

CS 696 Multi-platform Mobile App Development  
Fall Semester, 2020  
Doc 18 Navigation  
Oct 29, 2020

Copyright ©, All rights reserved. 2020 SDSU & Roger Whitney,  
5500 Campanile Drive, San Diego, CA 92182-7700 USA.  
OpenContent (<http://www.opencontent.org/opl.shtml>) license  
defines the copyright on this document.

# Navigation

React Native does not have its own system

React Navigation

<https://reactnavigation.org>

Third party library

Recommended by React Native Team

Supports

Stack navigation

Tab navigation

Drawer navigation

# NavigationContainer

Manages app state and linking top-level navigator to the app environment.

Takes care of platform specific integration

Provides

- Deep link integration with the linking prop.

- Notify state changes for screen tracking, state persistence etc.

- Handle system back button on Android

```
import { NavigationContainer } from '@react-navigation/native';  
import { createStackNavigator } from '@react-navigation/stack';
```

```
const Stack = createStackNavigator();
```

```
export default function App() {
```

```
  return (
```

```
    <NavigationContainer>
```

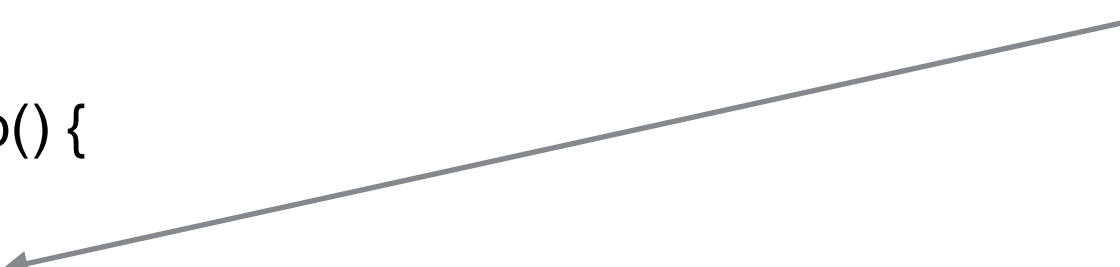
```
      <Stack.Navigator>{/* ... */}</Stack.Navigator>
```

```
    </NavigationContainer>
```

```
  );
```

```
3 }
```

Needs to be  
at top level



# createStackNavigator - Stack.Navigator

Provides a way transition between screens via a stack

```
import { createStackNavigator } from '@react-navigation/stack';

const Stack = createStackNavigator();

function MyStack() {
  return (
    <Stack.Navigator>
      <Stack.Screen name="Home" component={Home} />
      <Stack.Screen name="Notifications" component={Notifications} />
      <Stack.Screen name="Profile" component={Profile} />
      <Stack.Screen name="Settings" component={Settings} />
    </Stack.Navigator>
  );
}
```

# createStackNavigator - Stack.Navigator

A prop

`initialRouteName`

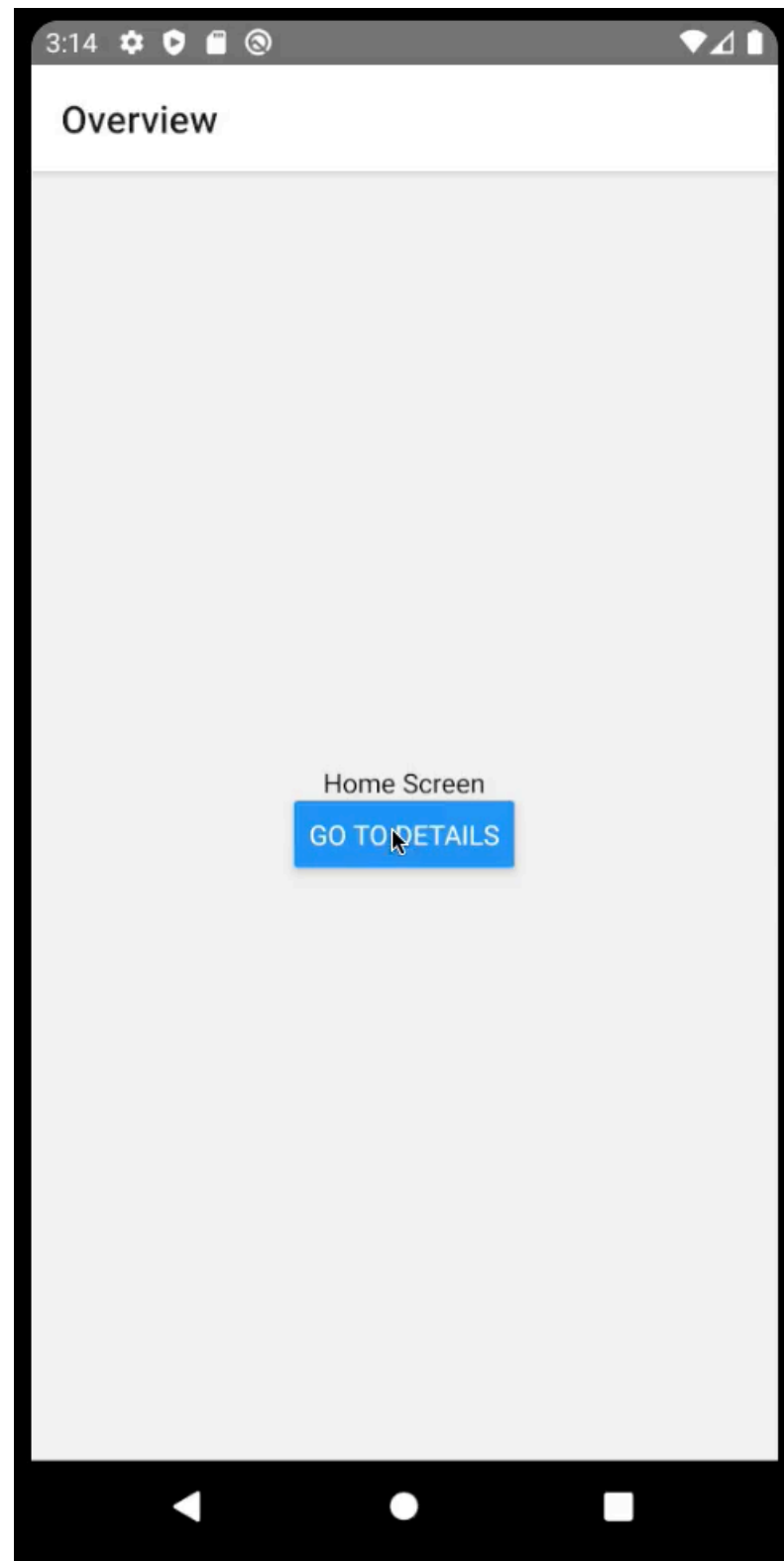
The name of the route to render on first load of the navigator.

