Hello React Navigation

Presentor: 高以任

- Basic Navigation Concept
- Pre-requisites
- Basic Usages
- Properties
- StackActions
- Custom Navigator(Another Topic)

Basic Navigation Concept

This native-stack navigator uses the native APIs: UINavigationController on iOS and Fragment on Android so that navigation built with createNativeStackNavigator will behave the same and have the same performance characteristics as apps built natively on top of those APIs.

- Different from Android native Navigation component(released 2018)

Pre-requisites

Dependencies

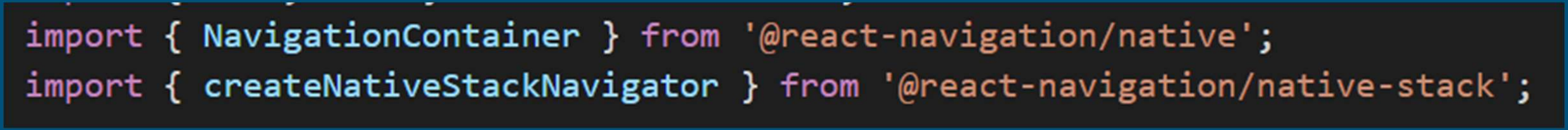
```
npm install react-native-screens react-native-safe-area-context
```

Packages

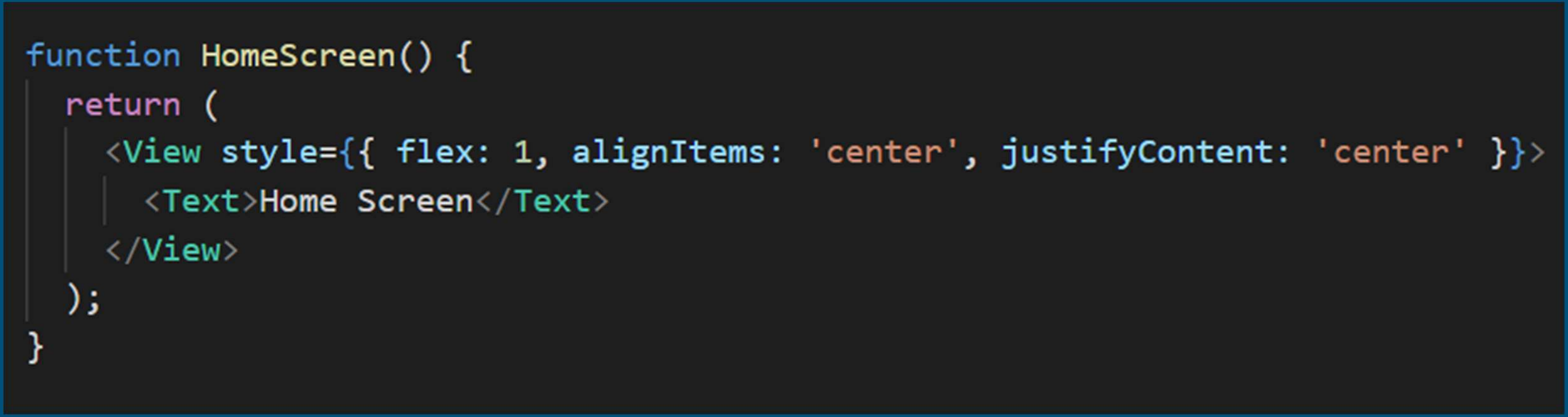
```
npm install @react-navigation/native
```

```
npm install @react-navigation/native-stack
```

