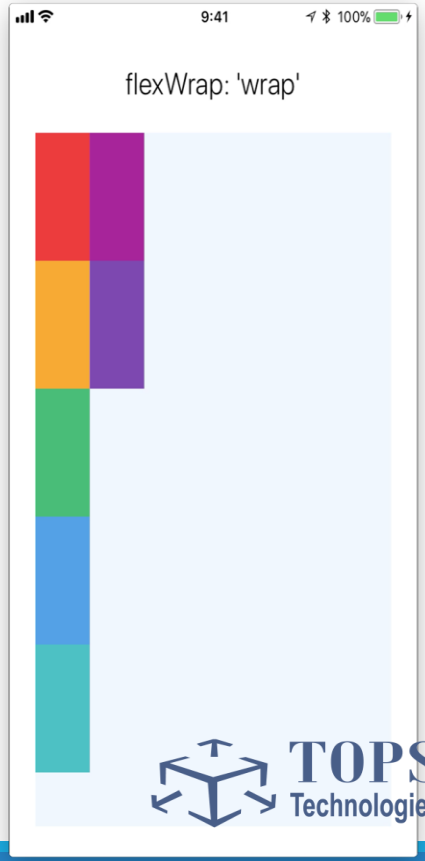






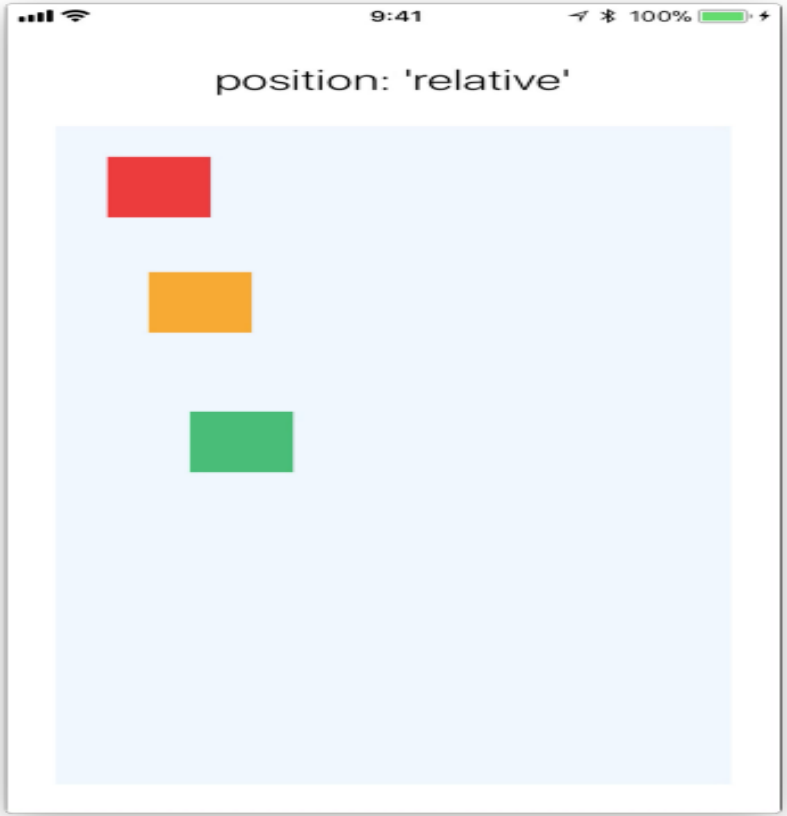


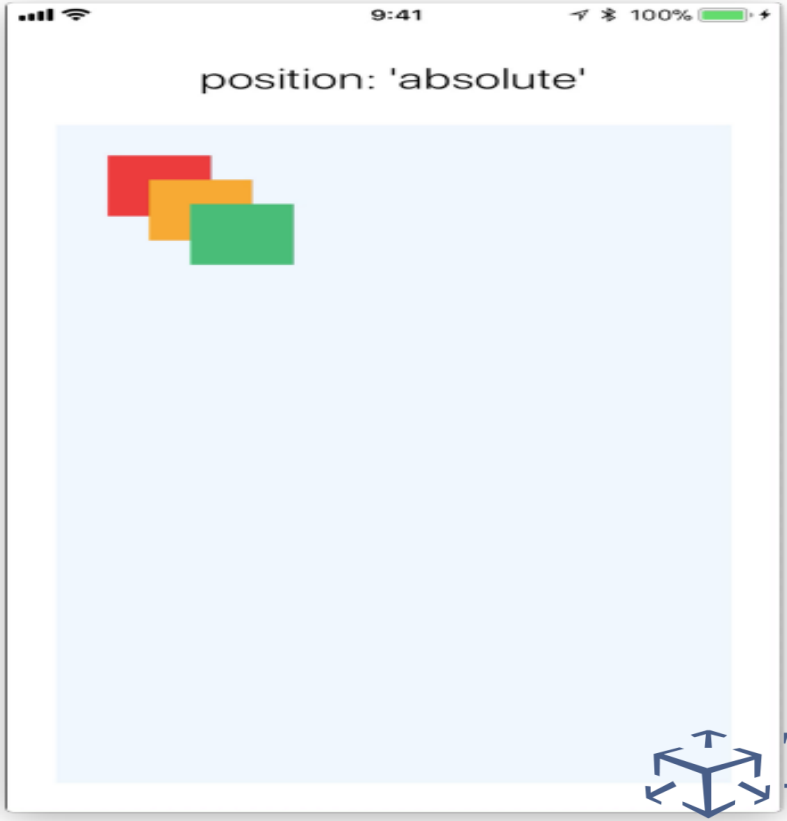




TOPS Technologies

Layout with Flexbox





Layout with Flexbox

Example(Link)

1. Flex
2. Flex Direction
3. Justify Content
4. Align Items
5. Align Self
6. Align Content
7. Flex Wrap
8. Absolute & Relative Layout

TOPS Technologies

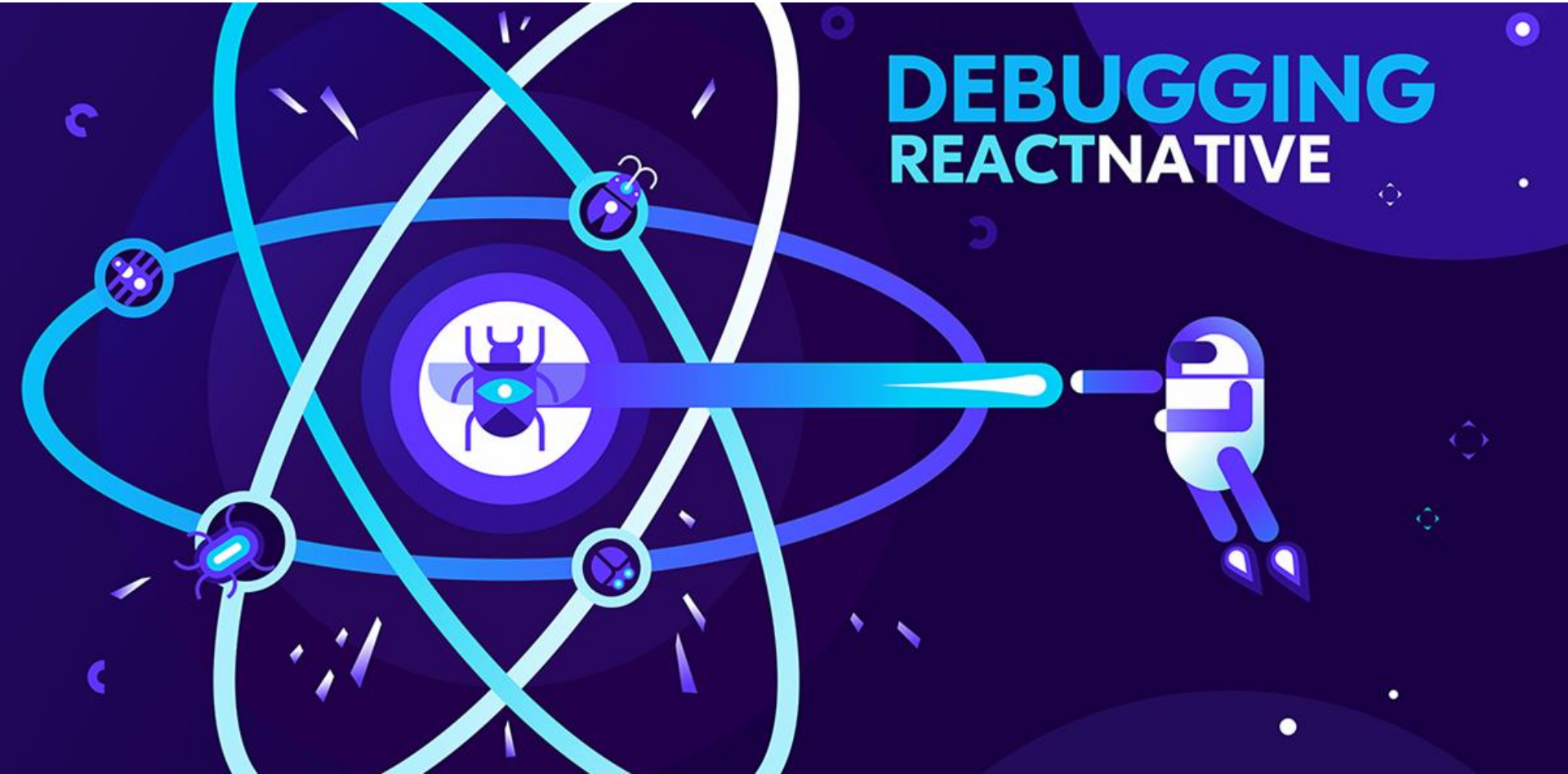
Styling the App

Styling the App

- React Native give us two powerful ways by default to style our application :
  - Style Props
  - Using Style Sheet

Example Using Style Prop Link
Example Using Style Sheet Link

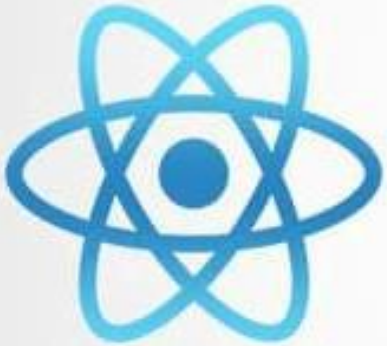
Inspecting and Debugging Style



Inspecting and Debugging Style

- Accessing the In-App Developer Menu
- Enabling Fast Refresh
- Enabling Keyboard Shortcut
- LogBox
- Consol Errors and Warning
- UnHandled Error
- Syntax Errors
- Chrome Developer Tool
- React Developer Tool
- Native Debugging

BUILT-IN COMPONENTS



REACT NATIVE
UI COMPONENTS

TOPS Technologies

Built-in Components

React Native provides a number of built-in Core Components ready for you to use in your app.

- Basic Components
- User Interface
- List Views
- iOS-specific
- Android-specific
- Others

Built-in Components

Most apps will end up using one of these basic components.

