***React-Native Notes:***

* ***Components in React Native and their use***

|  |  |  |
| --- | --- | --- |
| ***S.No*** | ***Component Name*** | ***Use*** |
| 1 | View | Used as a “div” element |
| 2 | Text | For adding text |
| 3 | StyleSheet | const styles = StyleSheet.create({  textStyle: { fontSize: 30, },    subHeaderStyle: { fontSize: 10,},  }); |
| 4 | FlatList:- for looping over elements  props:  horizontal, showsHorizontalScrollIndicator, keyExtractor, data, renderItem | <FlatList  horizontal  showsHorizontalScrollIndicator={false}  keyExtractor={(friend) => friend.name}  data={friends}  renderItem={({item, index}) => { return <Text style={styled.textStyle}>any text </Text>}}  /> |
| 5 | Button | <Button  onPress={() => console.log(‘do something’)}  title=”Click it”  /> |
| 6 | TouchableOpacity | <TouchableOpacity  onPress={() => console.log(‘hello world’)}>  <Text>Click it</Text>  </TouchableOpacity> |
| 7 | Image | <Image          source={require(“../assets/image.png”)}          style={{ width: "100px", height: "100px" }}        /> |
| 8 | TextInput | <TextInput  style={styles.input} autoCapitalize=”none”  autoCorrect={false}  ­­ value={name}  onChangeText={newValue => setName(newValue)} /> |
| 9 | …StyleSheet.absoluteFillObject  (the above property is a shorthand for the position properties on the right) | position: absolute;  top: 0; bottom: 0; left: 0; right: 0; |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**React Navigation Notes:**

* React Native doesn't have a built-in API for navigation like a web browser does. React Navigation provides this for you, along with the iOS and Android gestures and animations to transition between screens.
* Stack.Navigator is a component that takes route configuration as its children with additional props for configuration and renders our content.
* Each Stack.Screen component takes a name prop which refers to the name of the route and component prop which specifies the component to render for the route. These are the 2 required props.
* To specify what the initial route in a stack is, provide an initialRouteName as the prop for the navigator.
* To specify screen-specific options, we can pass an options prop to Stack.Screen, and for common options, we can pass screenOptions to Stack.Navigator

Sometimes we might want to pass additional props to a screen. We can do that with 2 approaches:

1. Use [React context](https://reactjs.org/docs/context.html) and wrap the navigator with a context provider to pass data to the screens (recommended).
2. Use a render callback for the screen instead of specifying a component prop:

<Stack.Screen name="Home">  
 {(props) => <HomeScreen {...props} extraData={someData} />}  
</Stack.Screen>

Note: By default, React Navigation applies optimizations to screen components to prevent unnecessary renders. Using a render callback removes those optimizations. So if you use a render callback, you'll need to ensure that you use [React.memo](https://reactjs.org/docs/react-api.html" \l "reactmemo" \t "_blank) or [React.PureComponent](https://reactjs.org/docs/react-api.html" \l "reactpurecomponent" \t "_blank) for your screen components to avoid performance issues.

* navigation.navigate('RouteName') pushes a new route to the native stack navigator if it's not already in the stack, otherwise it jumps to that screen.
* We can call navigation.push('RouteName') as many times as we like and it will continue pushing routes.
* The header bar will automatically show a back button, but you can programmatically go back by calling navigation.goBack(). On Android, the hardware back button just works as expected.
* You can go back to an existing screen in the stack with navigation.navigate('RouteName'), and you can go back to the first screen in the stack with navigation.popToTop().
* The navigation prop is available to all screen components (components defined as screens in route configuration and rendered by React Navigation as a route).