Option

`title`

String that can be used as a fallback for headerTitle

# Basic Stack Navigation Example



# Installing

Al pro 15->react-native init NavLecture

Al pro 16->cd NavLecture/

Al pro 17->npm install @react-navigation/native

Al pro 18->npm install react-native-reanimated react-native-gesture-handler react-native-screens react-native-safe-area-context @react-native-community/masked-view

Al pro 19->npm install @react-navigation/stack

# Imports & Style

```
import React from 'react';  
import {Button, Text, View, StyleSheet} from 'react-native';  
import {NavigationContainer} from '@react-navigation/native';  
import {createStackNavigator} from '@react-navigation/stack';
```

```
const styles = StyleSheet.create({  
  container: {flex: 1, alignItems: 'center', justifyContent: 'center'},  
});
```



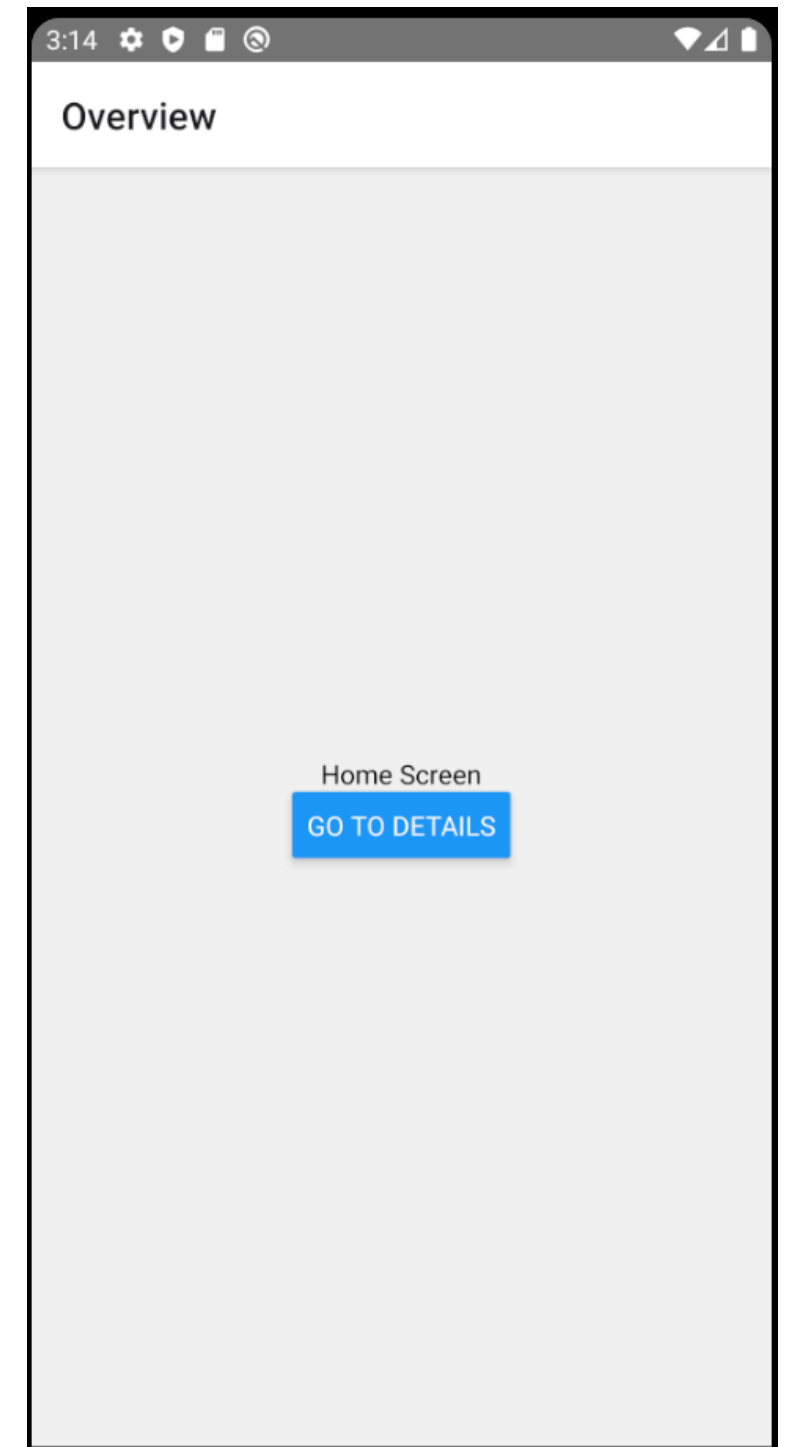
# Navigation Setup

```
const Stack = createStackNavigator();

const App: () => React$Node = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen
          name="Home"
          component={HomeScreen}
          options={{title: 'Overview'}}
        />
        <Stack.Screen name="Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
};
```

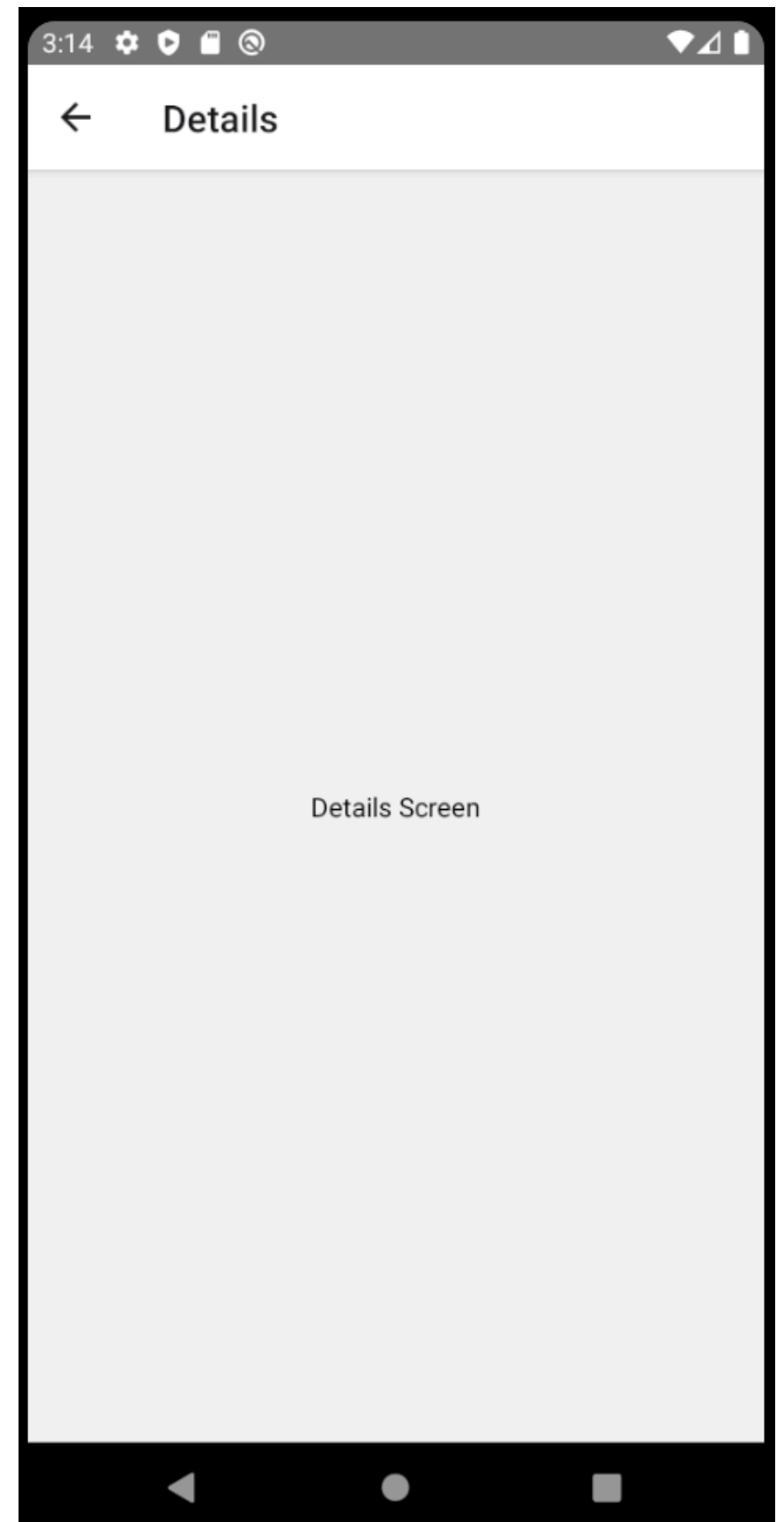
# HomeScreen

```
function HomeScreen({navigation}) {  
  return (  
    <View style={styles.container}>  
      <Text>Home Screen</Text>  
      <Button  
        title="Go to Details"  
        onPress={() => navigation.navigate('Details')}  
      />  
    </View>  
  );  
}
```