Basic Usages

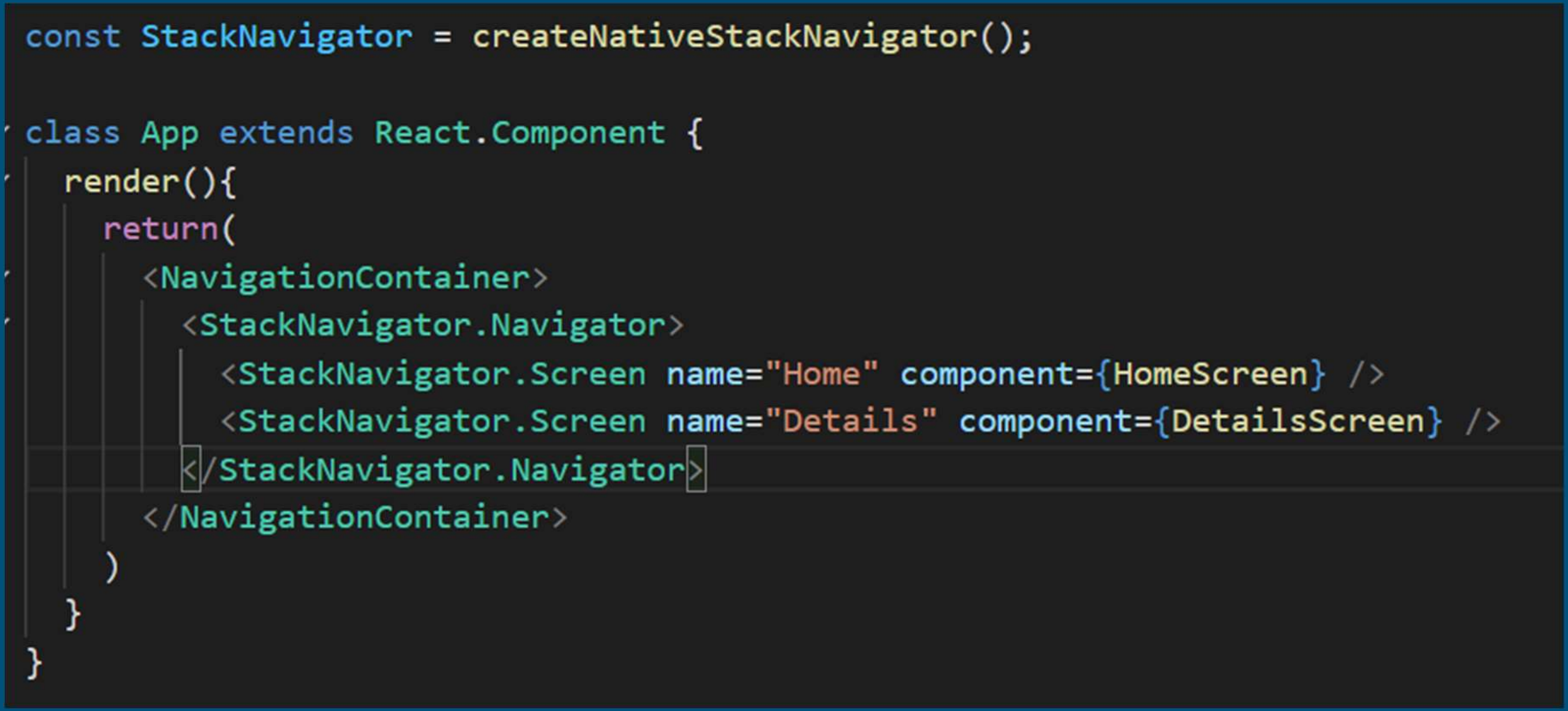


```
import { NavigationContainer } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';
```

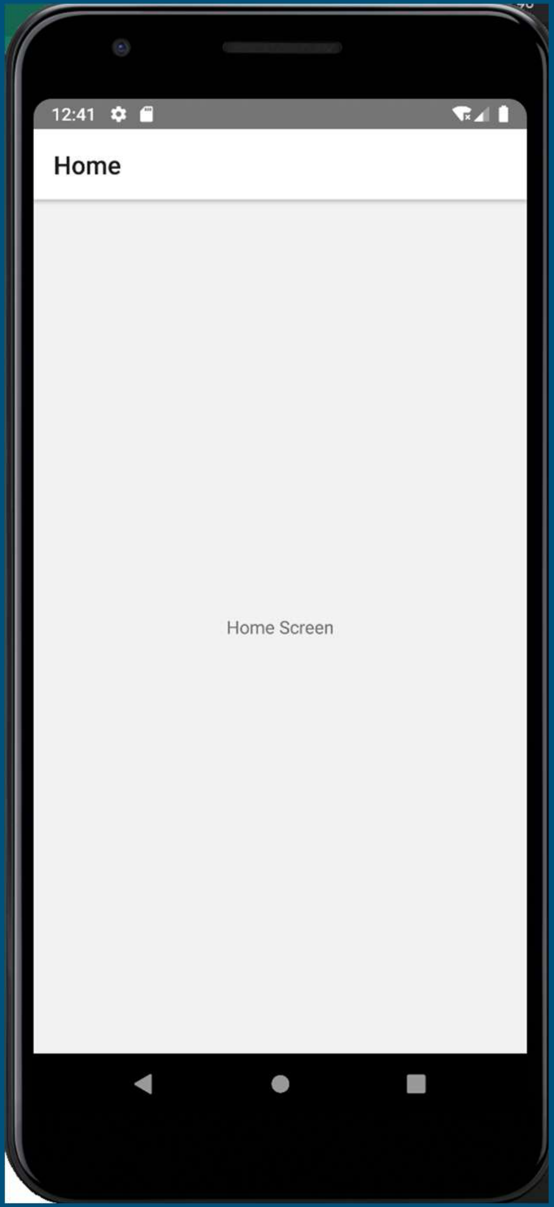


```
function HomeScreen() {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
    </View>
  );
}
```