- **View** - The most fundamental component for building a UI
- **Text** - A React component for displaying text.
- **Image** - A React component for displaying different types of images
- **TextInput** - A foundational component for inputting text into the app via a keyboard
- **Scroll View** - Component that wraps platform ScrollView while providing integration with touch locking "responder" system.
- **Style Sheet** - A Style Sheet is an abstraction similar to CSS StyleSheets

1. View
    a. Function Component Example
    b. Class Component Example
2. Text
    a. Function Component Example
    b. Class Component Example
    c. Nested Text Example
3. Image
    a. Function Component Example
    b. Class Component Example
    c. Style Image Example 1
    d. Style Image Example 2

Built-in Components

1. TextInput
    a. Example
    b. Example2(Multiple Line)
2. ScrollView
    a. Example
3. List Views
    a. Flat List
        i. Example1
        ii. Example2(Selectable)
    b. Section List
        i. Using Function Component
        ii. Using Class Component

Built-in Components

1. IOS-Specific
   a. ActionSheetIOS
      i. Example
   b. SafeAreaView
      i. Example
2. Android-Specific
   a. Back Handler
      i. Using Function Component
      ii. Using Class Component
   b. DrawerLayout
      i. Example
   c. Permissions
      i. Using Function Component
      ii. Using Class Component
   d. Toast
      i. ToastAndroid Example
      ii. ToastAndroid Example(Advance)

TOPS Technologies

Built-in Components

1. Others
    a. ActivityIndicator
        i. Using Function Component
        ii. Using Class Component
    b. Alert
        i. Function Component
        ii. Using ClassComponent
    c. Animated
        i. Using Function Component
        ii. Using Class Component
    d. Dimensions
        i. Using Function Component
        ii. Using Class Component
    e. Keyboard Avoiding View
        i. Example

Built-in Components

1. Others
    a. Linking
        i. OpenLinks and DeepLinks Example
        ii. OpenCustomSettings Example
        iii. GetDeepLink Example
        iv. SendIntents Example
    b. Modal
        i. Using FunctionComponent
        ii. Using ClassComponent
    c. Pixel Ratio
        i. Example
    d. Refresh Control
        i. Example
    e. Status Bar
        i. Example

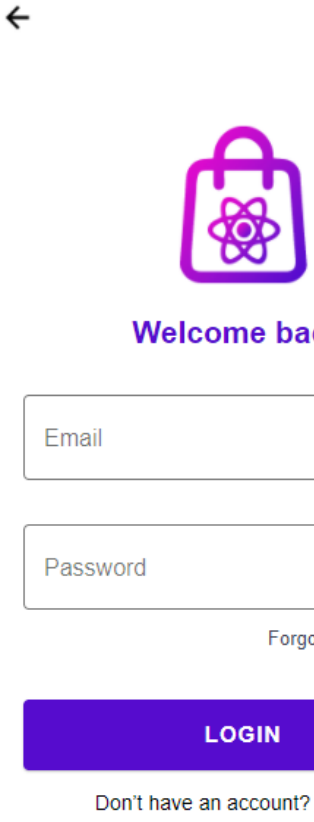
TOPS Technologies

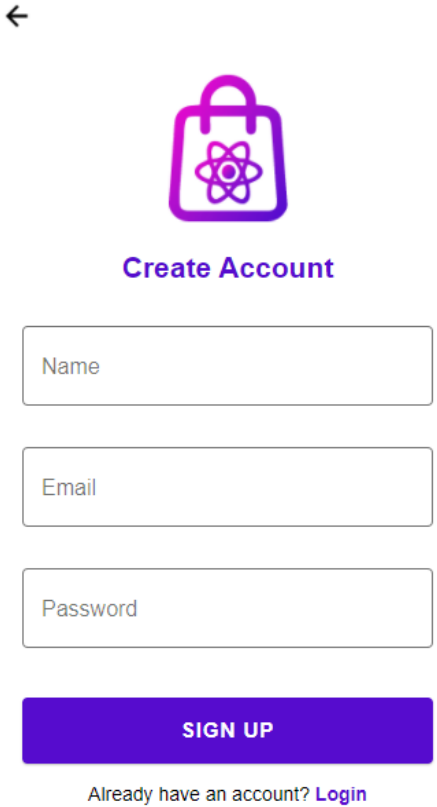
Built-in Components

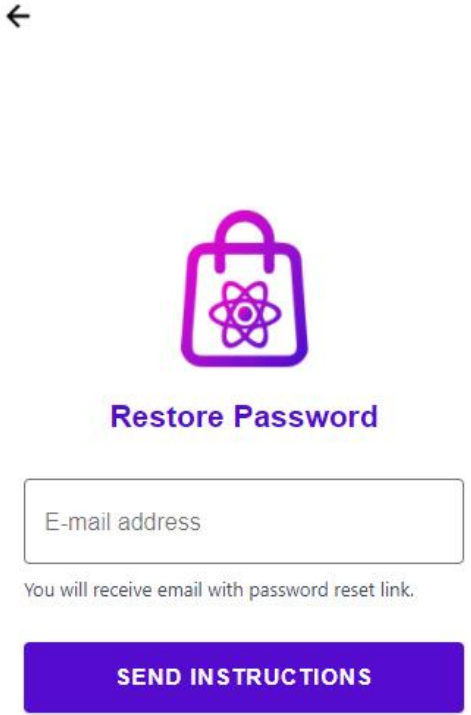
Touchables - Example

a. TouchableHighlight
b. TouchableNativeFeedback
c. TouchableOpacity
d. TouchableWithoutFeedback
e. Touchable with Long Press

Create Login Registration - Example







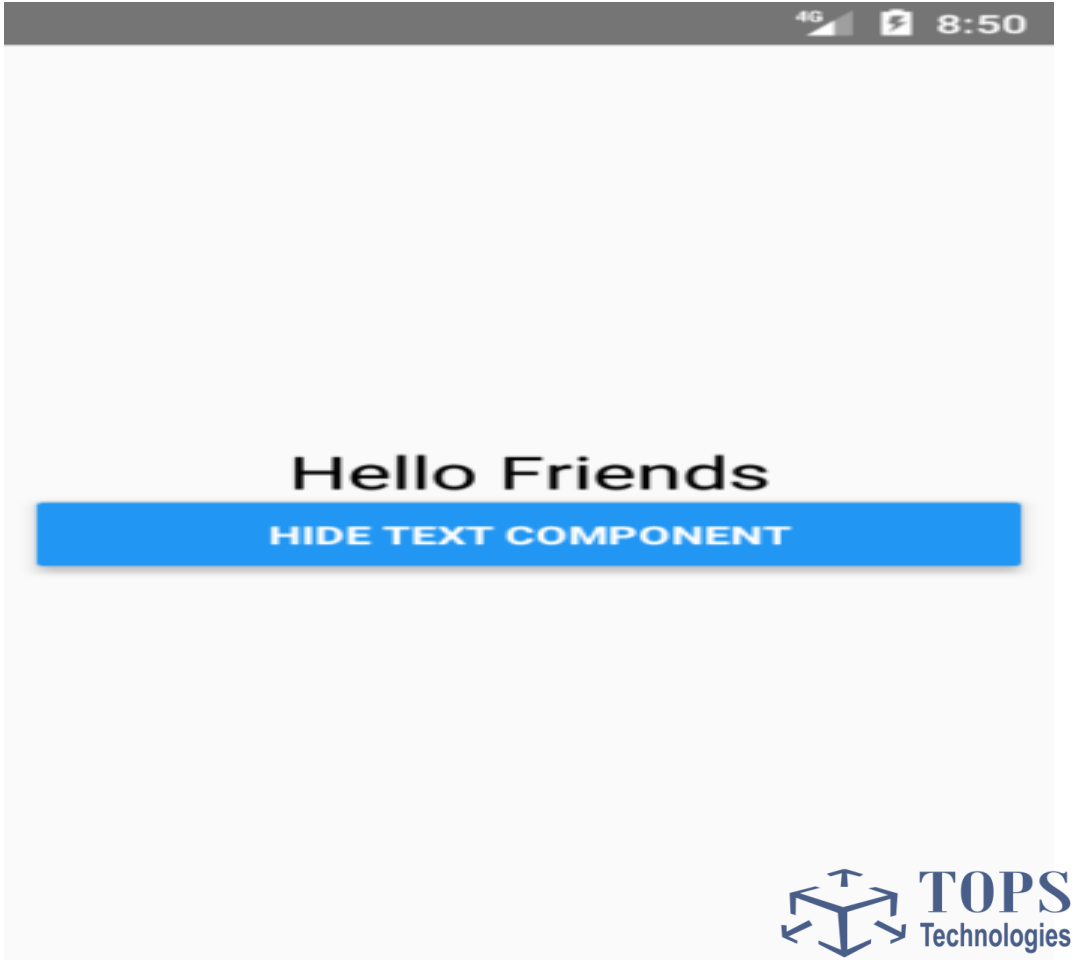
TOPS Technologies

Using State

Set background
Color Dynamically

Using State

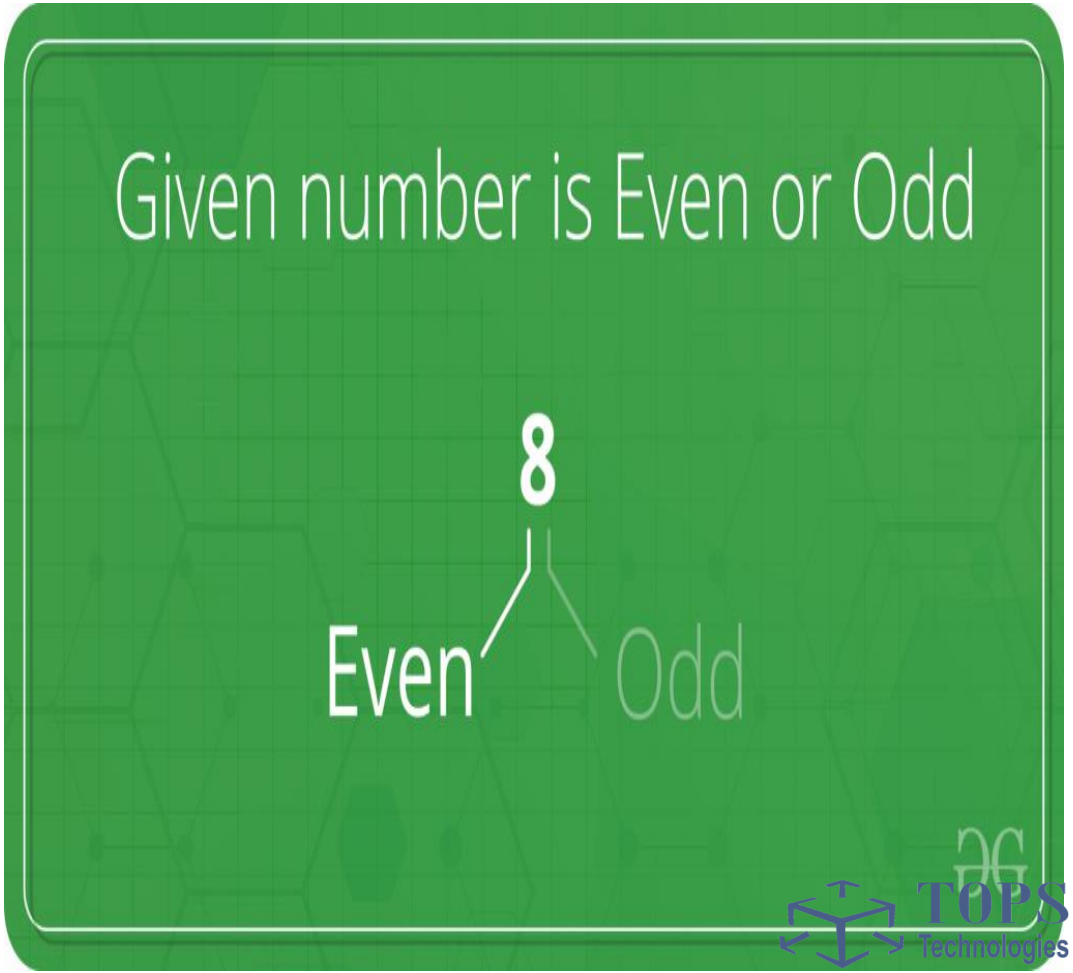
Hide - Unhide view on Button click event