# DetailsScreen

```
function DetailsScreen() {  
  return (  
    <View style={styles.container}>  
      <Text>Details Screen</Text>  
    </View>  
  );  
}
```



# navigation prop

Only passed to screen components

```
function HomeScreen({navigation}) {  
  return (  
    <View style={styles.container}>
```

screen  
components

```
    <NavigationContainer>  
      <Stack.Navigator initialRouteName="Home">  
        <Stack.Screen name="Home" component={HomeScreen}/>  
        <Stack.Screen name="Details" component={DetailsScreen} />  
      </Stack.Navigator>  
    </NavigationContainer>  
  );  
};
```

# Navigation & Buttons

```
<Button  
  title="Go to Details"  
  onPress={() => navigation.navigate('Details')}  
>
```

Buttons need access to navigation prop

But only send to screen components

Works if we nest button in screen component

With complex layouts likely break screen component into separate components  
So button may not be inside screen component

# useNavigation Hook

Gives access to navigation prop inside of functions

```
import * as React from 'react';  
import { Button } from 'react-native';  
import { useNavigation } from '@react-navigation/native';
```

```
function MyBackButton() {  
  const navigation = useNavigation();  
  
  return (  
    <Button  
      title="Back"  
      onPress={() => {  
        navigation.goBack();  
      }}  
    />  
  );  
}
```

# What about Class Based Component

Hooks are only for functions - wrap the class in a function

```
function ButtonWrapper() {  
  const navigation = useNavigation();  
  
  return <MyBackButton navigation={navigation} />;  
}  
  
class MyBackButton extends React.Component {  
  render() {  
    const {navigation} = this.props;  
    return (  
      <Button title="Back" onPress={() => { navigation.goBack();  
        }}  
    />  
    );  
  }  
}
```

# Stack Navigation Methods

navigate

- Go to another screen

- If there do nothing

- If screen is in stack go back that screen

- If screen is not in stack, push it on the stack

goBack

- Close active screen and move back in the stack

push

- Push a new route onto the stack

pop

- Go back in the stack

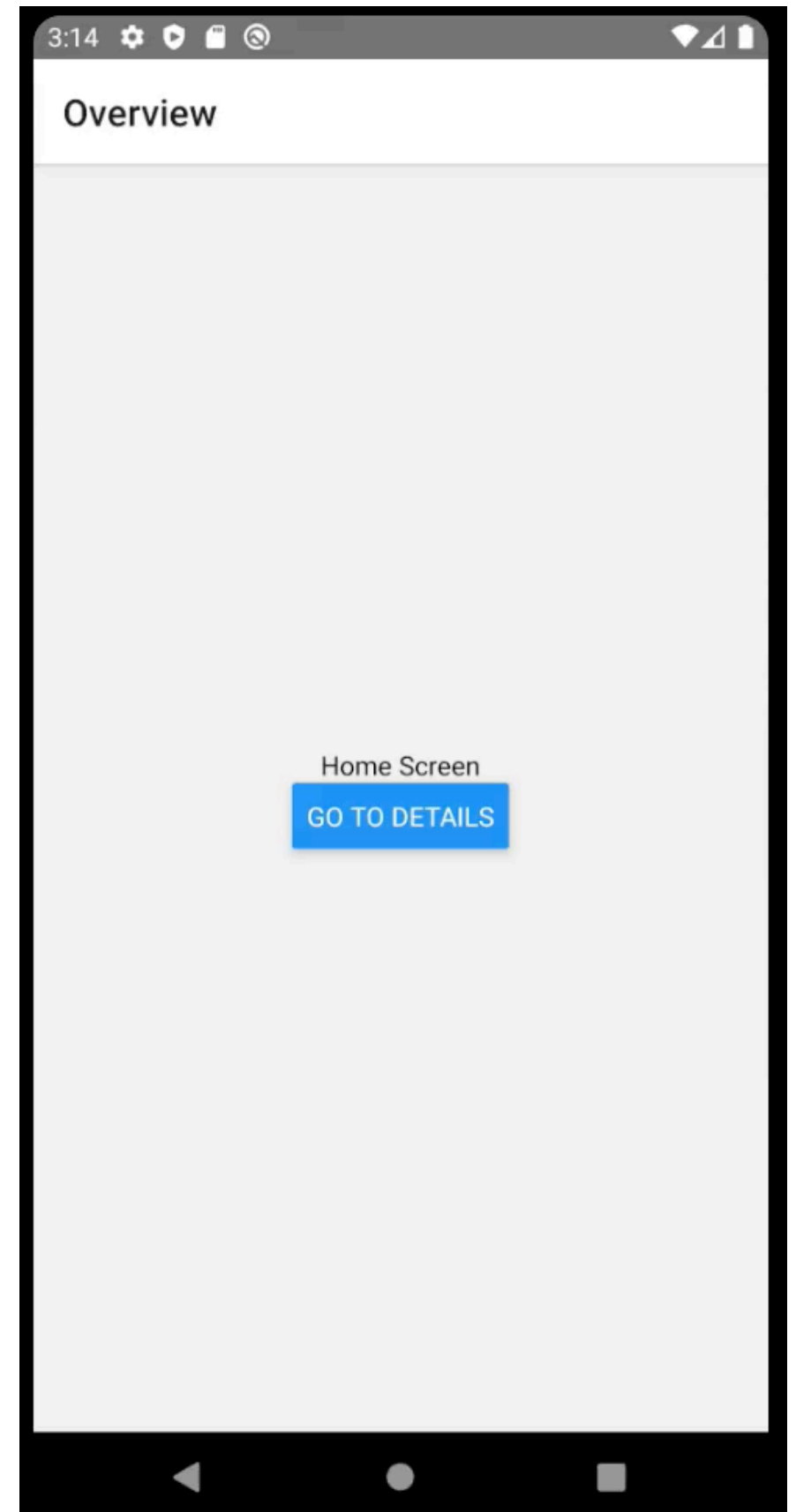
popToTop

- Go to the top of the stack



# Example

```
function DetailsScreen({navigation}) {  
  return (  
    <View style={styles.container}>  
      <Text>Details Screen</Text>  
      <Button title="navigation.navigate('Details')"  
        onPress={() => navigation.navigate('Details')}  
      />  
      <Button title="navigation.push('Details')"  
        onPress={() => navigation.push('Details')}  
      />  
      <Button title="navigation.navigate('Home')"  
        onPress={() => navigation.navigate('Home')}  
      />  
      <Button title="navigation.goBack()"  
        onPress={() => navigation.goBack()} />  
      <Button title="navigation.popToTop()"  
        onPress={() => navigation.popToTop()}  
      />  
    </View>  
  );  
}
```



