React Native - Basics - Touch and Touchable

- Dr Nick Hayward

A brief intro to the basics of text input in React Native app development.

Contents

Intro

React Native includes various options for supporting touch interactions in an app's UI.

Various default components support tapping, swiping, scrolling &c, plus a *Gesture Responder System* to manage various stages of the lifecycle of such user gestures.

Basic button usage

As expected, React Native provides a default `Button` component, which accepts various *props* to help manage usage and interaction,

e.g.

```
<Button
        title='Basic Button'
        onPress={this._buttonPress}
/>
```

This will then render the expected UI button with applicable default colours and styling for Android and iOS.

- Android = blue, rounded rectangle with white text
- iOS = blue label

We can also customise the background colour for this default button with a simple `color` prop, e.g.

```
<Button
        onPress={this._buttonPress}
        title='Tap for Alert'
        color='cadetblue'
/>
```

So, an example button usage might be as follows,

```
export default class ButtonPress extends Component {
        _buttonPress() {
                Alert.alert('a button has been tapped...')
        }
        render() {
                return (
                        <View style={styles.container}>
                                <View style={styles.buttonBox}>
                                        <Button
                                                onPress={this._buttonPress}
                                                title='Tap for Alert'
                                                color='cadetblue'
                                        />
                                </View>
                        </View>
                );
        }
}
```

Touchable components

Touchable components allow developers to add touch events to a custom component. These components are able to capture tap gestures, and then fire a response in recognition of such events.

There is no default styling for such components, so a developer may customise them as desired to fit a given app.

component options

There is a selection of touchable components available for use with custom UI elements and components.

Your choice of component will largely depend upon the type of app you're creating, and the interaction feedback you want to convey to a user.

Options include highlight, opacity, no feedback, and custom options for a given platform.

We can also check certain explicit user interactions using available *props*, such as `onPress` or `onLongPress`.

e.g.

```
export default class TouchablePress extends Component {
        _touchablePress() {
                Alert.alert('test touchable tap fired...');
        }
        render() {
                return (
                        <View style={styles.container}>
                                <View style={styles.buttonBox}>
                                        <TouchableHighlight
                                                onPress={this._touchablePress}
                                                underlayColor='cadetblue'
                                                >
                                                <View style={styles.customButton}>
                                                        <Text>Try a Touchable
Highlight</Text>
                                                </View>
                                        </TouchableHighlight>
                                </View>
                                <View style={styles.contentBox}>

                                </View>
                        </View>
                );
```

```
        }
}
```

In this example, we're wrapping a basic view in a touchable component, which includes support for *highlight* by default. Then. we pass some *props* to this component to allow us to modify the reaction for a user's press, `onPress`, and a modified colour for the highlight itself.

For the props `onPress`, we can call a custom function, which simply displays an alert as feedback to the user for each touch event.