TOPS Technologies

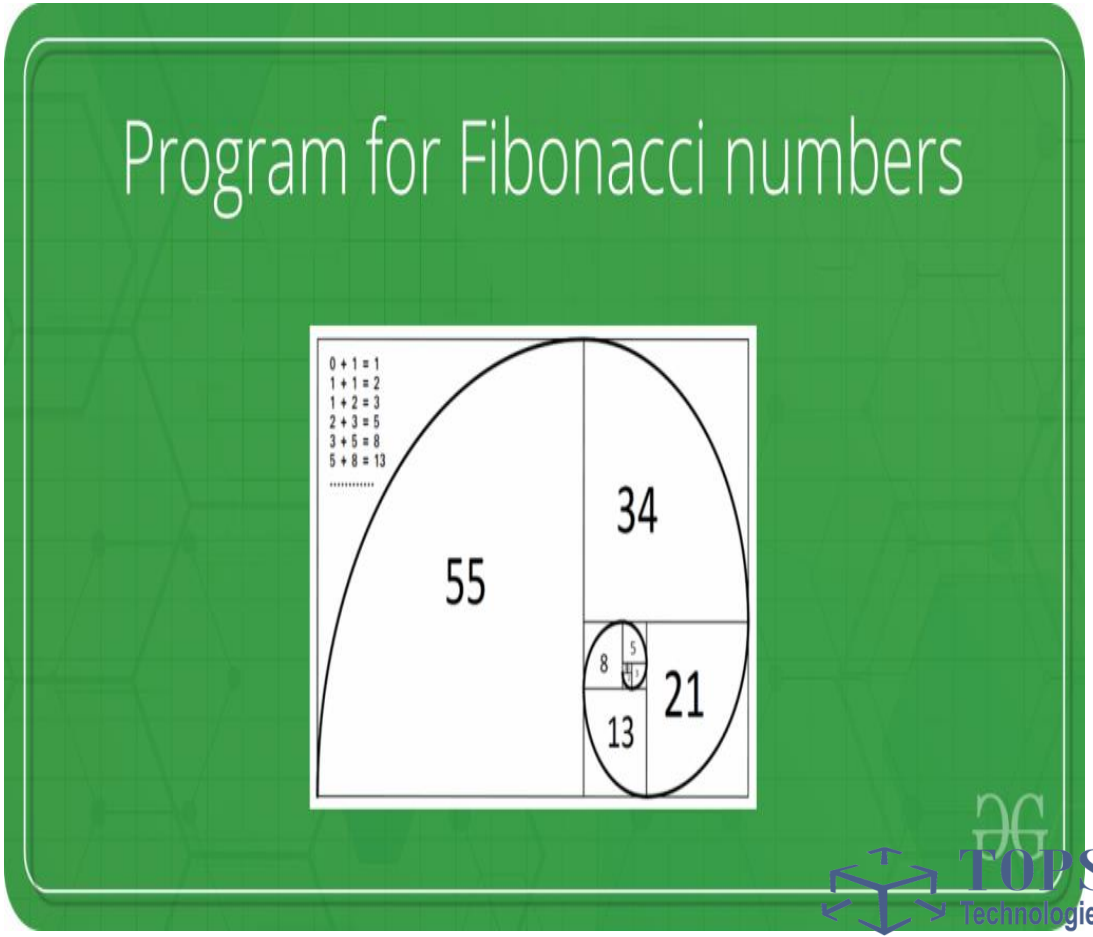
Using State

Find Even and Odd Number given by User



Using State

Find
Fibonacci Series of
given Number



Props

React Native Props

Props

- Communication from parent to child components.
- It can be done by using attributes values, and/or callback functions.
- Callback functions allows the child components to communicate back to the parents.

```
changeIndex(stockName, stockCode){
  API(stockCode).then((data)=>{
    this.setState({...data, stockName, stockCode});
  });
}
```

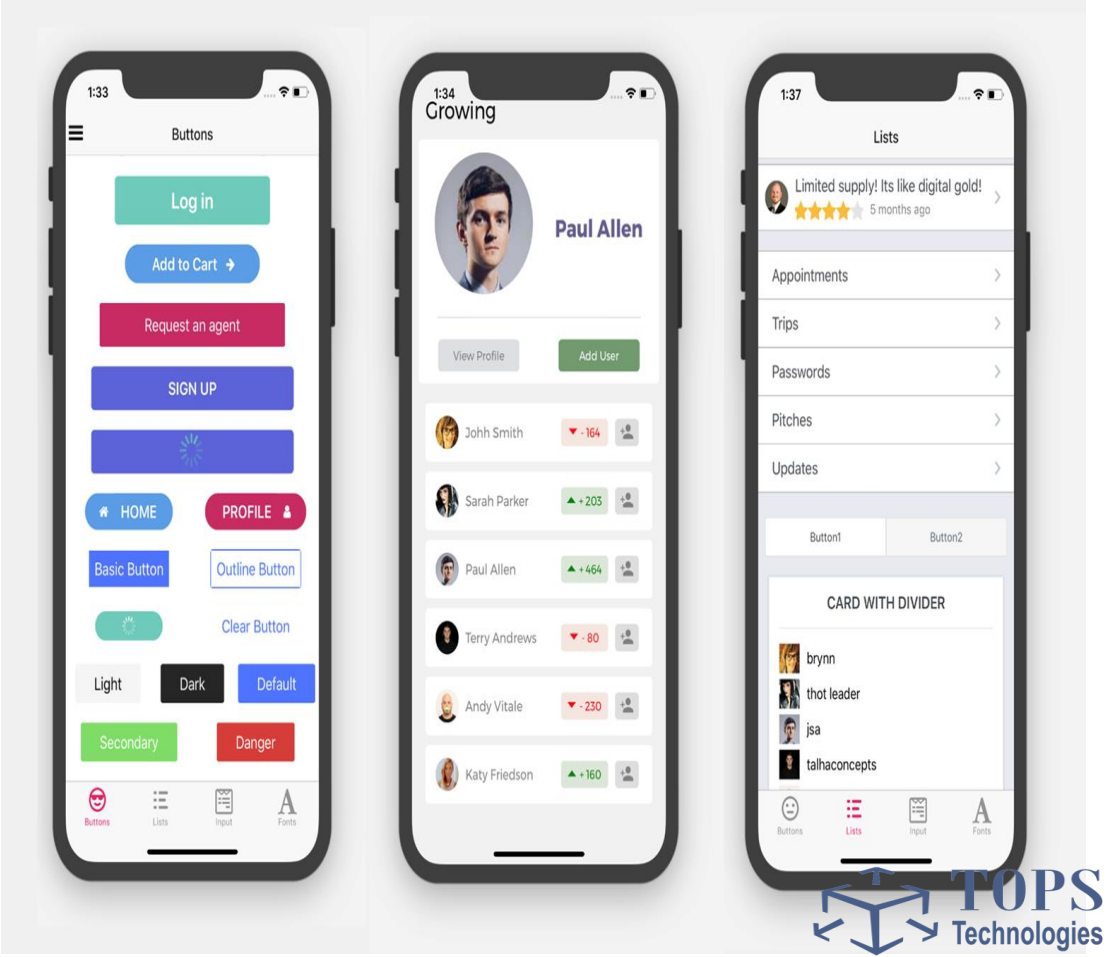
A Callback Function

```
<StockButton name="SET" code="INDEXBKK:SET" onPress={this.changeIndex}/>
```

Attribute Value Attribute Value Callback Function

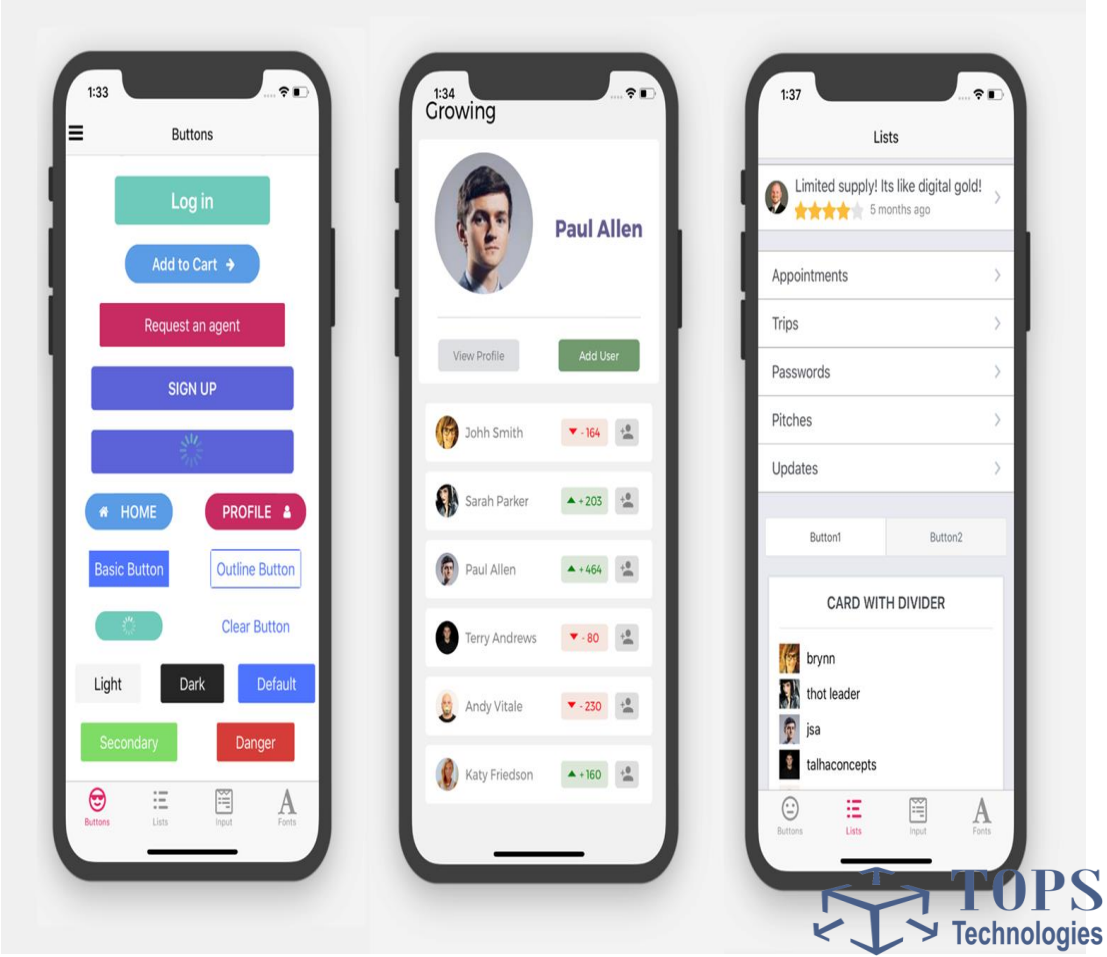
Props

- Text Props
  a. Practice
- TextInput Props
  a. Practice
- Button Props
  a. Practice
- Image Props
  a. Practice
- Switch
  a. Practice