# Passing Data to Screen

We can pass data to a new screen using props

```
<Button  
  title="Go to Details"  
  onPress={() => navigation.navigate('Details', {itemId: 42, count: 2})}
```

We can set initial values for the props

```
<Stack.Screen  
  name="Details"  
  component={DetailsScreen}  
  initialParams={{ count: -1 }}  
>
```

# Example - Routes

```
const Stack = createStackNavigator();
```

```
const App: () => React$Node = () => {
```

```
  return (
```

```
    <NavigationContainer>
```

```
      <Stack.Navigator initialRouteName="Home">
```

```
        <Stack.Screen
```

```
          name="Home"
```

```
          component={HomeScreen}
```

```
          options={{title: 'Overview'}}
```

```
        />
```

```
        <Stack.Screen name="Details" component={DetailsScreen} initialParams={{ count: -1 }}>
```

```
      />
```

```
    </Stack.Navigator>
```

```
  </NavigationContainer>
```

```
);
```

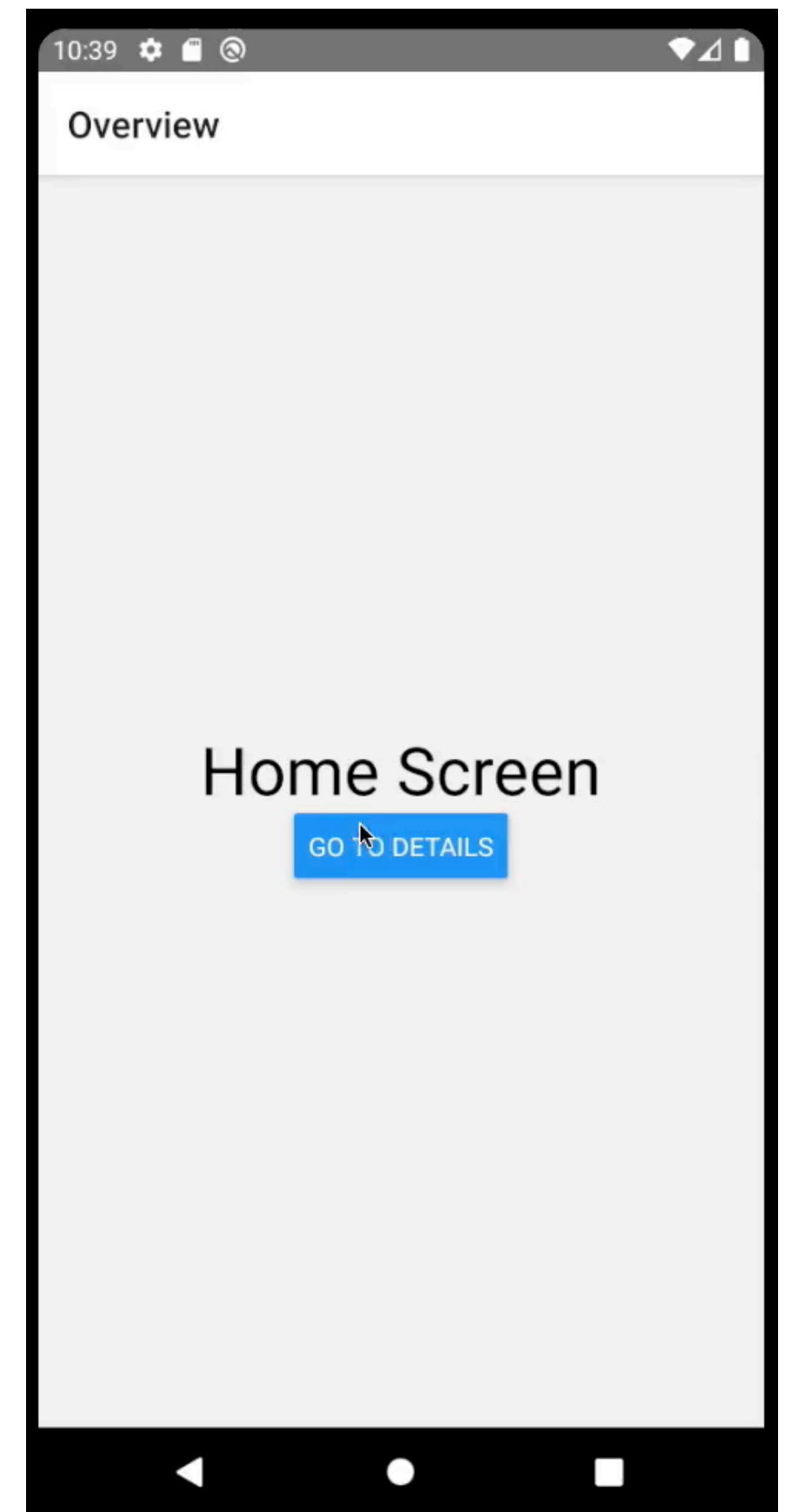
```
};
```

# Example - HomeScreen

```
function HomeScreen({navigation}) {  
  return (  
    <View style={styles.container}>  
      <Text style={styles.largeText}>Home Screen</Text>  
      <Button  
        title="Go to Details"  
        onPress={() => navigation.navigate('Details', {itemId: 42, count: 2})}  
      />  
    </View>  
  );  
}
```

# DetailsScreen

```
function DetailsScreen({route, navigation}) {  
  const {itemId, count} = route.params;  
  return (  
    <View style={styles.container}>  
      <Text>Details Screen</Text>  
      <Text style={styles.largeText}>  
        itemId: {JSON.stringify(itemId)}</Text>  
      <Text style={styles.largeText}>  
        count: {JSON.stringify(count)}</Text>  
      <Button  
        title="navigation.push('Details')"  
        onPress={() =>  
          navigation.push('Details', {  
            itemId: itemId + 2,  
          })  
        }  
      />  
    </View>  
  );  
}
```



