REACT NATIVE BASICS CHEAT SHEET

Learn the basics of creating an app using React Native.

by EliederSousa

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 = () => {
    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;
```

List Components

Component	Description
<ScrollView>	Makes a list of components scroll-able. Renders all components at once (performance may drops on very long lists).
<FlatList>	Makes a list of scrollable components. Render components lazily.
<SectionList>	Like a FlatList, but you can make sections.

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 childs 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.

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 na 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*.

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 ommit this parameter, the hook will run
// every re-render of the component.
useEffect(() => {
    // Do stuff
    conn.login();
    return () => {
        // Runs whenever this enviroment is deleted.
        conn.logout();
    }
}, []); // [myName] will make it run every time myName changes.
```

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 renderSectionHeader too
 renderItem = { ({item}) => <Text>{item.text}</Text> }
 refreshControl = {
  <RefreshControl refreshing={isRefreshing} onRefresh={handleFunc} />
 }
/>
```

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', 'lowercase', 'capitalize'],
  // see too: textShadow[Color|Offset|Radius]
  // see textDecoration[Line|Color|Style]
 },
 myView: {
  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.