21 SEP

https://www.javatpoint.com/react-native-tutorial

Advantages of react native

1. ***Native Components:*** *We will need to write some platform specific code if we want to create native functionality which is not designed yet.*
2. ***Cross-Platform Usage:*** *Provide facility of "Learn once write everywhere", it works for both platform Android as well iOS devices.*

1. ***Class Performance:*** *The code written in React Native are compiled into native code, which enables it for both operating systems as well as it functions in the same way on both the platforms.*
2. ***JavaScript:*** *The JavaScript knowledge is used to build native mobile apps.*
3. ***Community:*** *The large community of React and React Native around helps us to find any answer we require.*

React native **Views**

The **View** is the fundamental component of React Native for building a user interface.

Props of view -

onStartShouldSetResponder	accessibilityLabel	accessibilityHint	hitSlop
nativeID	onAccessibilityTap	onLayout	onMagicTap
onMoveShouldSetResponder	onMoveShouldSetResponderCapture	onResponderGrant	onResponderMove
onResponderReject	onResponderRelease	onResponderTerminate	onResponderTerminationReque
accessible	onStartShouldSetResponderCapture	pointerEvents	removeClippedSubviews
style	testID	accessibilityComponentType	accessibilityLiveRegion
collapsable	importantForAccessibility	needsOffscreenAlphaCompositing	renderToHardwareTextureAndr
accessibilityRole	accessibilityStates	accessibilityTraits	accessibilityViewIsModal
accessibilityElementsHidden	accessibilityIgnoresInvertColors	shouldRasterizeIOS	

Props of Buttons

Prop	Type	Required	Description
onPress	function	yes	Call the handler when user clicks the button.
title	string	yes	Display the text inside the button.
accessibilityLabel	string	no	Display the text for blindness accessibility features.
color	Color	no	Set the background color of the Android button or set the color of iOS text.
disabled	bool	no	It disables all interactions for this component, if true.
textID	string	no	Used to locate this view in end-to-end tests.
hasTVPreferredFocus	bool	no	It preferred TV focus work only for Apple TV.

▼ ScrollView

The **ScrollView** is a generic scrollable container, which scrolls multiple child components and views inside it. In the ScrollView, we can scroll the components in both direction **vertically** and **horizontally.**

Props of scroll view

alwaysBounceVertical	onScroll	horizontal	
contentContainerStyle	scrollEnabled	bouncesZoom	zoomScale
onScrollBeginDrag	onContentSizeChange	maximumZoomScale	minimumZoomScale
onScrollBeginDrag	onContentSizeChange	maximumZoomScale	minimumZoomScale
onScrollEndDrag	centerContent	contentInset	refreshControl
pagingEnabled	scrollsToTop	snapToAlignment	showsHorizontalScrollIndicator
snapToStart	snapToEnd	indicatorStyle	showsHorizontalScrollIndicator

▼ ListView

React Native **ListView** is a view component which contains the list of items and displays in a vertical scrollable list.

```
<ListView
        dataSource={this.state.dataSource}
        renderRow={
         (rowData) =>  <Text style={{fontSize: 20}}>{rowData}</Text>
      }
/>
```

▼ FlatList

The **FlatList** component displays the similar structured data in a **scrollable** list. It works well for large lists of data where the number of list items might change over time. The FlatList shows only those renders elements which are currently displaying on the screen, not all the elements of the list at once.

```
<FlatList
     data={[
        {key: 'Android'},{key: 'iOS'}, {key: 'Java'},{key: 'Swift'},
        {key: 'Php'},{key: 'Hadoop'},{key: 'Sap'},
        {key: 'Python'},{key: 'Ajax'}, {key: 'C++'},
        {key: 'Ruby'},{key: 'Rails'},{key: '.Net'},
        {key: 'Perl'},{key: 'Sap'},{key: 'Python'},
        {key: 'Ajax'}, {key: 'C++'},{key: 'Ruby'},
        {key: 'Rails'},{key: '.Net'},{key: 'Perl'}
       ]}
           renderItem={({item}) =>
            <Text style={styles.item}
            onPress={this.getListViewItem.bind(this, item)}>{item.key}
              </Text>}
          ItemSeparatorComponent={this.renderSeparator}
    />
```

▼ SectionList

The React Native **SectionList** component is a list view component which sets the list of data into broken logical section. The broken data can be implemented using its section header prop **renderSectionHeader**.

Props of SectionList

sections	renderItem	initialNumToRender	keyExtractor
renderSectionHeader	renderSectionFooter	onRefresh	inverted
extraData	onEndReached	keyExtractor	legacyImplementation

onViewableItemsChanged	refreshing	removeClippedSubviews	ListHeaderComponent
SectionSeparatorComponent	stickySectionHeadersEnabled	onEndReachedThreshold	ListEmptyComponent

```
<SectionList
     sections={[
         {title: 'A', data: ['ALTERED','ABBY','ACTION U.S.A.','AMUCK','ANGUISH']},
         {title: 'B', data: ['BEST MEN','BEYOND JUSTICE','BLACK GUNN','BLOOD RANCH','BEASTIES']},
         {title: 'C', data: ['CARTEL', 'CASTLE OF EVIL', 'CHANCE', 'COP GAME', 'CROSS FIRE',]},
        ]}
```

▼ Touchable

Touchable components provide the capability to capture the tapping functionality.

Props

Props	Type	Required	Platform	Description
activeOpacity	number	no		It determines the opacity of wrapped view when it is touched.
tvParallaxProperties	object	no	iOS	It is an object with property which is used to control the Apple TV parallax effects.
hasTVPreferredFocus	bool	no	iOS	It focuses TV preferred, it works on Apple TV only.

▼ Activity Indicator

ActivityIndicator is used to display a circular loading indicator.

Props

Props	Description
animating	Option to show the indicator (bydefault it is true) or hide it (false).
size	Set the size of indicator ('small','large', number). The default size is small. Number format support only in android.
color	Set the foreground color of the spinner (default is gray).
hidesWhenStopped	It provides an option to show or hide the indicator when there is no animating (true by default).

```
<ActivityIndicator size="small" color="#44ff00" />
        <ActivityIndicator size="large" color="#rtwrw" />
```

▼ Status Bar

StatusBar is a component which is used to decorate status bar of the app. It is used by importing the StatusBar component from the react-native library. We can use multiple StatusBar at the same time.

```
<View>
   <StatusBar
     backgroundColor = "#b3e6ff"
     barStyle = "dark-content"
   />
</View>
```

```
<View>
  <StatusBar
     backgroundColor = "#b3e6ff"
     barStyle = "dark-content"
```

```
    />
    <View>
      <StatusBar
         hidden={route.statusBarHidden} />
    </View>
  </View>
```

React Native StatusBar Props

Props	Description
animated	A status bar is animated if its property is changed. It supports backgrondColor, hidden, and barStyle.
barStyle	It sets the color of status bar text.
hidden	It is used to hide and show the status bar. By default, it is false. If hidden = {false} it is visible, if hidden = {true}, it hide the status bar.
backgroundColor	It sets the background color of the status bar.
translucent	When it is set of true, the app is built under the status bar.
showHideTransition	It displays the transition effect when showing and hiding the status bar. The default is 'fade'.
networkActivityIndicatorVisible	It checks the network activity indicator is visible or not. It supports in iOS.

React Native StatusBar Methods

setHidden	setBarStyle	setBackgroundColor
setNetworkActivityIndicatorVisible	setTranslucent	