# Changing Value of Params

setParams

Like setState

Changes params and updated the component

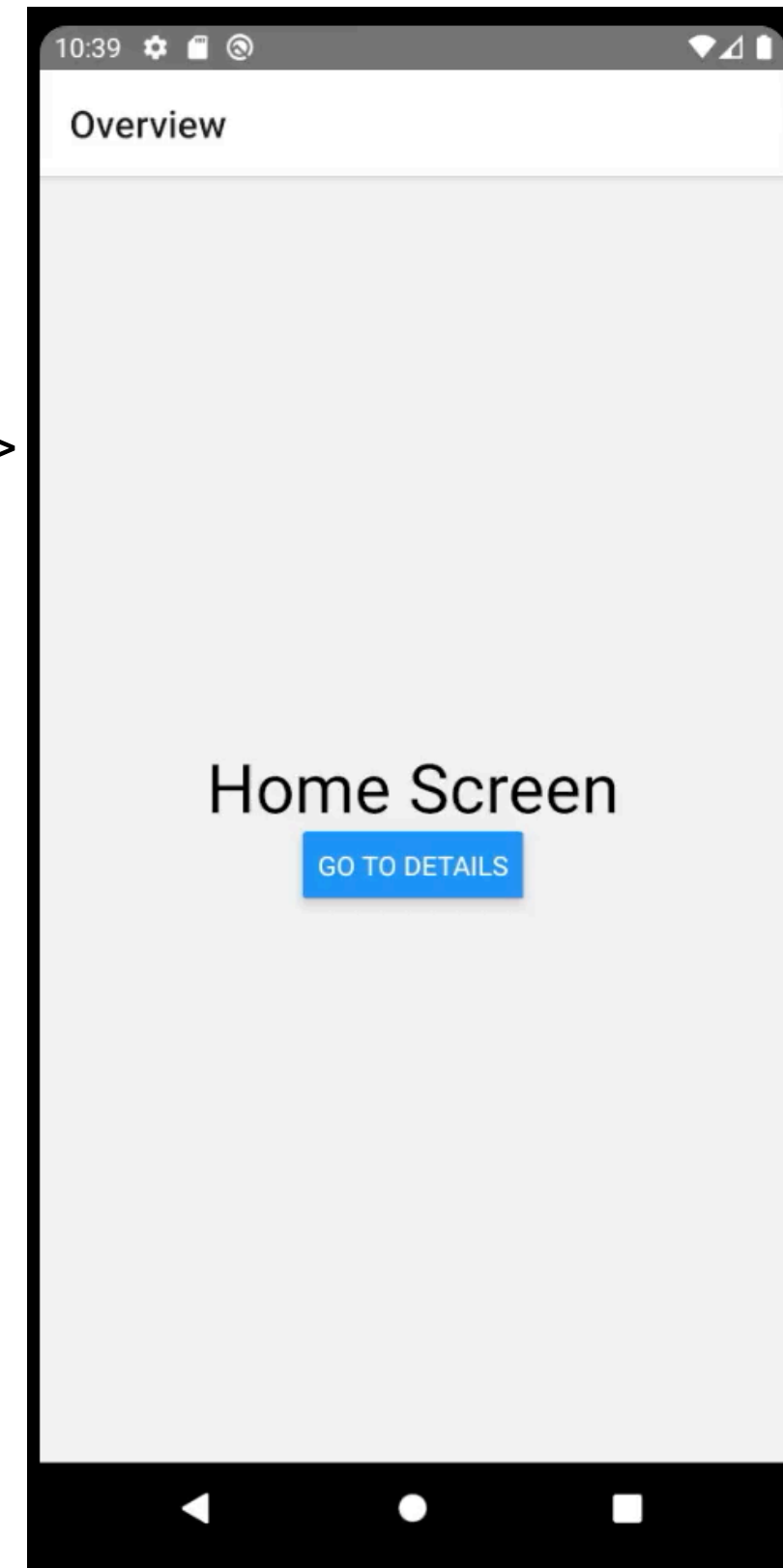
Odd syntax

```

function DetailsScreen({route, navigation, navigation: {setParams}}) {
  const {itemId, count} = route.params;

  return (
    <View style={styles.container}>
      <Text>Details Screen</Text>
      <Text style={styles.largeText}>itemId: {JSON.stringify(itemId)}</Text>
      <Text style={styles.largeText}>count: {JSON.stringify(count)}</Text>
      <Button
        title="setParams"
        onPress={() =>
          setParams({
            itemId: (route.params.itemId = itemId + 10),
          })
        }
      />
      <Button
        title="navigation.push('Details')"
        onPress={() => navigation.push('Details', { itemId: itemId + 2, }) }
      />
    </View>
  );
}

```



# Passing Data back to Caller

Use `navigation.navigate` to go back to caller

Pass props

What to do if there are multiple ways to reach current screen?