Basic Usages

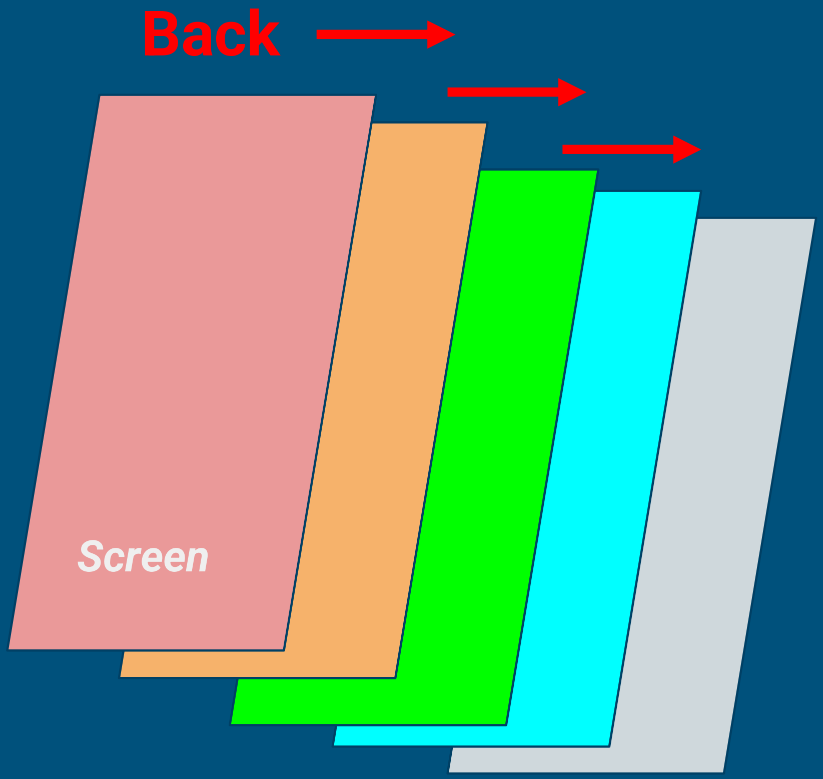


```
const StackNavigator = createNativeStackNavigator();

class App extends React.Component {
  render(){
    return(
      <NavigationContainer>
        <StackNavigator.Navigator>
          <StackNavigator.Screen name="Home" component={HomeScreen} />
          <StackNavigator.Screen name="Details" component={DetailsScreen} />
        </StackNavigator.Navigator>
      </NavigationContainer>
    )
  }
}
```



Navigating between Screens

Route



Properties – Navigation Props Reference

Each screen component in your app is provided with the navigation prop automatically. The prop contains various convenience functions that dispatch navigation actions. It looks like this:

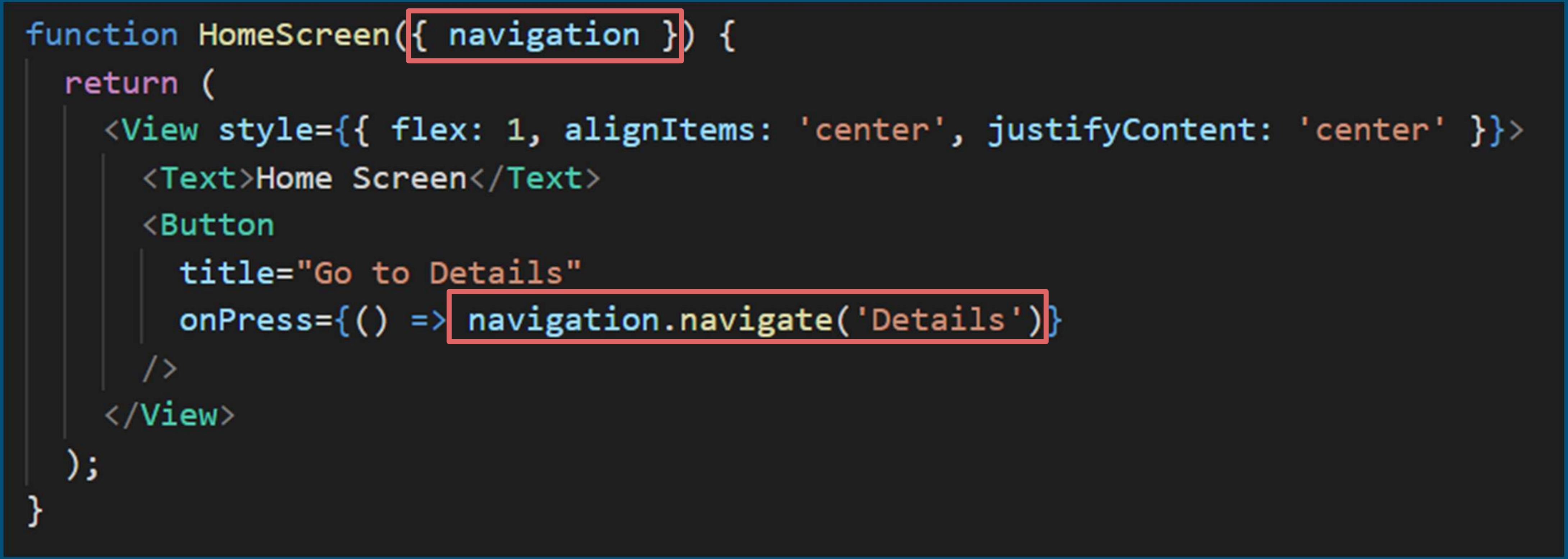
- navigate - go to another screen, figures out the action it needs to take to do it
- reset - wipe the navigator state and replace it with a new route
- goBack - close active screen and move back in the stack
- setParams - make changes to route's params
- dispatch - send an action object to update the navigation state
- setOptions - update the screen's options
- isFocused - check whether the screen is focused
- addListener - subscribe to updates to events from the navigators

Properties – Route Props Reference

Each screen component in your app is provided with the route prop automatically. The prop contains various information regarding current route (place in navigation hierarchy component lives).

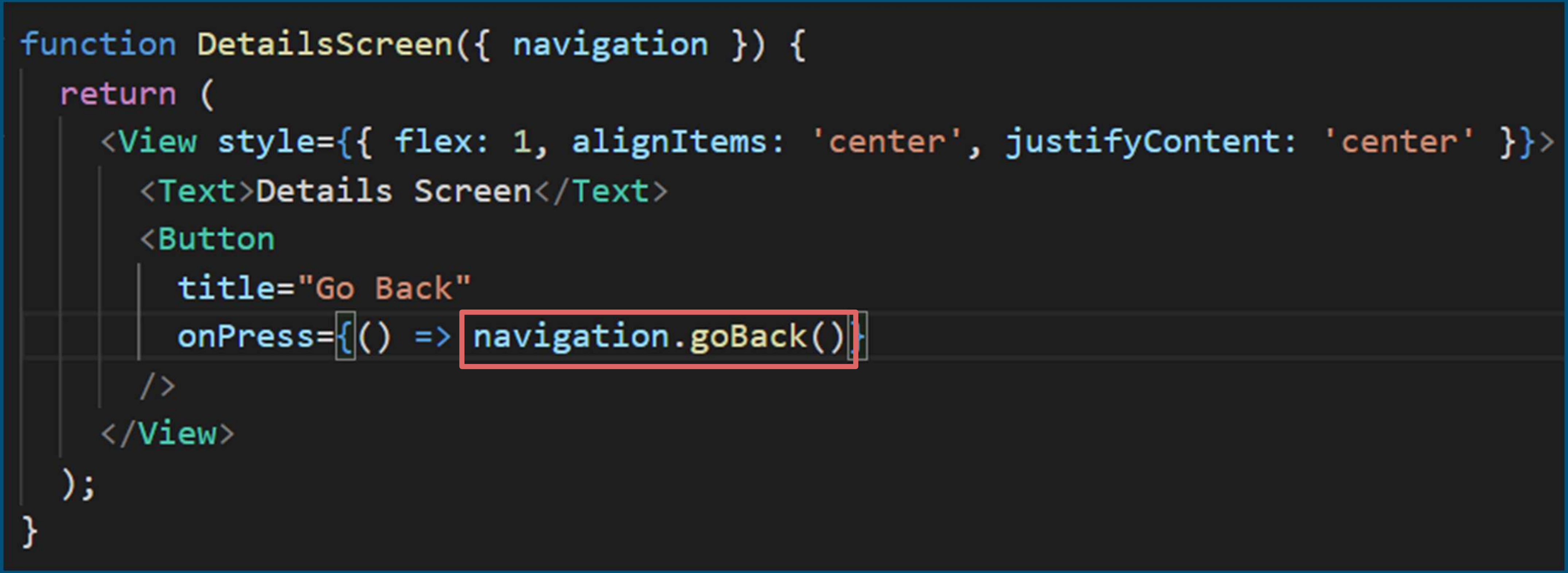
- key - Unique key of the screen. Created automatically or added while navigating to this screen.
- name - Name of the screen. Defined in navigator component hierarchy.
- path - An optional string containing the path that opened the screen, exists when the screen was opened via a deep link.
- params - An optional object containing params which is defined while navigating