Props

- Modal
  a. Practice
- Picker
  a. Practice
- Slider
  a. Practice

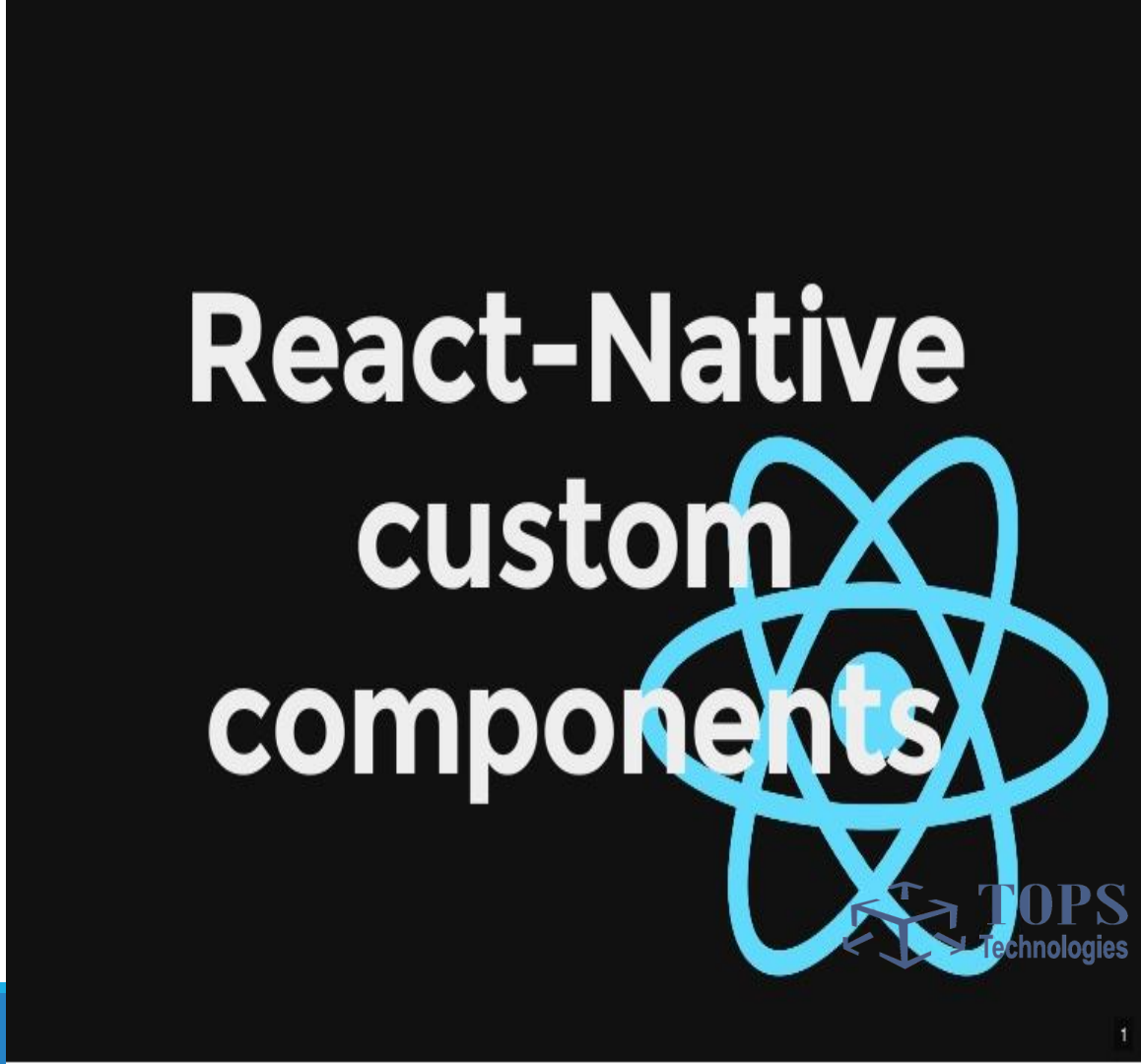


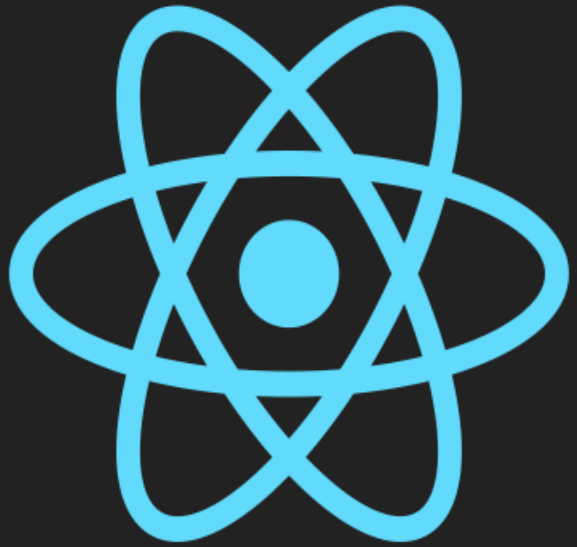
WebView - Example(Link)

Create Custom Components

Example

1. Custom TextInput
2. Custom TextView
3. Custom SnackBar
4. Custom CheckBox





Module - 3

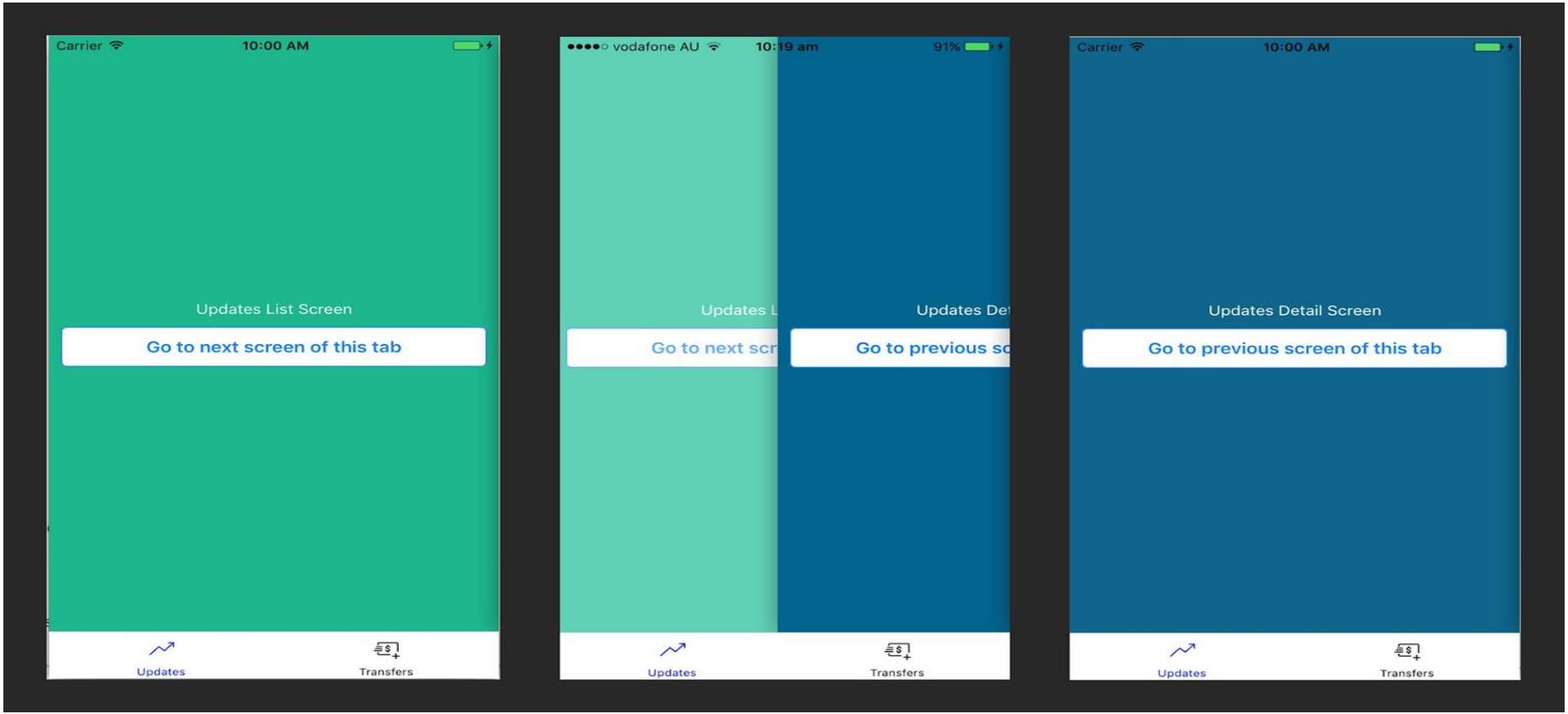
React Native Navigation

Navigating Between Screens

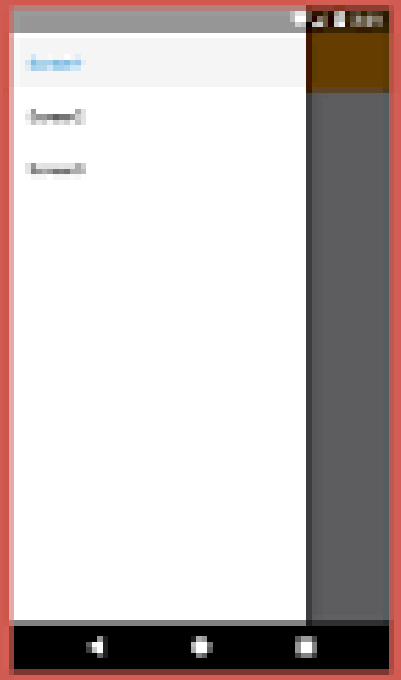
- Mobile apps are rarely made up of a single screen.
- Managing the presentation of, and transition between, multiple screens is typically handled by what is known as a navigator.
- If you are getting started with navigation, you will probably want to use React Navigation.
- React Navigation provides a straightforward navigation solution, with the ability to present common stack navigation and tabbed navigation patterns on both Android and iOS.
- If you'd like to achieve a native look and feel on both Android and iOS, or you're integrating React Native into an app that already manages navigation natively, the following library provides native navigation on both platforms: **react-native-navigation.**