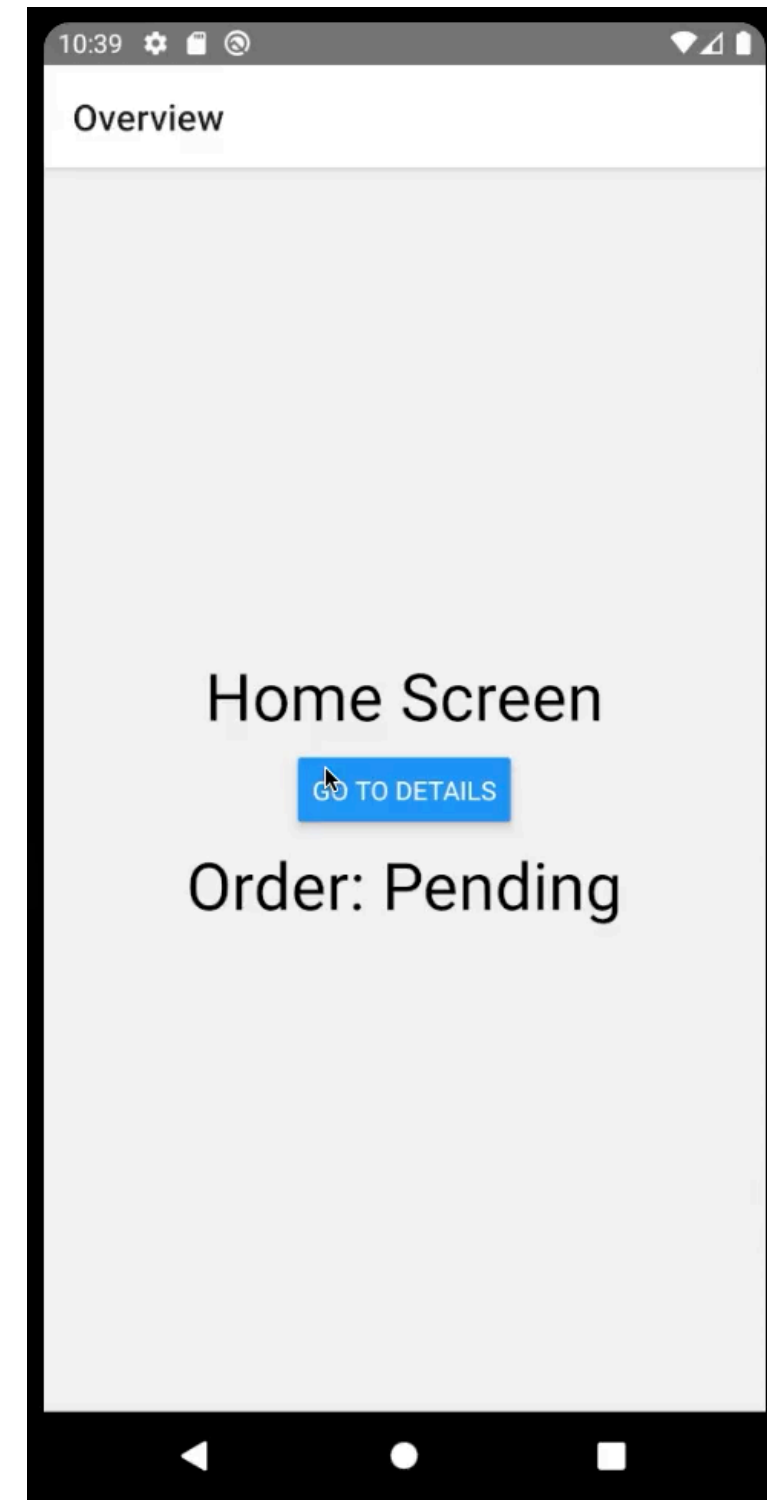


# HomeScreen

```
function HomeScreen({navigation, route}) {  
  return (  
    <View style={styles.container}>  
      <Text style={styles.largeText}>Home Screen</Text>  
      <Button  
        title="Go to Details"  
        onPress={() => navigation.navigate('Details', {itemId: 42, count: 2})}  
      />  
      <Text style={styles.largeText}>  
        Order: {route.params ? route.params?.order:'Pending'}  
      </Text>  
    </View>  
  );  
}
```

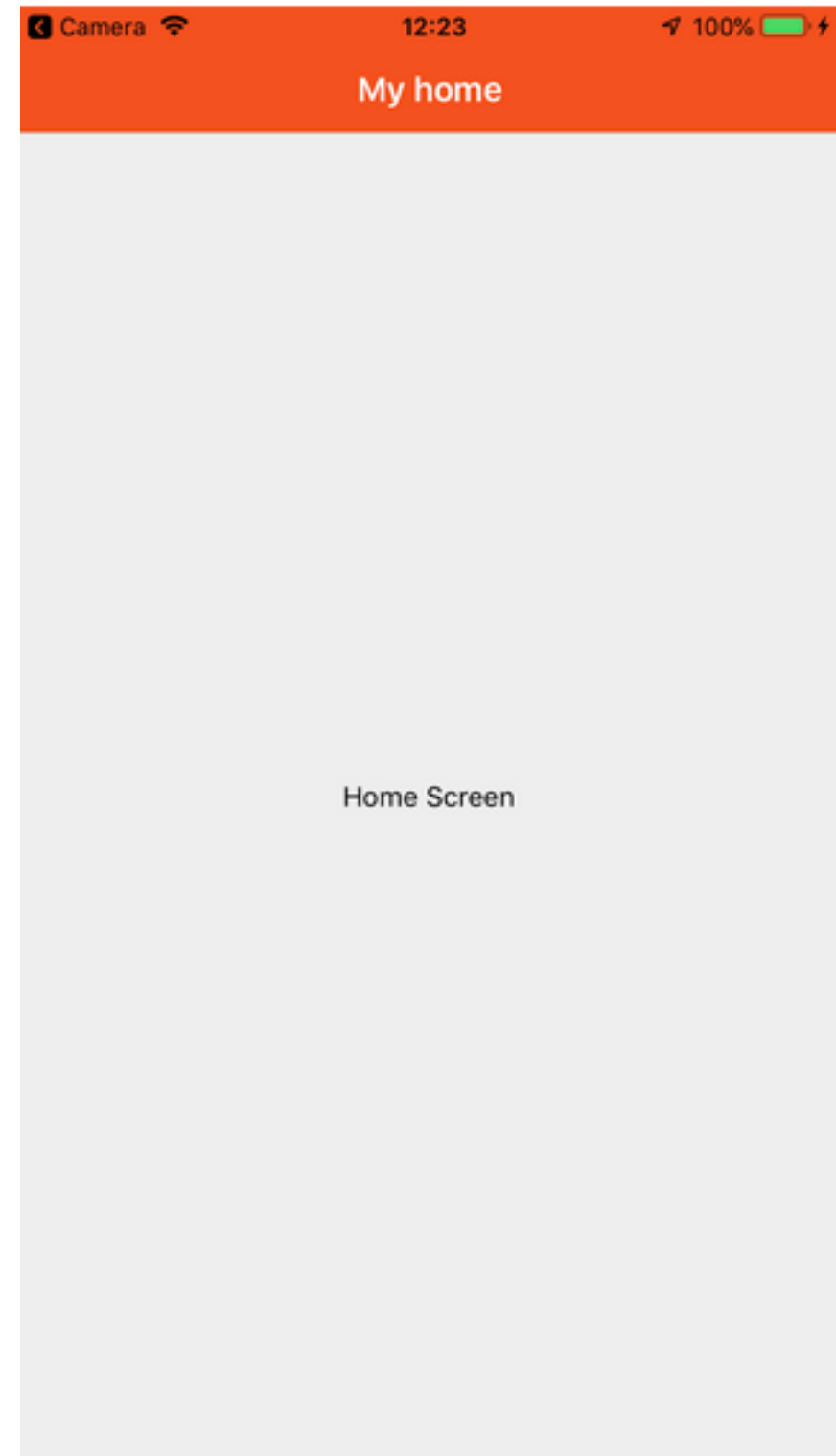
# DetailsScreen

```
function DetailsScreen({route, navigation}) {  
  const {itemId, count} = route.params;  
  return (  
    <View style={styles.container}>  
      <Text>Details Screen</Text>  
      <Text style={styles.largeText}>itemId: {JSON.stringify(itemId)}</Text>  
      <Text style={styles.largeText}>count: {JSON.stringify(count)}</Text>  
      <Button  
        title="Complete Order"  
        onPress={() => navigation.navigate('Home', {order: 'Done',})} />  
      <Button  
        title="Cancel Order"  
        onPress={() => navigation.navigate('Home', {order: 'Canceled', }) } />  
    </View>  
  );  
}
```



# Adjusting header styles

```
function StackScreen() {  
  return (  
    <Stack.Navigator>  
      <Stack.Screen  
        name="Home"  
        component={HomeScreen}  
        options={{  
          title: 'My home',  
          headerStyle: {  
            backgroundColor: '#f4511e',  
          },  
          headerTintColor: '#fff',  
          headerTitleStyle: {  
            fontWeight: 'bold',  
          },  
        }}  
      />  
    </Stack.Navigator>  
  );  
}
```



