

React Native

COMPONENTS

Components

- Components are the building blocks for a React Native application.
- A React Native user interface (UI) is specified by declaring components, possibly nested, and then those components are mapped to the native UI on the targeted platform.
- React Native has a number of core components that are commonly used in applications, either on their own or combined to build new components.

ActivityIndicator Component

- Displays a circular loading indicator.
- Different in design for IOS vs Android

IOS



Android



ActivityIndicator **Props:**

animating

Whether to show the indicator (true, the default) or hide it (false).

color

The foreground color of the spinner (default is gray).

size

Size of the indicator (default is 'small'). Passing a number to the size prop is only supported on Android. ios

Button Component

A basic button component that should render nicely on any platform. Supports a minimal level of customization.

If this button doesn't look right for your app, you can build your own button using **TouchableOpacity** etc.

Button **Props:**

`onPress`

Handler to be called when the user taps the button

`title`

Text to display inside the button

`color`

Color of the text (iOS), or background color of the button (Android)

`accessibilityLabel`

Text to display for blindness accessibility features

`disabled`

If true, disable all interactions for this component.

TouchableOpacity Component

- A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, dimming it.
- For Making component clickable, we wrap component with **TouchableOpacity**.

TouchableOpacity Component

Props:

`onPress`

Handler to be called when the user taps the button

`activeOpacity`

Determines what the opacity of the wrapped view should be when touch is active. Defaults to 0.2.

TouchableHighlight Component

- A wrapper for making views respond properly to touches. On press down, the opacity of the wrapped view is decreased, which allows the underlay color to show through, darkening or tinting the view.
- For Making component clickable, we wrap component with **TouchableOpacity**.

TouchableHighlight **Props:**

`onPress`

Handler to be called when the user taps the button

`activeOpacity`

Determines what the opacity of the wrapped view should be when touch is active.

`underlayColor`

The color of the underlay that will show through when the touch is active.
ios

etc....

ScrollView Component

- lets you specify different scroll directions for child content
- simply renders all its react child components at once. That makes it very easy to understand and use.
- On the other hand, this has a performance downside. Imagine you have a very long list of items you want to display

ScrollView **Props:**

horizontal

When true, the scroll view's children are arranged horizontally in a row instead of vertically in a column. The default value is false.

onScroll

Fires at most once per frame during scrolling.

scrollEnabled

When false, the view cannot be scrolled via touch interaction. The default value is true.

etc....

ScrollView:

Example code:

```
<ScrollView>
  <Text style={{fontSize:96}}>Scroll me plz</Text>
  <Image source={require('./img/favicon.png')} />
  <Image source={require('./img/favicon.png')} />
  <Image source={require('./img/favicon.png')} />
  <Image source={require('./img/favicon.png')} />
  <Image source={require('./img/favicon.png')} />
</ScrollView>
```

FlatList Component

- A performant interface for rendering simple, flat lists
- Fully cross-platform.
- Optional horizontal mode.
- Configurable viewability callbacks.
- Header support.
- Footer support.
- Separator support.
- Pull to Refresh.
- Scroll loading (Lazy Loading).
- ScrollToIndex support.

FlatList **Props:**

`data:[ReadOnlyArray]`

For simplicity, data is just a plain array.

`renderItem`

Takes an item from data and renders it into the list. Example usage:

`numColumns`

Multiple columns can only be rendered with `horizontal={false}` and will zig-zag like a `flexWrap` layout.

etc.

FlatList:

Minimal Example:

```
<FlatList
  data={[{key: 'a'}, {key: 'b'}]}
  renderItem={({item}) => <Text>{item.key}</Text>}
/>
```

WebView Component

- Renders web content in a native view.
- Can render HTML via URL
- Can load static HTML

WebView **Props:**

`Source:{uri:string} | {html:string}`

Loads static html or a uri (with optional headers) in the WebView.

`style`

The style to apply to the WebView.

`injectJavaScript`

Function that accepts a string that will be passed to the WebView and executed immediately as JavaScript.

`injectedJavaScript`

Set this to provide JavaScript that will be injected into the web page when the view loads.

Continue...

WebView **Props:**

onLoad

Function that is invoked when the WebView has finished loading.

etc..

Image Component

- A React component for displaying different types of images, including network images, static resources, temporary local images, and images from local disk, such as the camera roll.

Note: that for network and data images, you will need to manually specify the dimensions of your image!

Image **Props**:

`Source:{uri:string}`

The image source (either a remote URL or a local file resource).

`style`

The style to apply to the Image.

`resizeMode: 'cover' | 'contain' | 'stretch' | 'repeat' | 'center'`

Determines how to resize the image when the frame doesn't match the raw image dimensions.

Continue...

Image **Props**:

`resizeMode: 'cover'`

Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or larger than the corresponding dimension of the view (minus padding).

`resizeMode: 'contain'`

Scale the image uniformly (maintain the image's aspect ratio) so that both dimensions (width and height) of the image will be equal to or less than the corresponding dimension of the view (minus padding).

`resizeMode: 'stretch'`

Scale width and height independently, This may change the aspect ratio of the src.

Continue...

Image **Props**:

`resizeMode: repeat`

repeat: Repeat the image to cover the frame of the view. The image will keep its size and aspect ratio. (iOS only)

TextInput Component

- A foundational component for inputting text into the app via a keyboard.
- auto-correction
- auto-capitalization
- placeholder text
- different keyboard types, such as a numeric keypad.

TextInput **Props:**

keyboardType: default | numeric | email-address | phone-pad (Work cross platform)

Determines which keyboard to open, e.g.numeric

.

style

The style to apply to the TextInput.

placeholder

The string that will be rendered before text input has been entered.

onChangeText

Callback that is called when the text input's text changes. Changed text is passed as an argument to the callback handler. It has access to only value entered in text input

Continued...

TextInput **Props:**

onChange

Callback that is called when the text input's text changes. It has access to event object.

style

The style to apply to the TextInput.

maxLength


Limits the maximum number of characters that can be entered. Use this instead of implementing the logic in JS to avoid flicker.

etc..

React vs React Native Component reference guide






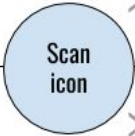

React	React Native
<code><div></code>	<code><View></code>
<code></code>	<code><Text></code>
<code>, </code>	<code><ListView></code>
<code></code>	<code><Image></code>

Today's **Exercise:**



Login with Facebook

Skip this step

Your Scans		
	Some title 08:00 AM, 6th Sep 2017	>
	Personal Info 08:00 AM, 26th Aug 2017	>
	Some title 07:40 PM, 21st Aug 2017	>
	Some title 08:00 AM, 6th Sep 2017	>
	Some title 08:00 AM, 6th Sep 2017	
	Some title 08:00 AM, 6th Sep 2017	

< Some selected Item


08:00 AM, 6th Sep 2017

Personal Info

Name

Manoj Nama

Email

manoj.nama@tothenew.com

Company

TO THE NEW

State

UP

Country

India

Thank You