Example Navigation Scene

Stack Navigator Example



Drawer Navigator Example



React Native
Navigation Drawer

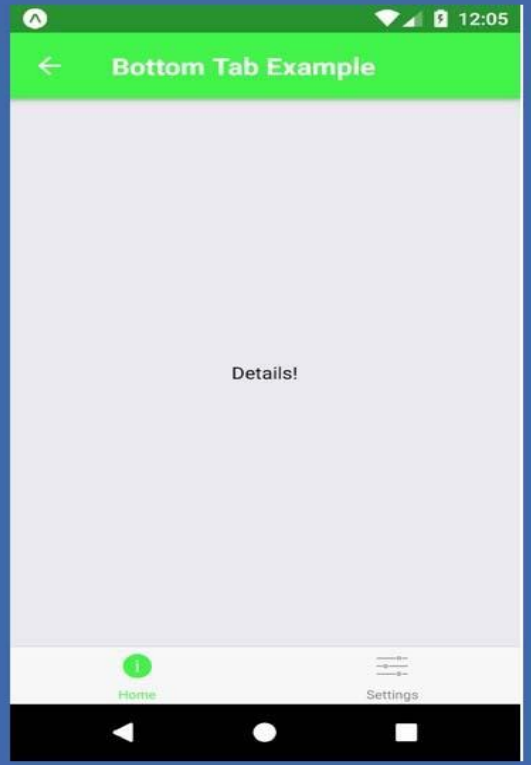
TOPS Technologies

Top Tab Navigator Example

React Native
Tab

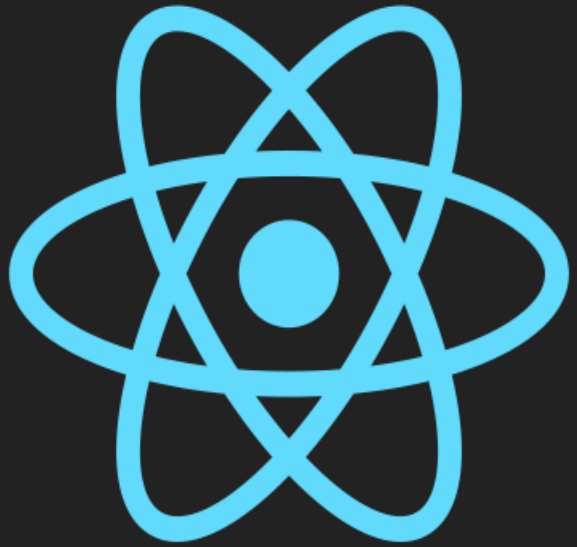
TOPS Technologies

Bottom Tab Navigator Example



React Native
Bottom Navigation

Splash Screen Example



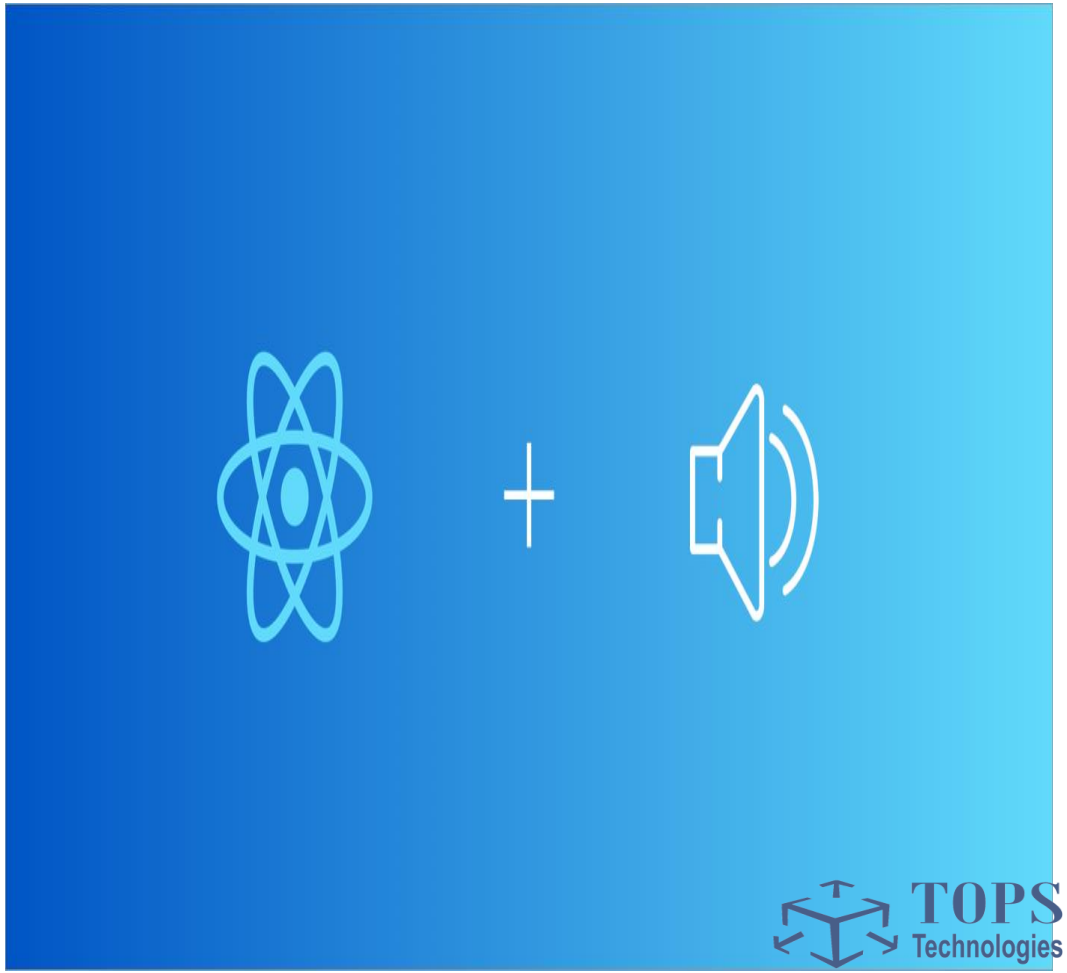
Module - 4

React Native OS

Working with Music and Video

1. Music

1. Video



Text to Speech and Speech to Text

Example

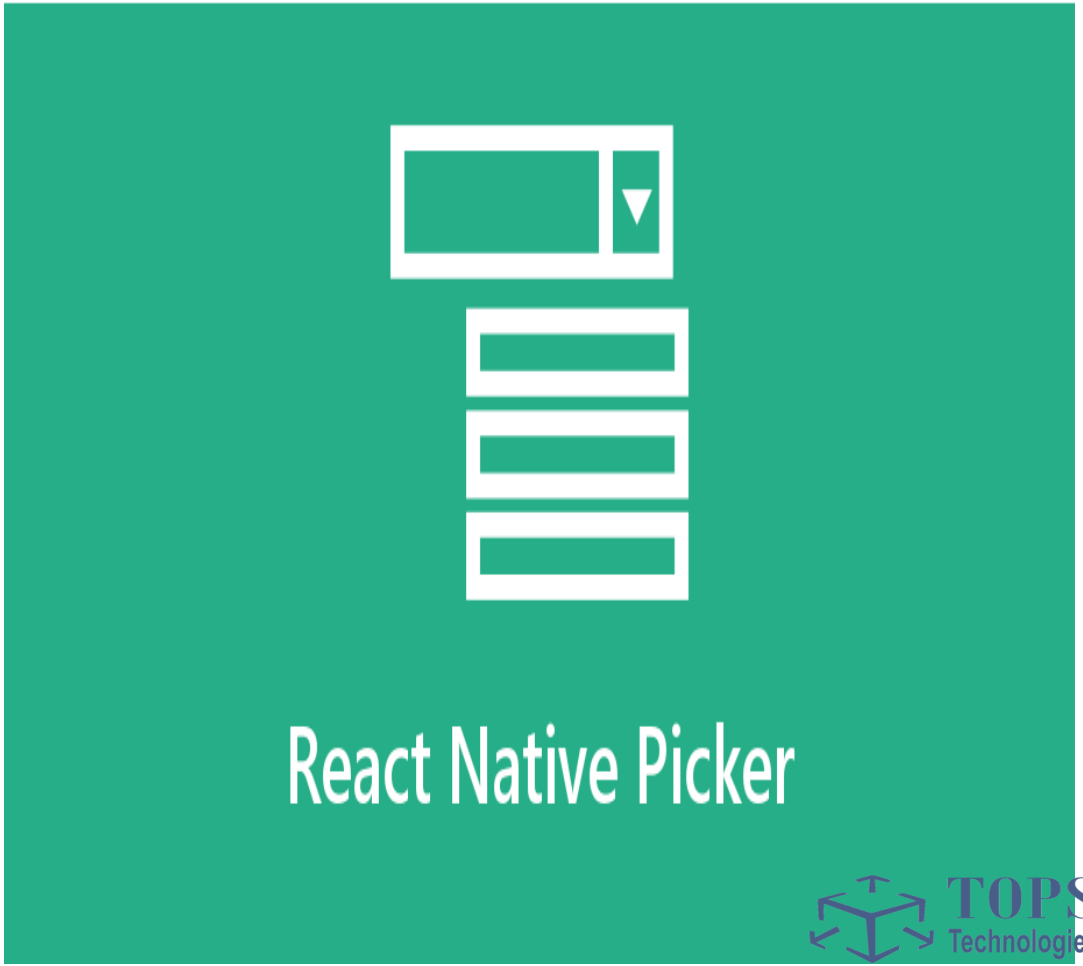
Working With Camera

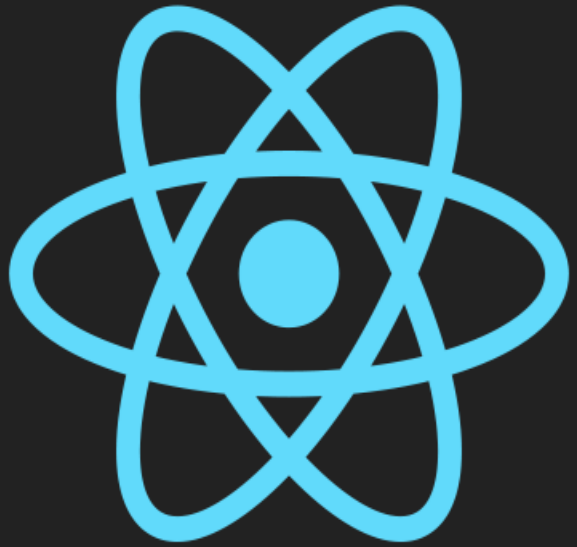
Example

Pickers

Examples:

1. Date
2. Time
3. Calendar
4. File





Module - 5

React Native Database

React Native Databases





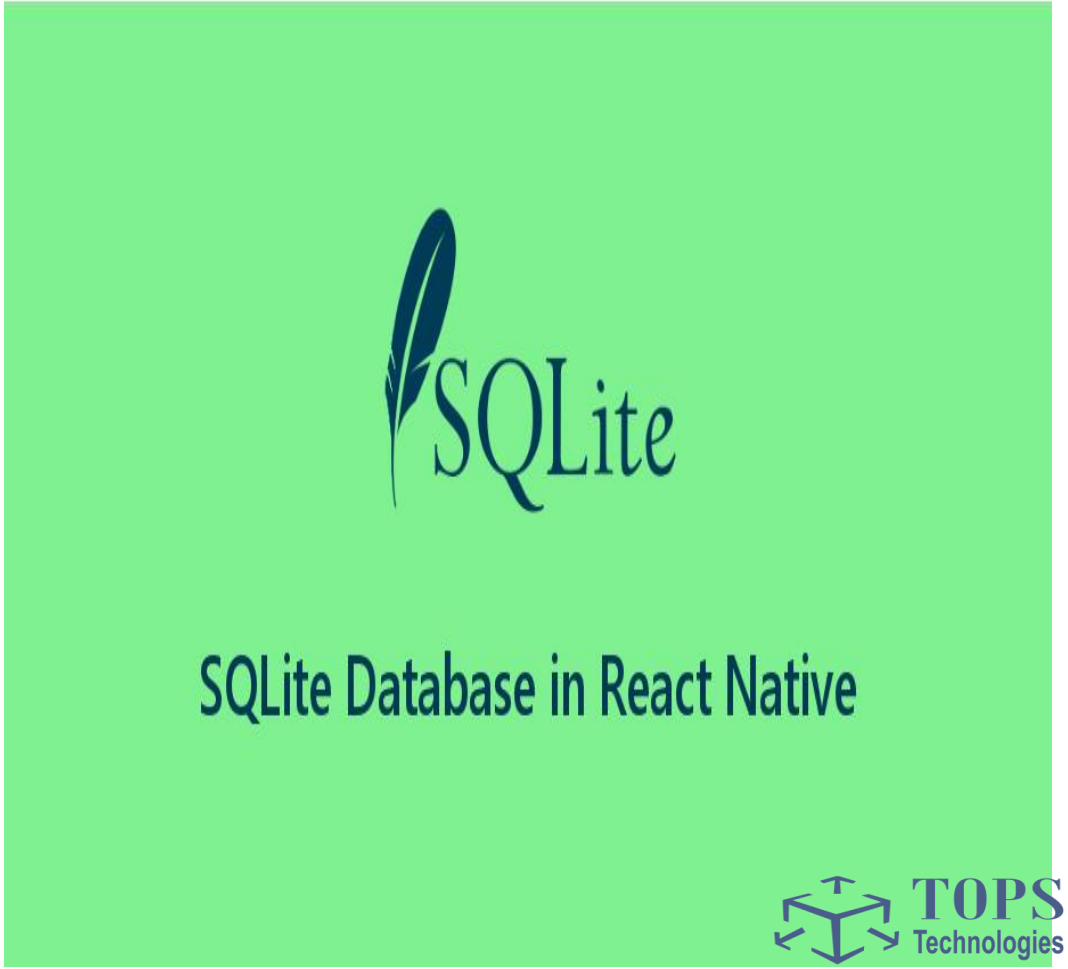
TOPS Technologies

Async Storage

Example

SQLite(CRUD)

Example

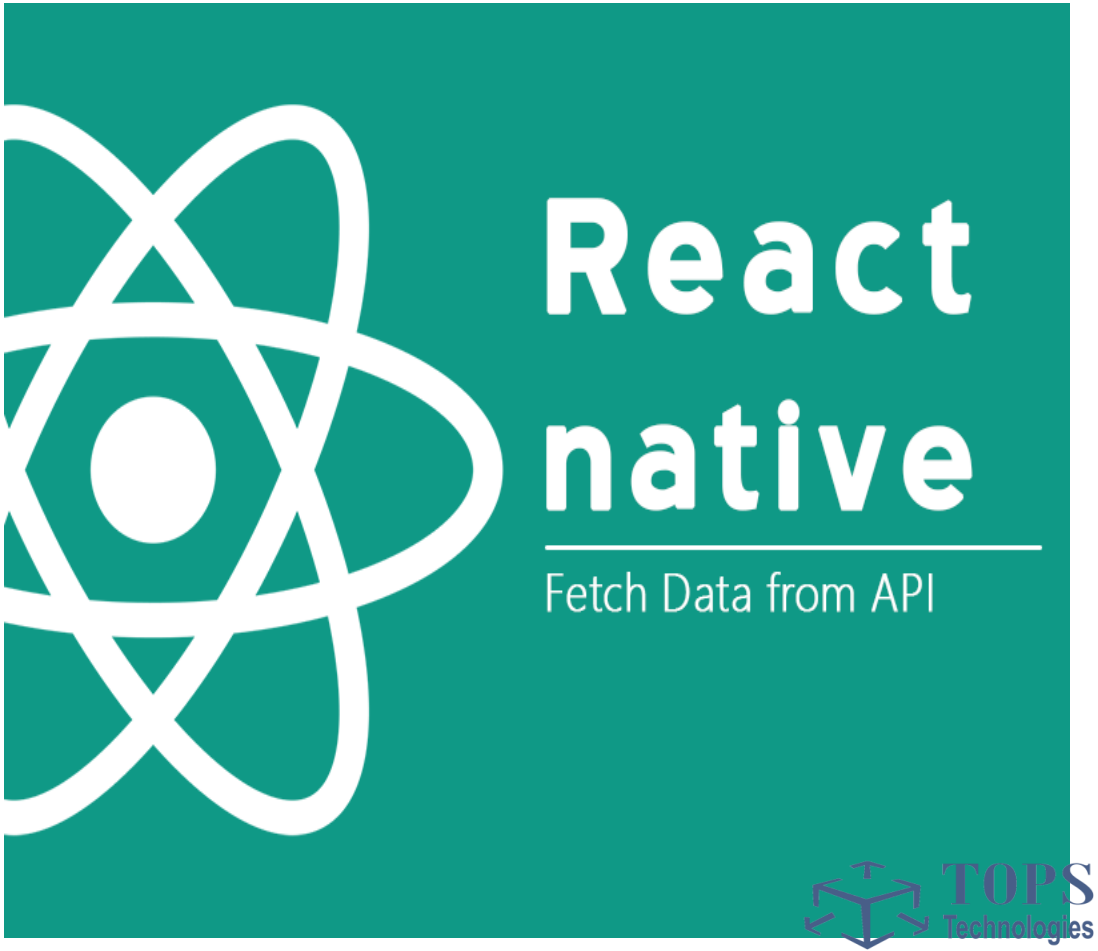


Realm(CRUD)

Example

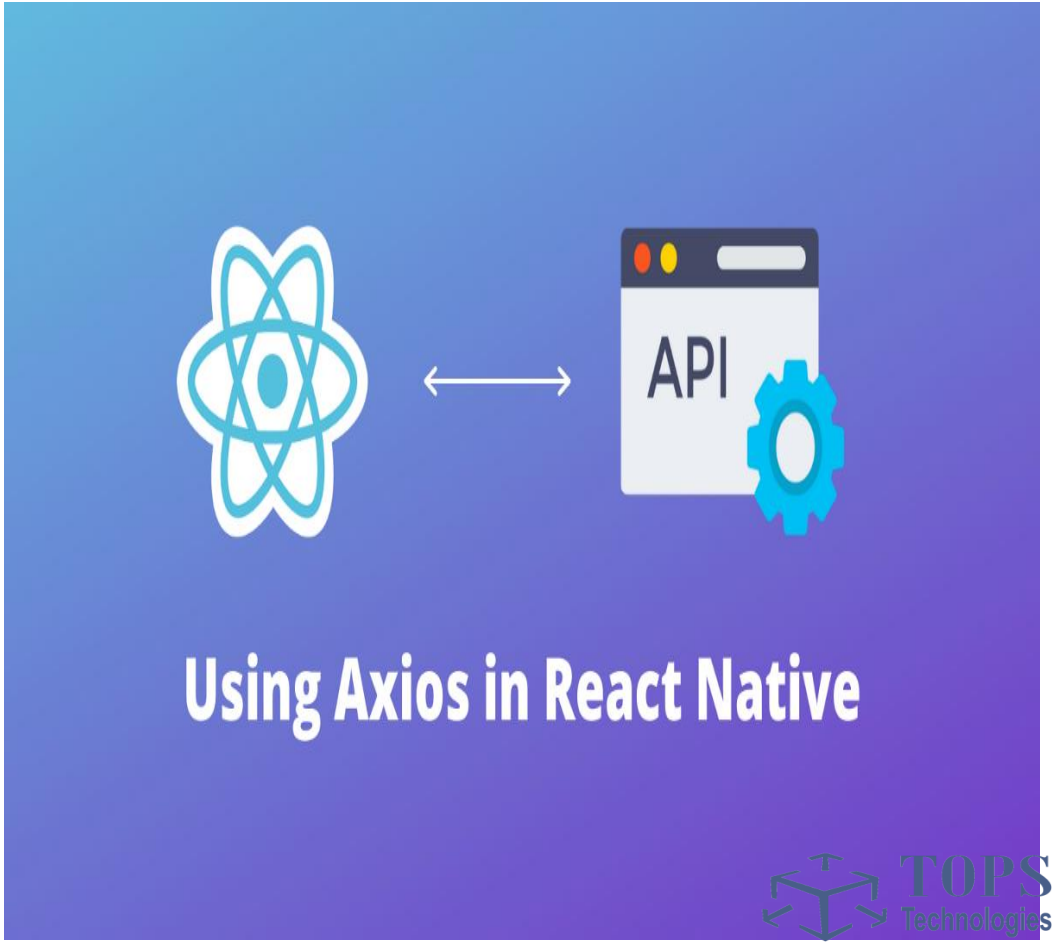
Fetch Data

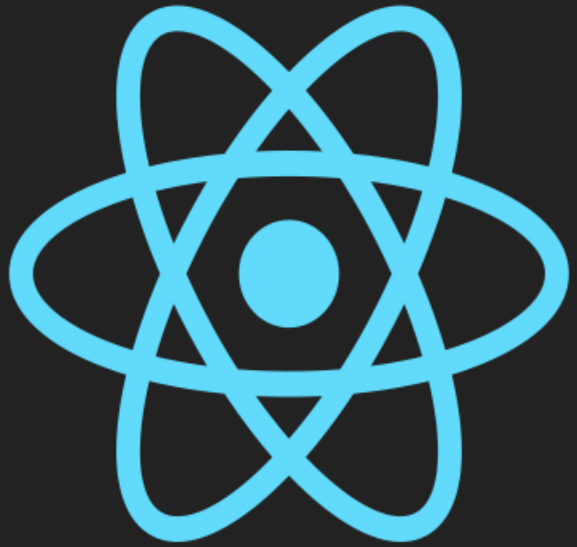
Example



Using Axios

Example



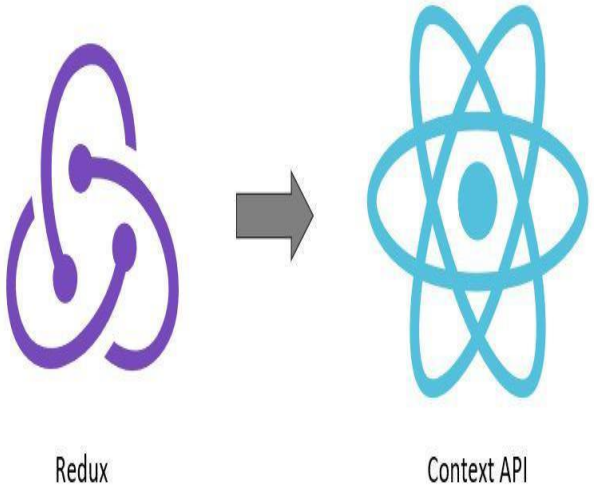


Module - 6

React Native Advance Development

What is State..?

- Anything that changes over time is known as state.
- If we had a Counter app, the state would be the counter itself.
- If we had a to-do app, the list of to-dos would change over time, so this list would be the state.
- Even an input element is in a sense a state, as it over time as the user types into it.



REDUX

Introduction

TOPS Technologies

Redux Basics

What is Redux? “**Redux** is a predictable state container for JavaScript apps. It helps you write applications that behave consistently, run in different environments.” this means the entire data flow of the app is handle within a single container while persisting previous state.

Redux can be broken down into few sections while building the application which are listed below.

