

声明，本文并非原创，本文属于优质好文，故转载了，[原文链接](#)

本文出自 [《React Native学习笔记》](#) 系列文章。

在做App开发过程中离不了的需要用户交互，说到交互，我们首先会想到的就是按钮了，在React Native中没有专门的按钮组件。为了让视图能够响应用的的点击事件，我们需要借助Touchablexxx组件，来包裹我们的视图。为什么说是Touchablexxx呢，因为它不只是一个组件，而是一组组件，一下四个组件都可以用来包裹视图来响应用户的点击事件。

- **TouchableWithoutFeedback**: 响应用户的点击事件，如果你想在处理点击事件的同时不显示任何视觉反馈，使用它是个不错的选择。
- **TouchableHighlight**: 在TouchableWithoutFeedback的基础上添加了当按下时背景会变暗的效果。
- **TouchableOpacity**: 相比TouchableHighlight在按下去会使背景变暗的效果，TouchableOpacity会在用户手指按下时降低按钮的透明度，而不会改变背景的颜色。
- **TouchableNativeFeedback**: 在Android上还可以使用TouchableNativeFeedback，它会在用户手指按下时形成类似水波纹的视觉效果。注意，此组件只支持Android。

心得：以上四个组件，其中TouchableHighlight、TouchableOpacity以及TouchableNativeFeedback都是在TouchableWithoutFeedback的基础上做了一些扩展，我们从它们的源码中可以看出：

#### TouchableHighlight:

```
var TouchableHighlight = React.createClass({
  propTypes: {
    ...TouchableWithoutFeedback.propTypes,
```

#### TouchableOpacity:

```
var TouchableOpacity = React.createClass({
  mixins: [TimerMixin, Touchable.Mixin, NativeMethodsMixin],

  propTypes: {
    ...TouchableWithoutFeedback.propTypes,
```

#### TouchableNativeFeedback:

```
var TouchableNativeFeedback = React.createClass({
  propTypes: {
    ...TouchableWithoutFeedback.propTypes,
```

因为TouchableWithoutFeedback有其它三个组件的共同属性，所以我们先来学习一下TouchableWithoutFeedback。

## TouchableWithoutFeedback使用详解

TouchableWithoutFeedback一个Touchable系列组件中最基本的一个组价，只响应用户的点击事件不会做任何UI上的改变，在使用的过程中需要特别留意。

提示：无论是TouchableWithoutFeedback还是其他三种Touchable组件，都是在根节点都是只支持一个组件，如果你需要多个组件同时相应单击事件，可以用一个View将它们包裹着，它的这种根节点只支持一个组件的特性和ScrollView很类似。

接下来让我们来看一下，TouchableWithoutFeedback有哪些常用的属性：

### TouchableWithoutFeedback常用的属性

说到常用的属性TouchableWithoutFeedback首先要提到的就是 `onPress` 了。

#### `onPress` function

当触摸操作结束时调用，但如果被取消了则不调用（譬如响应者被一个滚动操作取代）。

心得：`onPress` 可谓是Touchable系列组件的最常用的属性之一了，如果你要让视图响应用户的单击事件，那么用 `onPress` 就可以了。

接下来呢，我们就来实现一个统计按钮单击次数的例子。

```
<TouchableWithoutFeedback
  onPress={() => {
    this.setState({count: this.state.count+1})
  }}
>
  <View style={styles.button}>
    <Text style={styles.buttonText}>
      我是TouchableWithoutFeedback,单击我
    </Text>
  </View>
</TouchableWithoutFeedback>
<Text style={styles.text}>您单击了:{this.state.count}次</Text>
```

[下载源码](#)

Carrier 5:07 PM

我是TouchableWithoutFeedback,单击我  
您单击了:0次

### onLongPress function

当用户长时间按压组件(长按效果)的时候调用该方法。

心得: `onLongPress` 也是Touchable系列组件的最常用的属性之一，通常用于响应长按的事件，如长按列表弹出删除对话框等。

接下来呢，我们就来实现使用 `onLongPress` 属性来响应用户的长按事件。

```
<TouchableWithoutFeedback
  onPress={() => {
    this.setState({count: this.state.count + 1})
  }}
  onLongPress={() => {
    this.setState({countLong: this.state.countLong + 1})
    Alert.alert(
      '提示',
      '确定要删除吗?',
      [
        {text: '取消', onPress: () => console.log('Cancel Pressed'), style: 'cancel'},
        {text: '确实', onPress: () => console.log('OK Pressed')},
      ]
    )
  }}
>
  <View style={styles.button}>
    <Text style={styles.buttonText}>
      我是TouchableWithoutFeedback,单击我
    </Text>
  </View>
</TouchableWithoutFeedback>
<Text style={styles.text}>您单击了:{this.state.count}次</Text>
<Text style={styles.text}>您长按了:{this.state.countLong}次</Text>
```

[下载源码](#)