Navigating between Screens



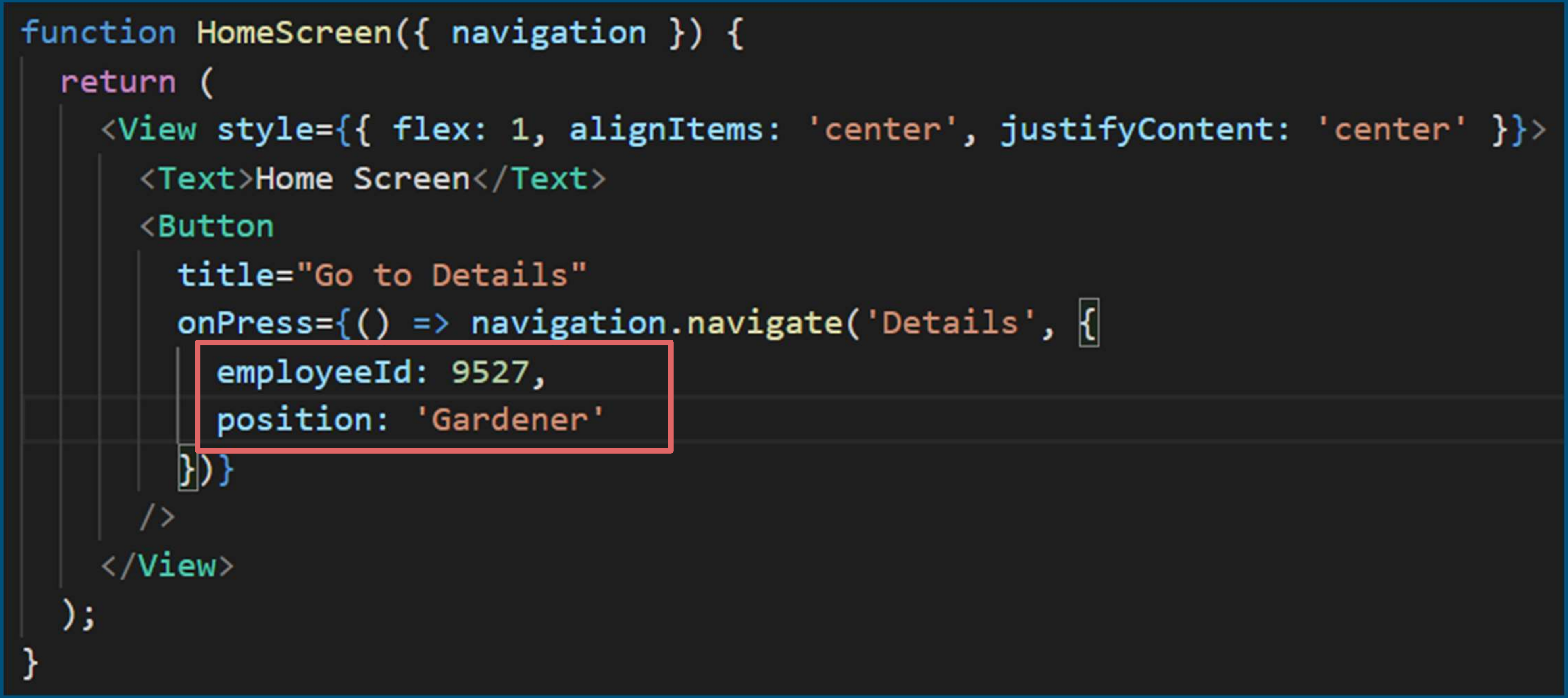
```
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details')}
      />
    </View>
  );
}
```