Redux Basics

1. **Actions: “**are payloads of information that send data from your application to your store. They are the *only* source of information for the store.” this means if any state change necessary the change required will be dispatched through the actions.
2. **Reducers: “**Actions describe the fact that *something happened,* but don’t specify how the application’s state changes in response. This is the job of reducers.” when an action is dispatched for state change its the reducers duty to make the necessary changes to the state and return the new state of the application.
3. **Store:** with help of reducers a store can be created which holds the entire state of the application it is recommended to use a single store for the entire application than having multiple stores which will violate the use of redux which only has a single store.
4. **Components:** this is where the UI of the application is kept.

Redux Principles

Redux can be described in three fundamental principles:

1. **Single source of truth**

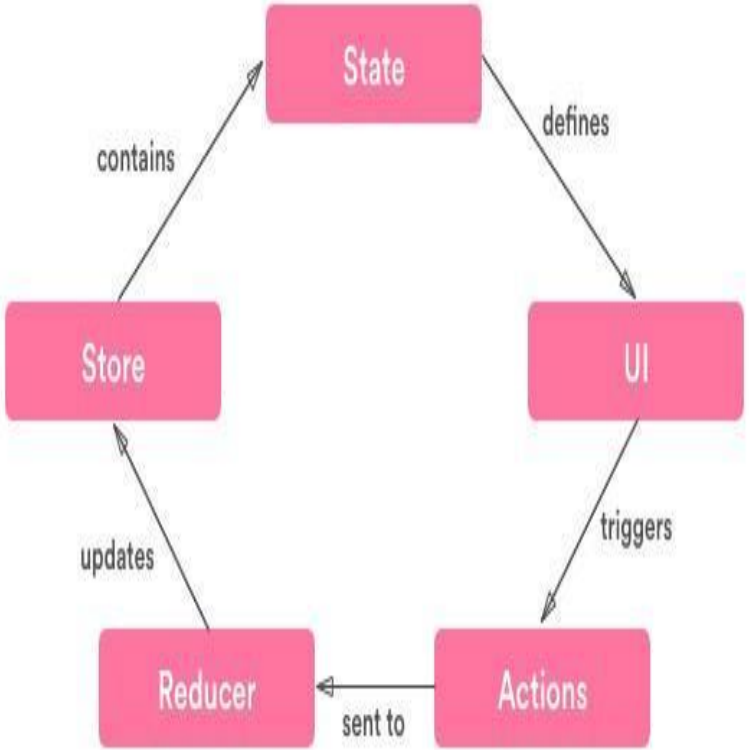
The global state of your application is stored in an object tree within a single store.
This makes it easy to create universal apps, as the state from your server can be serialized and hydrated into the client with no extra coding effort. A single state tree also makes it easier to debug or inspect an application; it also enables you to persist your app's state in development, for a faster development cycle. Some functionality which has been traditionally difficult to implement - Undo/Redo, for example - can suddenly become trivial to implement, if all of your state is stored in a single tree.

Redux Principles

2. **State is read-only**

The only way to change the state is to emit an action, an object describing what happened. This ensures that neither the views nor the network callbacks will ever write directly to the state. Instead, they express an intent to transform the state. Because all changes are centralized and happen one by one in a strict order, there are no subtle race conditions to watch out for. As actions are just plain objects, they can be logged, serialized, stored, and later replayed for debugging or testing purposes.

- Changes are made with pure functions
- To specify how the state tree is transformed by actions, you write pure reducers.
- Reducers are just pure functions that take the previous state and an action, and return the next state.
- Remember to return new state objects, instead of mutating the previous state.
- You can start with a single reducer, and as your app grows, split it off into smaller reducers that manage specific parts of the state tree.
- Because reducers are just functions, you can control the order in which they are called, pass additional data, or even make reusable reducers for common tasks such as pagination.



Why Use React Redux?

- Redux itself is a standalone library that can be used with any UI layer or framework, including React, Angular, Vue, Ember, and vanilla JS.
- Although Redux and React are commonly used together, they are independent of each other.
- If you are using Redux with any kind of UI framework, you will normally use a "UI binding" library to tie Redux together with your UI framework, rather than directly interacting with the store from your UI code.
- React Redux is the official Redux UI binding library for React.
- If you are using Redux and React together, you should also use React Redux to bind these two libraries.

Redux Implementation

Steps for Implementing Redux in React Native app

We will follow these step-by-step instructions to create our React Native with Redux

Step 1: Create a Basic React Native app

Step 2: Running app on device

Step 3: Add simple counter into the App.js.

Redux Implementation

Step 4: Install the necessary packages to connect your app with redux.

Step 5: Create the necessary folders inside Root.

Step 6: Create Actions and Reducer function.

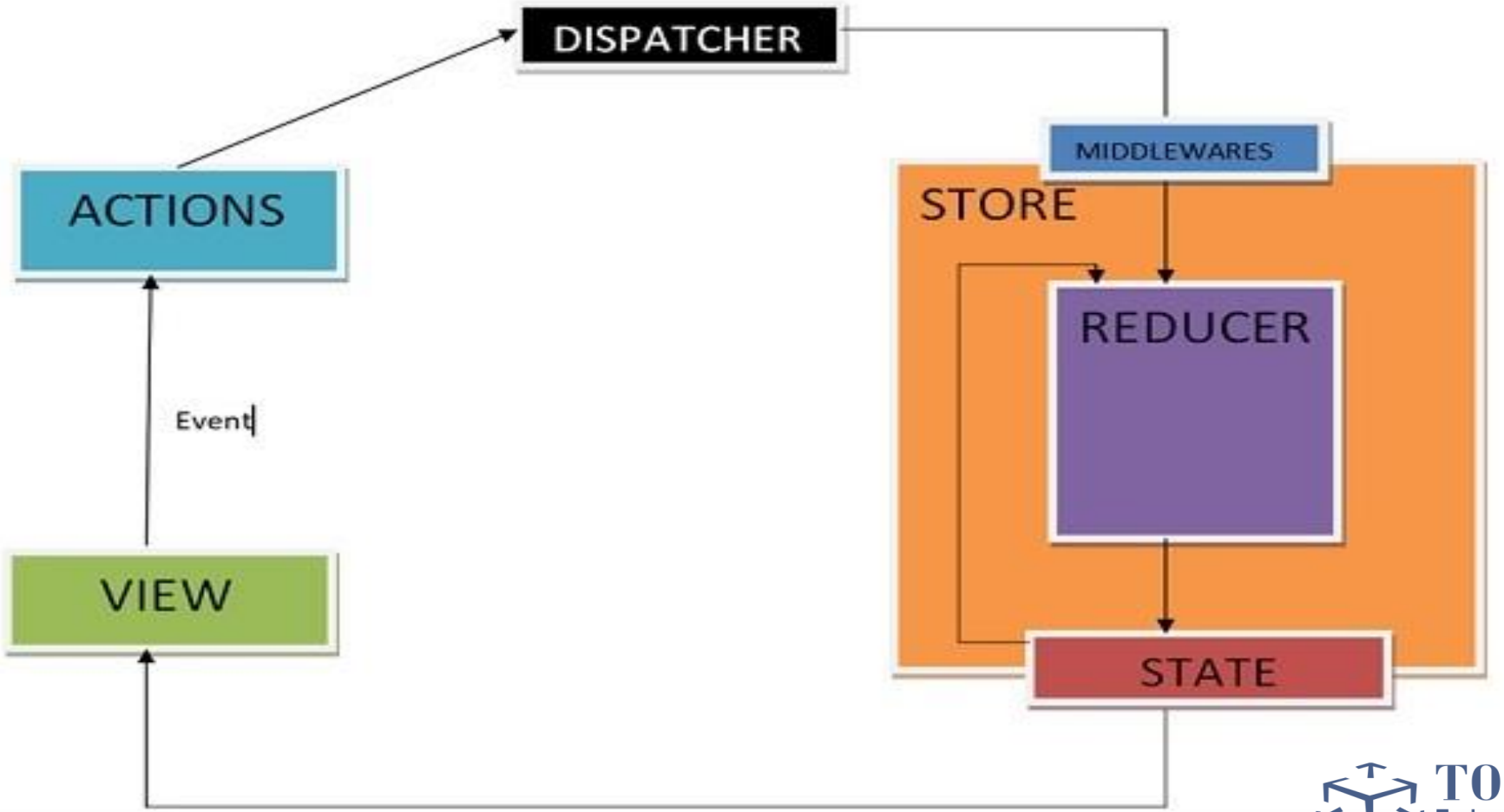
Step 7: Create a Redux Store.

Step 8: Pass the Store to the React Native app.

Step 9: Connect React Native app to Redux store.

Step 10: Test your app

Middleware



Middleware

- You've seen middleware in action in the Async Actions example.
- If you've used server-side libraries like Express and Koa, you were also probably already familiar with the concept of middleware.
- In these frameworks, middleware is some code you can put between the framework receiving a request, and the framework generating a response.
- For example, Express or Koa middleware may add CORS headers, logging, compression, and more.
- The best feature of middleware is that it's composable in a chain. You can use multiple independent third-party middleware in a single project.
- Redux middleware solves different problems than Express or Koa middleware, but in a conceptually similar way.
- It provides a third-party extension point between dispatching an action, and the moment it reaches the reducer.
- People use Redux middleware for logging, crash reporting, talking to an asynchronous API, routing, and more.

Let us understand how the react-redux works through the below diagram −

1. STORE − Stores all your application state as a JavaScript object
2. PROVIDER − Makes stores available
3. CONTAINER − Get apps state & provide it as a prop to components
4. COMPONENT − User interacts through view component
5. ACTIONS − Causes a change in store, it may or may not change the state of your app
6. REDUCER − Only way to change app state, accept state and action, and returns updated state.

Example React-Redux

Integrating Map API

TOPS Technologies

React Native Map

React Native Map Example is to integrate the Google Map into your React Native Application.

To integrate the Map in our example we will use a very good library called `react-native-maps`.

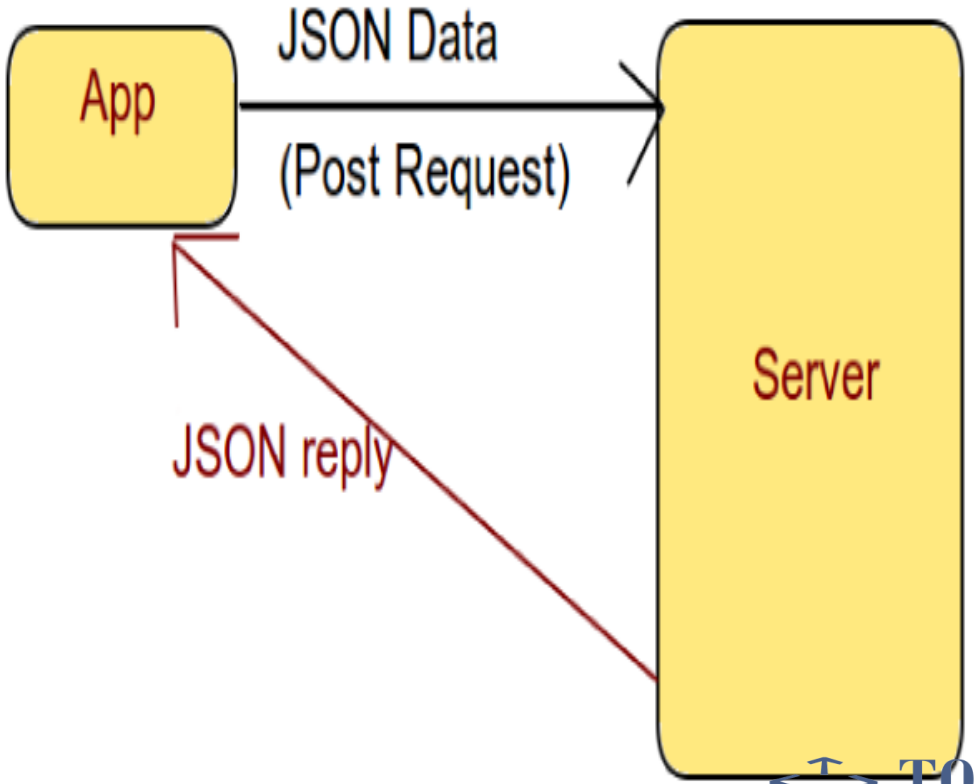
Which provides MapView component which is very easy to use.