我是TouchableWithoutFeedback,单击我  
您单击了:0次  
您长按了:0次

我们在上面例子的基础上为Touchable设置了 `onLongPress` 属性，当用户长时间按压按钮是会弹出一个对话框。

心得：当我们没有对Touchable组件设置 `onLongPress` 属性而设置了 `onPress` 属性的时候，我们长按按钮之后会回调 `onPress` 方法。另外，我们也可以通过 `delayLongPress` 方法来设置从 `onPressIn` 被回调开始，到 `onLongPress` 被调用的延迟。

## disabled bool

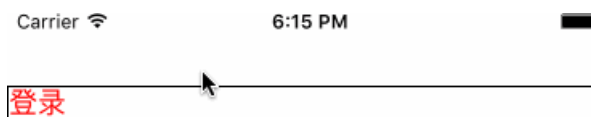
如果设为true，则禁止此组件的一切交互。

心得：`disabled` 也是Touchable系列组件的最常用的属性之一，通常用于禁止按钮相应用户的点击事件，比如，当用户单击按钮进行登录时，需要进行网络请求，在请求操作完成之前如果用户多次单击登录按钮我们通常不希望发起多次登录请求，这个时候就可以借助 `disabled` 的属性来禁用按钮的交互。

接下来呢，我们就来模拟用户登录的例子来介绍一下 `disabled` 的使用。

```
<TouchableWithoutFeedback
  disabled={this.state.waiting}
  onPress={() => {
    this.setState({text: '正在登录...', waiting: true})
    setTimeout(() => {
      this.setState({text: '网络不流畅', waiting: false})
    }, 2000);
  }}
>
  <View style={styles.button}>
    <Text style={styles.buttonText}>
      登录
    </Text>
  </View>
</TouchableWithoutFeedback>
<Text style={styles.text}>{this.state.text}</Text>
```

[下载源码](#)



在上面例子中我们模拟了用户登录的效果，默认状态下按钮是可以响应用户点击事件的，在正在登录过程中我们通过 `disabled` 属性来禁用了按钮，这时无无论是单击还是长按按钮都是没有任何响应的，在间隔2s后，我们又将按钮解除禁用，这时按钮又可以重新响应用户的点击事件了。当用户长时间按压按钮时会弹出一个对话框。

心得：有朋友问我，想禁用按钮，但是通过设置Touchable的 `accessible` 属性为false没有效果，这也是因为即使 `accessible` 为false的情况下，Touchable组件还是可以响应交互事件的，要想禁用Touchable的交互事件，只能通过 `disabled` 属性。

## onPressIn function与onPressOut function

这两个方法分别是当用户开始点击按钮时与点击结束后被回调。

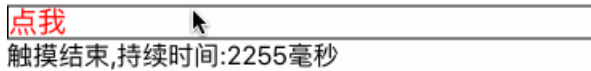
通过这两个方法我们可以计算出用户单击按钮所用的时长，另外也可以做一些其它个性化的功能。现在我们将通过一个例子来计算出用户点击按钮所用的时长。

```

<TouchableWithoutFeedback
  onPressIn={() => {
    this.setState({text: '触摸开始', startTime: new Date().getTime()})
  }}
  onPressOut={() => {
    this.setState({text: '触摸结束, 持续时间: ' + (new Date().getTime() - this.state.startTime) + '毫秒'})
  }}
>
  <View style={styles.button}>
    <Text style={styles.buttonText}>
      点我
    </Text>
  </View>
</TouchableWithoutFeedback>
<Text style={styles.text}>{this.state.text}</Text>

```

[下载源码](#)



在上述例子中我们记录下用户单击按钮的时间戳，当单击结束后我们获取当前时间减去刚单击时的时间，它们的差值就是用户单击按钮所用的时间了。

心得：另外我们也可以通过 `delayPressIn` 与 `delayPressOut` 两个方法来分别设置，从用户点击按钮到 `onPressIn` 被回调的延时与从点击结束到 `onPressOut` 被回调时的延时。

## TouchableHighlight使用详解

`TouchableHighlight` 是 `Touchable` 系列组件中比较常用的一个，它是在 `TouchableWithoutFeedback` 的基础上添加了一些UI上的扩展，既当手指按下时，该视图的不透明度会降低，同时会看到相应的颜色(视图变暗或者变亮)，从 `TouchableHighlight` 的源码中我们可以看出，其实这个颜色就是在 `TouchableHighlight` 的最外层添加了一个 `View`，通过改变这个 `View` 的背景色及透明度来达到这一效果。

```

render: function() {
  return (
    <View
      accessible={this.props.accessible !== false}
      accessibilityLabel={this.props.accessibilityLabel}
      accessibilityComponentType={this.props.accessibilityComponentType}
      accessibilityTraits={this.props.accessibilityTraits}
      ref={UNDERLAY_REF}
      style={this.state.underlayStyle}
      onLayout={this.props.onLayout}
      hitSlop={this.props.hitSlop}
      onStartShouldSetResponder={this.touchableHandleStartShouldSetResponder}
      onResponderTerminationRequest={this.touchableHandleResponderTerminationRequest}
      onResponderGrant={this.touchableHandleResponderGrant}
      onResponderMove={this.touchableHandleResponderMove}
      onResponderRelease={this.touchableHandleResponderRelease}
      onResponderTerminate={this.touchableHandleResponderTerminate}
      testID={this.props.testID}>
      {React.cloneElement(
        React.Children.only(this.props.children),
        {
          ref: CHILD_REF,
        }
      )}
      {Touchable.renderDebugView({color: 'green', hitSlop: this.props.hitSlop})}
    </View>
  );
}

