

Basic Components

Component	Description	Props
<Button>	Simple button. Don't forget it's title.	title [str] onPress [func] color [str]
<Text>	Displays text.	
<TextInput>	Allows users to input text using keyboard.	onChange [func] value [str] placeholder [str] maxLength [num] multiline [bool]
<Image>	Displays image.	resizeMode [str] source [str]
<ImageBackground>	Displays an image in background (under all its child's elements). Same props of <Image>.	
<View>	Container to create layouts. Think it like a <div> in HTML.	

PS: Some components doesn't need to have a closing tag, you can close it using <Component />. Not all props are listed here; the 'style' prop for example, is common to almost all components.

List Components

Component	Description
<ScrollView>	Makes a list of components scrollable. Renders all components at once (performance may drops on very long lists).
<FlatList>	Makes a list of scrollable components. Render components lazily.

List Components (cont)

<SectionList> Like a FlatList, but you can make sections.

Example displaying lists

```
const dataArray = [
  {id: 1, text: 'First' },
  {id: 2, text: 'Second' },
  {id: 3, text: 'Third' }, ];
<ScrollView> {
  dataArray.map((i) => {
    return ( <View key={i.id}> <Text> {i.text}
  </Text> </View> )
  })
}</ScrollView>
<FlatList
  // SectionList uses sections = { dataArray }
  data = { dataArray }
  keyExtractor = { (item, index) => index.toString() }
  // SectionList uses renderItem too
  renderItem = { ({item}) => <Text> {item.text} </Text> }
  refreshingControl = {
    <RefreshControl refreshing={isRefreshing}
    onRefresh={handleFunc} />
  }
/>
```

useState Hook

```
const myApp = () => {
  const [myName, setMyName] = useState( "foo " );
  setMyName (myName + " bar"); // myName
  changed to "foo bar"
  // now you can use it like <Text> {myName} </Text>
  // use whatever you like (string, number,
  object, boolean, etc)
  const [page, setPage] = useState({id: 1,
  title: " Home" });
}
```



useEffect Hook

```
// Second param is dependencies array. Leave it
empty to make
// it call only once. If you omit this parameter,
the hook will run
// every re-render of the component.
useEffect(() => {
  // Do stuff
  console.log();
  return () => {
    // Runs whenever this environment is
    deleted.
    console.log();
  }
}, []); // [myName] will make it run every time
myName changes.
```

Touchable Components (custom buttons)

Component	Description
<code><TouchableWithoutFeedback></code>	Makes its children touchable (pressable), but without any visual feedback.
<code><TouchableHighlight></code>	Same as above, but decreases opacity to show an underlay color. Props: <i>activeOpacity [num]</i> , <i>underlayColor [str]</i> .
<code><TouchableOpacity></code>	Opacity decreases and can be controlled by an <code><Animated.View></code> .
<code><Pressable></code>	Makes its child pressable, and handle many stages of press interactions.

Use `onPress [func]` in any of these to handle interactions. `<Pressable>` can handle too: `onHoverIn`, `onHoverOut`, `onLongPress`, `onPressIn` and `onPressOut`.

Basic Component Structure

```
// Importing basic components
import React, {useState, useEffect} from 'react';
import {View, Text, StyleSheet} from 'react-native';
import MyCustomComponent from './MyFolder/CustomComponentFile';
const myApp = () => {
```

Basic Component Structure (cont)

```
> return (
  <View style={styles.myViewStyle}>
    <Text style={styles.myText}>Hello World</Text>
  </View>
);
};
const styles = StyleSheet.create({
  body: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  myText: {
    color: '#aaaaff',
    fontSize: 20,
  },
});
export default myApp;
```

Basic StyleSheet Syntax

```
import { StyleSheet } from 'react-native'; //
don't forget to include
const styles = StyleSheet.create({
  myText: {
    color: '#00f',
    fontFamily: 'Helvetica',
    fontStyle: 'italic',
    fontWeight: 'bold',
    letterSpacing: 4,
    textAlign: ['center', 'auto', 'left',
'right', 'justify' (only iOS)],
    textTransform: ['uppercase', 'lower case',
'capitalize'],
    // see too: textShadow[COLOR|OFFS| -
Radius]
    // see textDecoration[LINE|COLOR|STYLE]
  },
  myView: {
```



Basic StyleSheet Syntax (cont)

```
> backgroundColor: '#0f0',
flex: 1,
flexDirection: 'column', // row, [row|column]-reverse
margin: 10,
padding: 10,
width: 50, // (see [min|max]Width)
height: "90%", // (see [min|max]Height)
alignItems: ['center', 'flex-[start|end]', 'stretch', 'baseline'],
// flex-[start|end], space-[between|around|evenly]
justifyContent: 'center',
overflow: ['hidden', 'visible', 'scroll'],
position: ['absolute', 'relative'],
top: 150, // used with position: 'absolute'
right: "100%", // used with position: 'absolute'
},
myImage: {
opacity: 90,
resizeMode: ['stretch', 'cover', 'contain', 'repeat', 'center'],
// see too: border[Color|Radius],
}
});
```

PS: not all properties are listed here, only the most important. Take only one value from styles with arrays.



By **Elieder**
cheatography.com/elieder/

Not published yet.
Last updated 30th June, 2023.
Page 3 of 3.

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
<http://crosswordcheats.com>