Example ReactNativeMap

Integrating Data

Using API

Example API integration



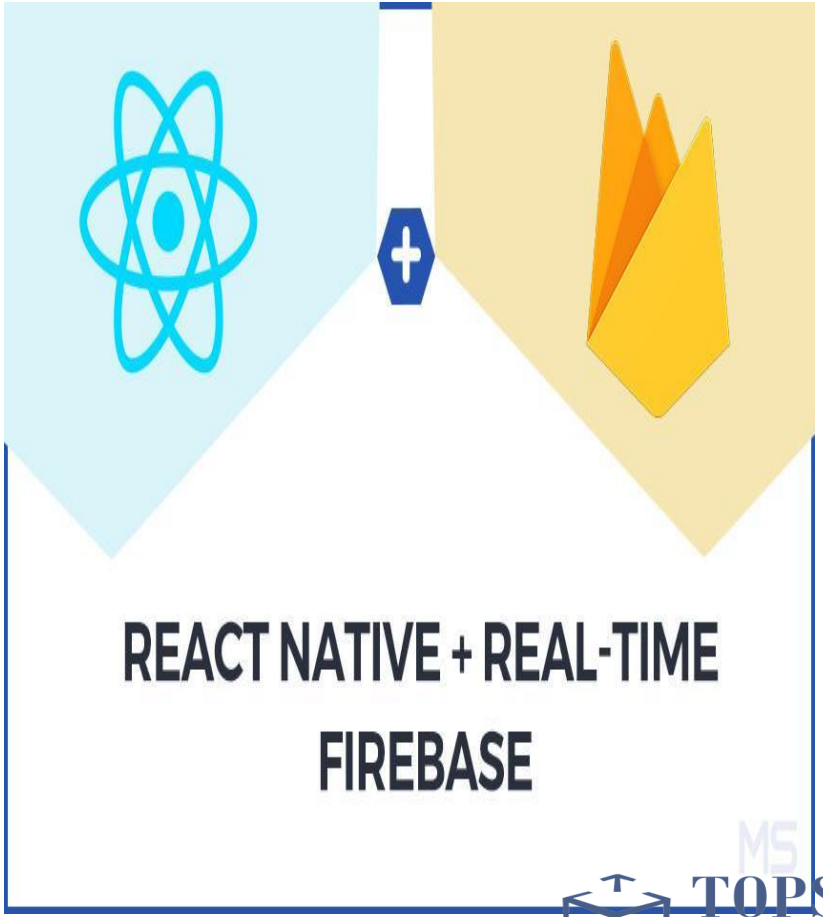
First of all, we need to install some dependency after creating a simple React application.

```
npm install redux react-redux redux-thunk
redux-form
```

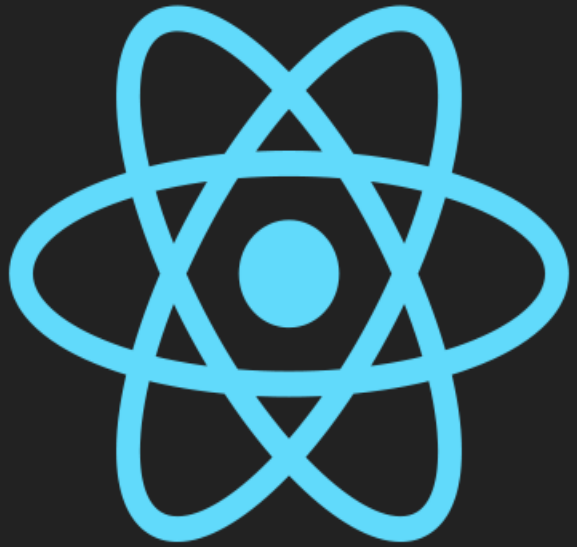
Example Wire Up Redux-Form with Form Inputs

Integrating Firebase with React Native

(Example)



TOPS Technologies



Module - 7

React Native Publishing App

Running your app on iOS devices

A Mac is required in order to build your app for iOS devices. Alternatively, you can refer to build your app using Expo CLI, which will allow you to run your app using the Expo client app.

npm install -g expo-cli

Install the Expo client app on your iOS or Android phone and connect to the same wireless network as your computer.

On Android, use the Expo app to scan the QR code from your terminal to open your project.

On iOS, use the built-in QR code scanner of the Camera app.

Running your app on Android devices

1. Enable Debugging over USB

To enable USB debugging on your device, you will first need to enable the "Developer options" menu by going to Settings → About phone → Software information and then tapping the Build number row at the bottom seven times. You can then go back to Settings → Developer options to enable "USB debugging".

2. Plug in your device via USB

Now check that your device is properly connecting to ADB, the Android Debug Bridge, by running adb devices.

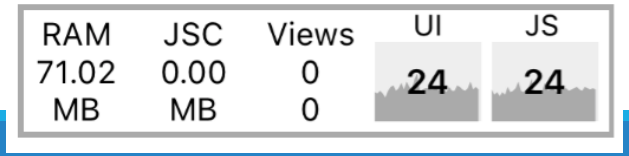
3. Run your app

$ npx react-native run-android

Performance

A compelling reason for using React Native instead of WebView-based tools is to achieve 60 frames per second and a native look and feel to your apps.

Where possible, we would like for React Native to do the right thing and help you to focus on your app instead of performance optimization, but there are areas where we're not quite there yet, and others where React Native (similar to writing native code directly) cannot possibly determine the best way to optimize for you and so manual intervention will be necessary. We try our best to deliver buttery-smooth UI performance by default, but sometimes that isn't possible.



TOPS Technologies

Deploying React Native Apps to App Store and Play Market

Gather a required information for both Apps Store and Play Market

Here is the main information you need to deploy React Native app.

1. Name of the app
2. Short description

How to submit React Native app to iOS App Store

1. Setting up the application.

2. Setting up to run the application on real devices for development.

3. Setting up to submit the application to App Store

Setting up the application

1. Create a developer account in https://developer.apple.com. Keep that credential. You will need it for all the following steps whenever Apple asks you to sign in.
2. Go to “Certificates, Identifiers & Profiles” section.
3. Create an application at “Identifiers” > “App IDs” section. Forms should be easy to understand. One thing to note is the bundle id and make sure your bundle id in RN is the same as this id.

Setting up to run the application on real devices for development

1. Plug your device → Open iTunes → Click on “Serial Number” to see UDID
2. Register your devices at “Devices” > “iPhone” section.
3. developer.apple.com → Provision Profiles “All” → Click “+” sign → Choose “iOS App Development” → Choose the App ID that you want to generate the profile for → Select the certificates (sidenote: certificates are tied to your mac and I select all) → Select all the devices you want to run the development on (Usually, I select all) → Give the name that you can recognize.
4. Done. You have added devices and you now get a provisioning profile to run as development.

Setting up to run the application on real devices for development

5. Now is the tricky part. I could not choose “Automatically manage signing” option in my Xcode. I have to choose Manual Signing. So, deselect “Automatically manage signing”.
6. In Signing (debug) → Provisioning Profile → (Import) and choose the above mentioned created file.
7. Now you can run your app as development on your device while connected to your Mac.

Setting up to submit the application to App Store

1. Create another provision file as above but choose “Distribution”.
2. I could not choose “Automatically manage signing” option in my Xcode. I have to choose Manual Signing. So, deselect “Automatically manage signing”.
3. In Signing (Release) → Provisioning Profile → (Import) and choose the above mentioned created file.
4. !Important! Make sure for every build, you increase the build number and they are not the same. If the build number is the same, you will have to do all the process again and waste time.
5. Product → Scheme → Edit Scheme → Run → Info → Build Configuration → Change to “Release”
6. Product → Clean (will take about 1 min)

TOPS Technologies

Setting up to submit the application to App Store

7. Product → Build (will take about 5 mins)
8. Product → Archive (will take about 5 mins)
9. Then a popup will appear. If no, Window → Organizer
10. You should see a list of applications with build numbers. Click “Distribute app”
11. Done.

How to submit React Native app to Android Play Store

A concise list of deployment steps with the assumption that :

- Android Studio is installed and that
- your native project is set up already through `react-native init YourProject` and contains the android folder with build/gradle files

***TIP:** if for some reason, the below steps don't seem to work, try to reset all the project dependencies
1. deleting the ***package-lock.json*** file and *node-modules* folder;
2. then re-install node modules using `npm install`.

TOPS Technologies

1. In android/app/build.gradle (Module: app) in **defaultConfig** object increment

   versionCode and versionName only.

2. After build version is changed you need to sync build.gradle file with project:

   go to File > Sync Project with Gradle Files
   - Go to Build > Rebuild Project

3. in your emulator/physical USB device **make sure any old app version are uninstalled/deleted** as this often produces conflicts when building new version on the device.
4. To rebuild, first, start rebuilding the JSX part (react-native code).

5. now go to play.google.com/apps/publish console, press CREATE APPLICATION, or ‘Manage’ existing application to update.

6. go to Release management > App releases > Alpha or Beta or Production section, press `create release/edit release`.

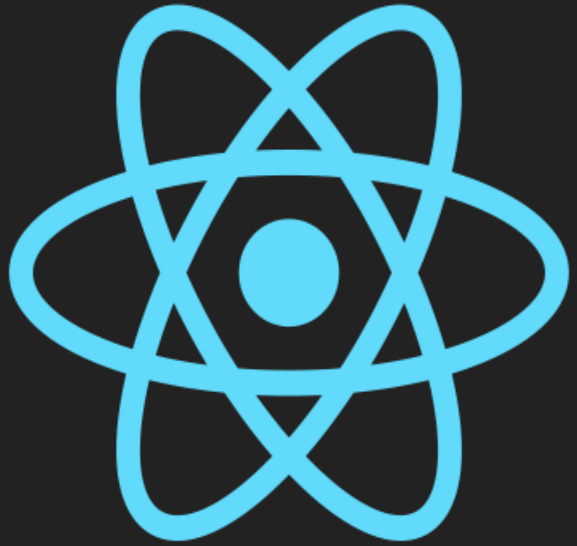
7. **Upload the app-release.apk** from the correct folder (react-native generates one, android studio generates one as well, but in a different locations, see step 6) choose Alpha, Beta or Production sections. You can always ‘upgrade’ alpha into Beta, into Production.

8. give reason for release of new version description between the <en-GB> </en-GB> tags at the bottom of page.

9. ‘Save’ and then ‘Review’ .

10. Fill out store presence, copyright and other tabs with grey circles until the sections reach publishable (green circle) status.

TOPS Technologies



Thank You