```

### TouchableHighlight所扩展出来的属性

#### activeOpacity number

我们可以通过 `activeOpacity` 来设置 `TouchableHighlight` 被按下时的不透明度，从 `TouchableHighlight` 的源码中可以看出，它的默认不透明度为 0.85，我们可以根据需要进行调节。

```
var DEFAULT_PROPS = {
  activeOpacity: 0.85,
  underlayColor: 'black',
};
```

## underlayColor color

我们可以通过 `underlayColor` 属性来设置 `TouchableHighlight` 被按下去的颜色，默认状态下为 `black` 黑色。

## onHideUnderlay function

当衬底(也就是上文讲到的最外层的View)被隐藏的时候调用。

心得，通常情况下，当手指结束点击时衬底会被隐藏。

## onShowUnderlay function

当衬底(也就是上文讲到的最外层的View)显示的时候调用。

心得，通常情况下，当手指刚开始点击时衬底会显示。

## style View#style

因为 `TouchableHighlight` 的最外层个添加了一个View，所以我们可以设置这个View的样式来做出一个形形色色的按钮。

接下来，我们通过一个例子来看一下这些属性的使用。

```
<TouchableHighlight
  style={{marginTop:20}}
  activeOpacity={0.7}
  underlayColor='green'
  onHideUnderlay={()=>{
    this.setState({text:'衬底被隐藏'})
  }}
  onShowUnderlay={()=>{
    this.setState({text:'衬底显示'})
  }}
  onPress={()=>{
  }}
>
  <View style={styles.button}>
    <Text style={styles.buttonText}>
      TouchableHighlight
    </Text>
  </View>
</TouchableHighlight>
<Text style={styles.text}>{this.state.text}</Text>
```

[下载源码](#)

## TouchableHighlight

衬底被隐藏

# TouchableOpacity使用详解

`TouchableOpacity` 也是Touchable系列组件中比较常用的一个，它是在 `TouchableWithoutFeedback` 的基础上添加了一些UI上的扩展，但这些扩展相比 `TouchableHighlight` 少了一个额外的颜色变化。它是通过在按下去改变视图的不透明度来表示按钮被点击的。

## TouchableOpacity所扩展出来的属性

在扩展属性方面 `TouchableOpacity` 相比 `TouchableHighlight`，就少了很多，只有一个 `activeOpacity`，来设置按下去的透明度。

## activeOpacity number

同，`TouchableHighlight` 的`activeOpacity`。

另外我们也可以通过 `TouchableOpacity` 的 `setOpacityTo(value, duration)` 方法来动态修改 `TouchableOpacity` 被按下去的不透明度。

## TouchableNativeFeedback使用详解

为了支持Android5.0新增的触控反馈，React Native加入了 `TouchableNativeFeedback` 组件，`TouchableNativeFeedback` 在 `TouchableWithoutFeedback` 所支持的属性的基础上增加了按下去的水波纹效果。我们可以通过 `background` 属性来自定义原生触摸操作反馈的背景。

### TouchableNativeFeedback所扩展出来的属性

#### background backgroundPropTypes

决定在触摸反馈的时候显示什么类型的背景。它接受一个有着`type`属性和一些基于`type`属性的额外数据的对象。推荐使用以下的静态方法之一来创建这个对象：

- 1) `TouchableNativeFeedback.SelectableBackground()` - 会创建一个对象，表示安卓主题默认的对于被选中对象的背景。(?`android:attr/selectableItemBackground`)
- 2) `TouchableNativeFeedback.SelectableBackgroundBorderless()` - 会创建一个对象，表示安卓主题默认的对于被选中的无边框对象的背景。(?`android:attr/selectableItemBackgroundBorderless`)。只在Android API level 21+适用。
- 3) `TouchableNativeFeedback.Ripple(color, borderless)` - 会创建一个对象，当按钮被按下时产生一个涟漪状的背景，你可以通过`color`参数来指定颜色，如果参数`borderless`是`true`，那么涟漪还会渲染到视图的范围之外。（参见原生的`actionbar buttons`作为该效果的一个例子）。这个背景类型只在Android API level 21+适用也就是Android5.0或以上设备。

```
<TouchableNativeFeedback
  onPress={()=>{
    this.setState({count: this.state.count + 1})
  }}
  background={TouchableNativeFeedback.SelectableBackground()}>
  <View style={{width: 150, height: 100, backgroundColor: 'red'}}>
    <Text style={{margin: 30}}>TouchableNativeFeedback</Text>
  </View>
</TouchableNativeFeedback>
<Text style={styles.text}>{this.state.text}</Text>
```

[下载源码](#)