# Adjusting header styles

```
function StackScreen() {  
  return (  
    <Stack.Navigator>  
      <Stack.Screen  
        name="Home"  
        component={HomeScreen}  
        options={{  
          title: 'My home',  
          headerStyle: {  
            backgroundColor: '#f4511e',  
          },  
          headerTintColor: '#fff',  
          headerTitleStyle: {  
            fontWeight: 'bold',  
          },  
        }}  
      />  
    </Stack.Navigator>  
  );  
}
```

headerTintColor

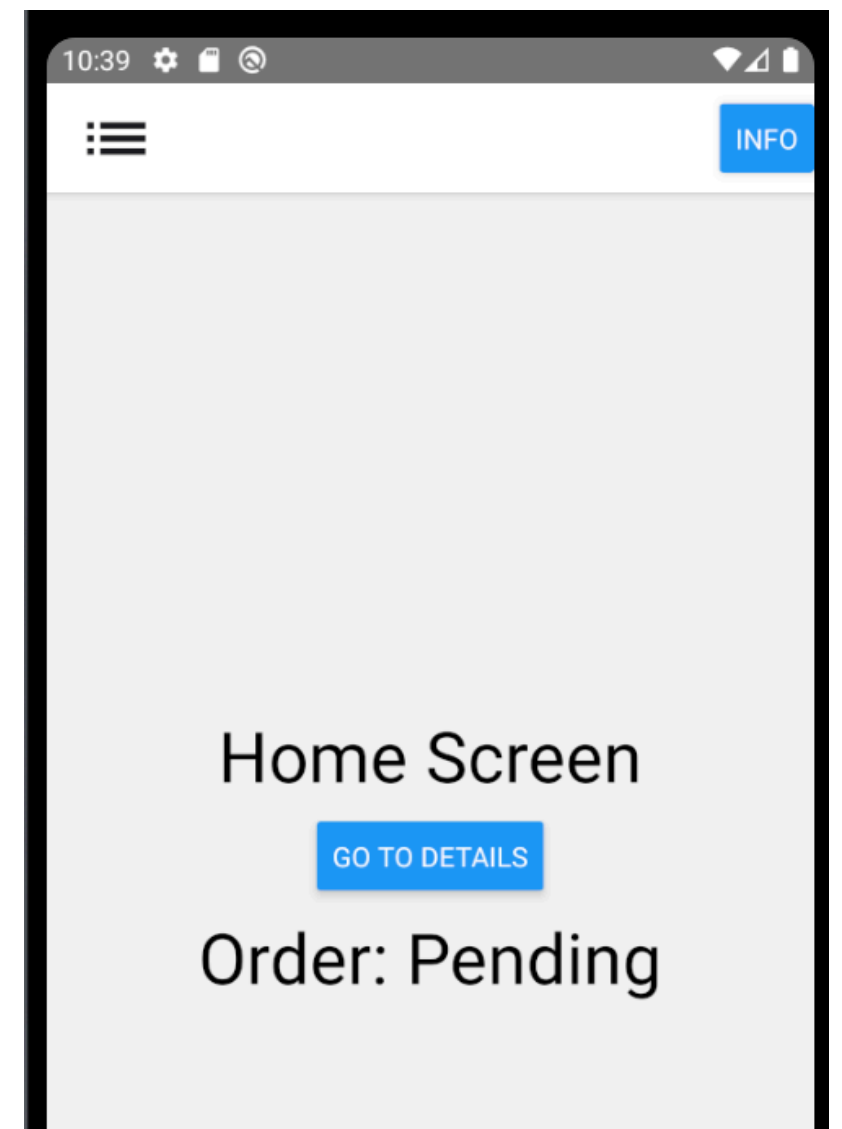
For back button and title

headerStyle

Applied to the View that wraps the header

# Header Buttons & Icons

```
const App: () => React$Node = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Home">
        <Stack.Screen
          name="Home"
          component={HomeScreen}
          options={{
            headerTitle: <Icon name="list" size={40} />,
            headerRight: () => (
              <Button onPress={() => alert('This is a button!')} title="Info" />
            ),
          }}
        />
        <Stack.Screen
          name="Details"
          component={DetailsScreen}
          initialParams={{count: -1}}
        />
      </Stack.Navigator>
    </NavigationContainer>
  )
}
```



# Icons

react-native-vector-icons

Installing

<https://github.com/oblador/react-native-vector-icons#icon-component>

List & usage

<https://reactnativeelements.com/docs/icon/>

Importing an Icon set

```
import Icon from 'react-native-vector-icons/MaterialIcons';
```

# Tabs

createBottomTabNavigator

Need

@react-navigation/native

@react-navigation/bottom-tabs

npm install @react-navigation/bottom-tabs



createMaterialBottomTabNavigator

Need

@react-navigation/native

@react-navigation/material-bottom-tabs

npm install @react-navigation/material-bottom-tabs react-native-paper



createMaterialTopTabNavigator

Need

@react-navigation/native

@react-navigation/material-top-tabs

npm install @react-navigation/material-top-tabs react-native-tab-view

# Tab Navigators

```
import React from 'react';
import { Text, View, } from 'react-native';
import {NavigationContainer} from '@react-navigation/native';
import Icon from 'react-native-vector-icons/MaterialIcons';
import {createMaterialBottomTabNavigator} from '@react-navigation/material-bottom-tabs';
```

```
function HomeScreen() {
  return (
    <View style={{flex: 1, justifyContent: 'center', alignItems: 'center'}}>
      <Text>Home!</Text>
    </View>
  );
}
```

```
function SettingsScreen() {
  return (
    <View style={{flex: 1, justifyContent: 'center', alignItems: 'center'}}>
      <Text>Settings!</Text>
    </View>
  );
}
```

---

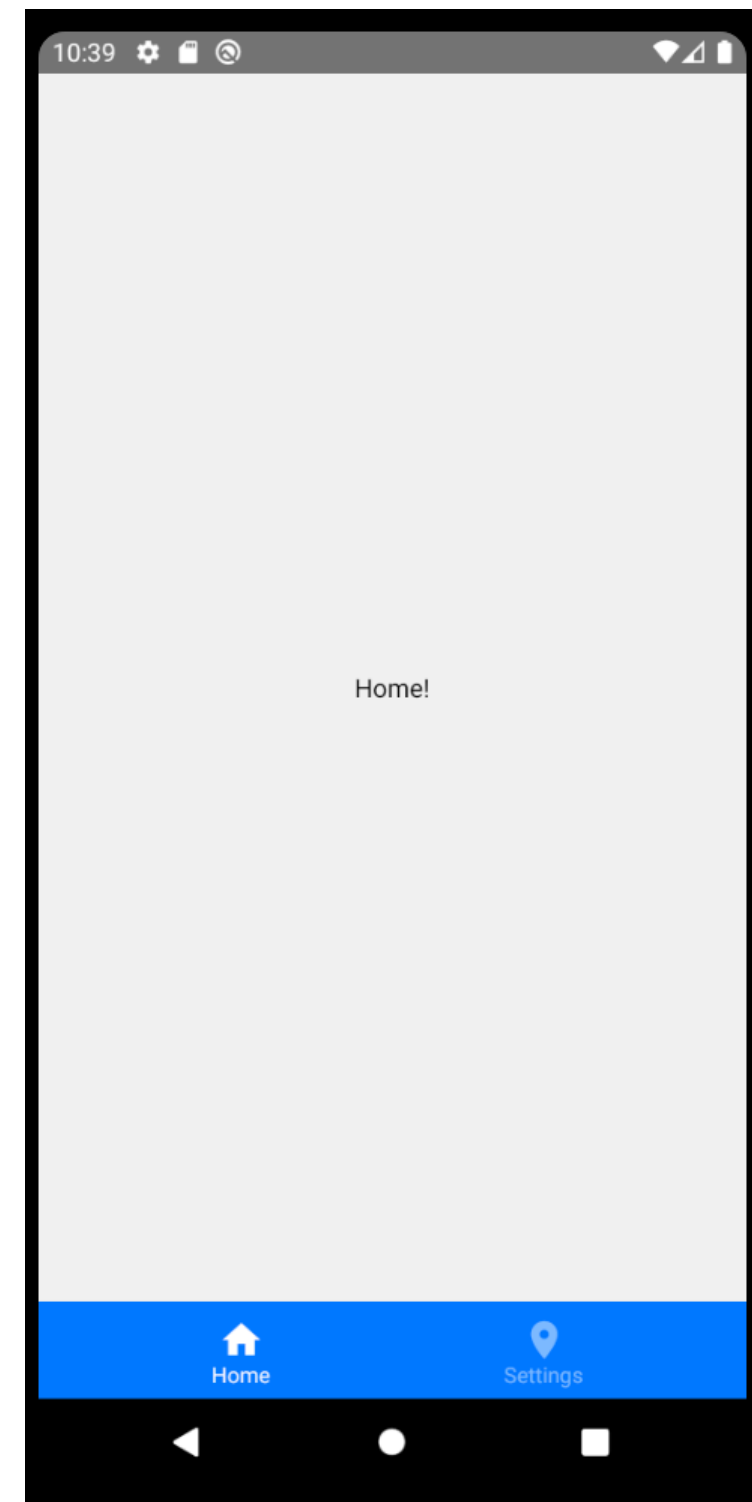


```

const Tab = createMaterialBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator
        screenOptions={({route}) => ({
          tabBarIcon: ({focused, color, size}) => {
            let iconName;
            if (route.name === 'Home') { iconName = 'home';
            } else if (route.name === 'Settings') { iconName = 'room';
            }
            return <Icon name={iconName} size={25} color={color} />;
          },
        )}
        tabBarOptions={{
          activeTintColor: 'tomato',
          inactiveTintColor: 'gray',
        }}
        <Tab.Screen name="Home" component={HomeScreen} />
        <Tab.Screen name="Settings" component={SettingsScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}

