Cheatography

React Native Basics Cheat Sheet

by Elieder via cheatography.com/131560/cs/39345/

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]
<ImageBackgr-ound>	Displays an image in background (under all its childs elements). Same props of <Image>.	
<View>	Container to create layouts. Think it like a in HTML.	

PS: Some components doesn't need to have a closing tag, you can close it using `<Co mponent />`. 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' }, ];
<Sc rol lVi ew> {
  dat aAr ray.ma p((i) => {
    return ( <View key={i.id }> <Te xt> {i.t ex t}
< /Te xt> </V iew> )
 })
} </S cro llV iew>
<Fl atList
 // Sectio nList uses sections = { dataArray }
 data = { dataArray }
  key Ext ractor = { (item, index) => index.t oS -
tring() }
 // Sectio nList uses render Sec tio nHeader too
  ren derItem = { ({item}) => <Te xt> {it em.t ex -
t}< /Te xt> }
  ref res hCo ntrol = {
    <Re fre shC ontrol refres hin g={ isR efr -
eshing} onRefr esh ={h and leFunc} />
 }
/>
```

useState Hook

```
const myApp = () => {
        const [myName, setMyName] = useSta te( " -
foo ");
        set MyN ame (myName + " bar"); // myName
changed to "foo bar"
       // now you can use it like <Te xt> {my Nam -
e}< /Te xt>
       // use whatever you like (string, number,
object, boolean, etc)
        const [page, setPage] = useSta te({id: 1,
title: " Hom e"});
}
```

useEffect Hook

```
// Second param is dependencies array. Leave it empty to make
// it call only once. If you ommit this parameter, the hook will run
// every re-render of the component.
useEff ect(() => {
       // Do stuff
        con n.l ogin();
        return () => {
               // Runs whenever this enviroment is deleted.
                con n.l ogo ut();
       }
}, []); // [myName] will make it run every time myName changes.
```

Touchable Components (custom buttons)

Component	Description
<TouchableWithoutFeedback>	Makes its childs touchable (pressable), but without any visual feedback.
<TouchableHighlight>	Same as above, but decreases opacity to show an underlay color. Props: *activeOpacity [num]*, *underlayColor [str]*.
<TouchableOpacity>	Opacity decreases and can be controlled by an *<Animated.View>*.
<Pressable>	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 - na tive';
import MyCust omC omp onent from './MyF old er/ - Cus tom Com pon ent File';
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 = StyleS hee t.c reate({
  myText: {
     color: '#00f',
     fon tFa mily: 'Helve tica',
     fon tStyle: 'italic',
     fon tWe ight: 'bold',
     let ter Spa cing: 4,
     tex tAlign: ['center', 'auto', 'left', 'right', 'justi fy' (only iOS)],
     tex tTr ans form: ['uppe rcase', 'lower case', 'capit ali ze'],
    // see too: textSh ado w[C olo r|O ffs et| - Radius]
    // see textDe cor ati on[ Lin e|C olo r|S tyle]
  },
  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.