```



# createDrawerNavigator

Need

`@react-navigation/native`

`@react-navigation/drawer`

`npm install @react-navigation/drawer`

# Drawer Example

```
import React from 'react';
import {Text, View, useWindowDimensions} from 'react-native';
import {NavigationContainer} from '@react-navigation/native';
import {createDrawerNavigator} from '@react-navigation/drawer';

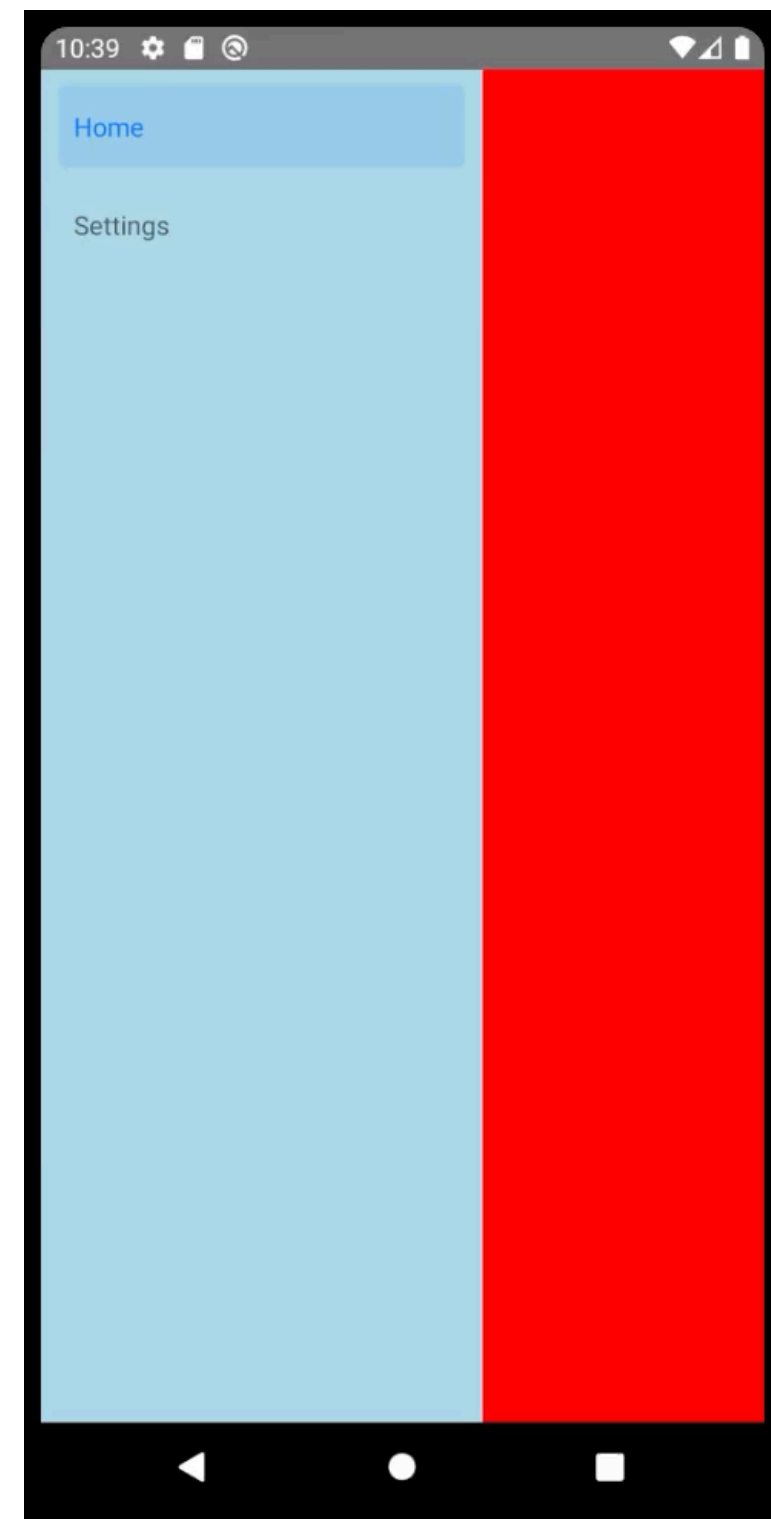
function HomeScreen() {
  return (
    <View style={{flex: 1, justifyContent: 'center', alignItems: 'center'}}>
      <Text>Home!</Text>
    </View>
  );
}

function SettingsScreen() {
  return (
    <View style={{flex: 1, justifyContent: 'center', alignItems: 'center'}}>
      <Text>Settings!</Text>
    </View>
  );
}
```

---

```
const Drawer = createDrawerNavigator();
```

```
export default function App() {  
  const dimensions = useWindowDimensions();  
  const isLargeScreen = dimensions.width >= 768;  
  return (  
    <NavigationContainer>  
      <Drawer.Navigator  
        openByDefault  
        drawerStyle={{  
          backgroundColor: 'lightblue',  
          width: 240,  
        }}  
        drawerType={isLargeScreen ? 'permanent' : 'back'}  
        overlayColor="red">  
        <Drawer.Screen name="Home" component={HomeScreen} />  
        <Drawer.Screen name="Settings" component={SettingsScreen} />  
      </Drawer.Navigator>  
    </NavigationContainer>  
  );  
}
```



# Development Size - Navigation Example

Directory	Size MB
Android	275
iOS	209
node_modules	243
Sum of Above	727
Entire Project	782