Navigating between Screens



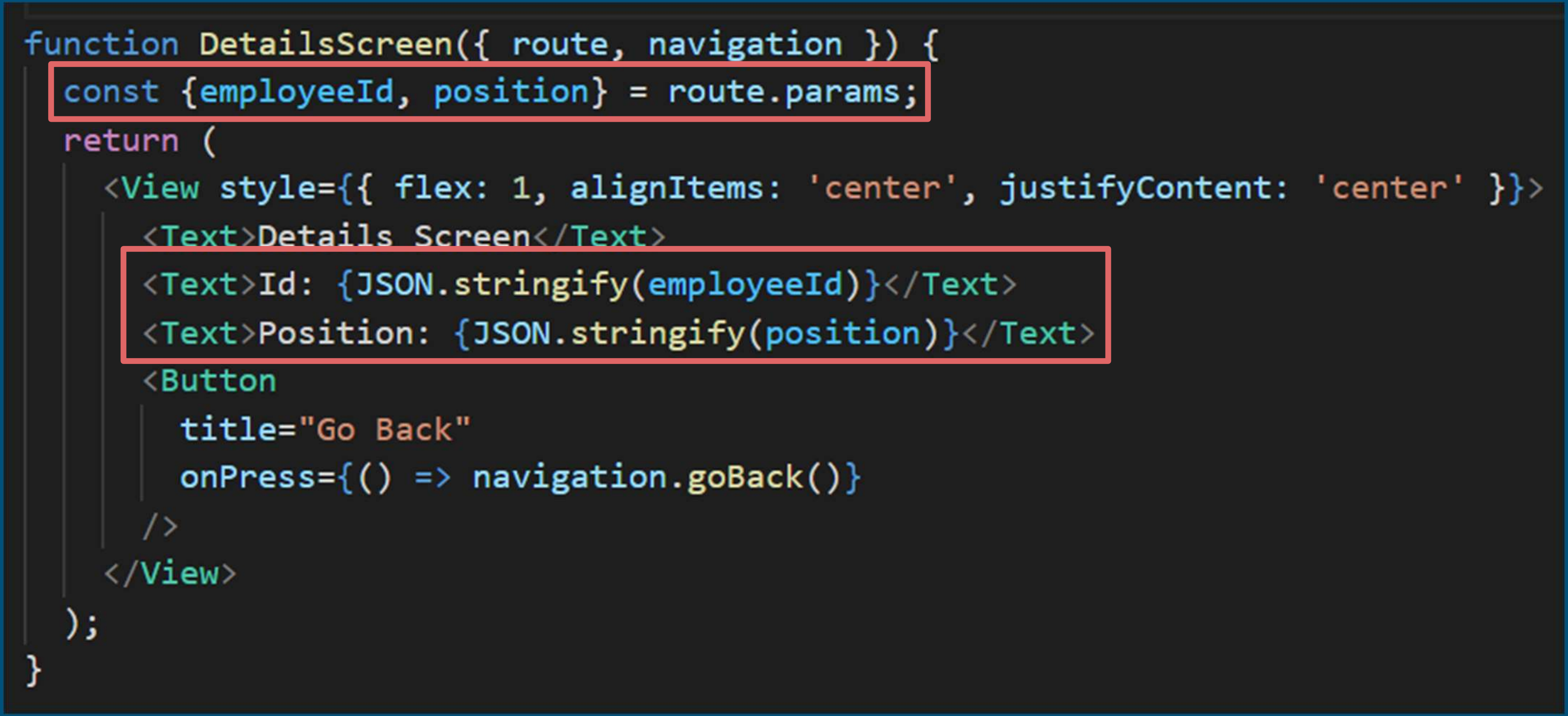
```
function DetailsScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
      <Button
        title="Go Back"
        onPress={() => navigation.goBack()}
      />
    </View>
  );
}
```

Passing Parameters to Routes



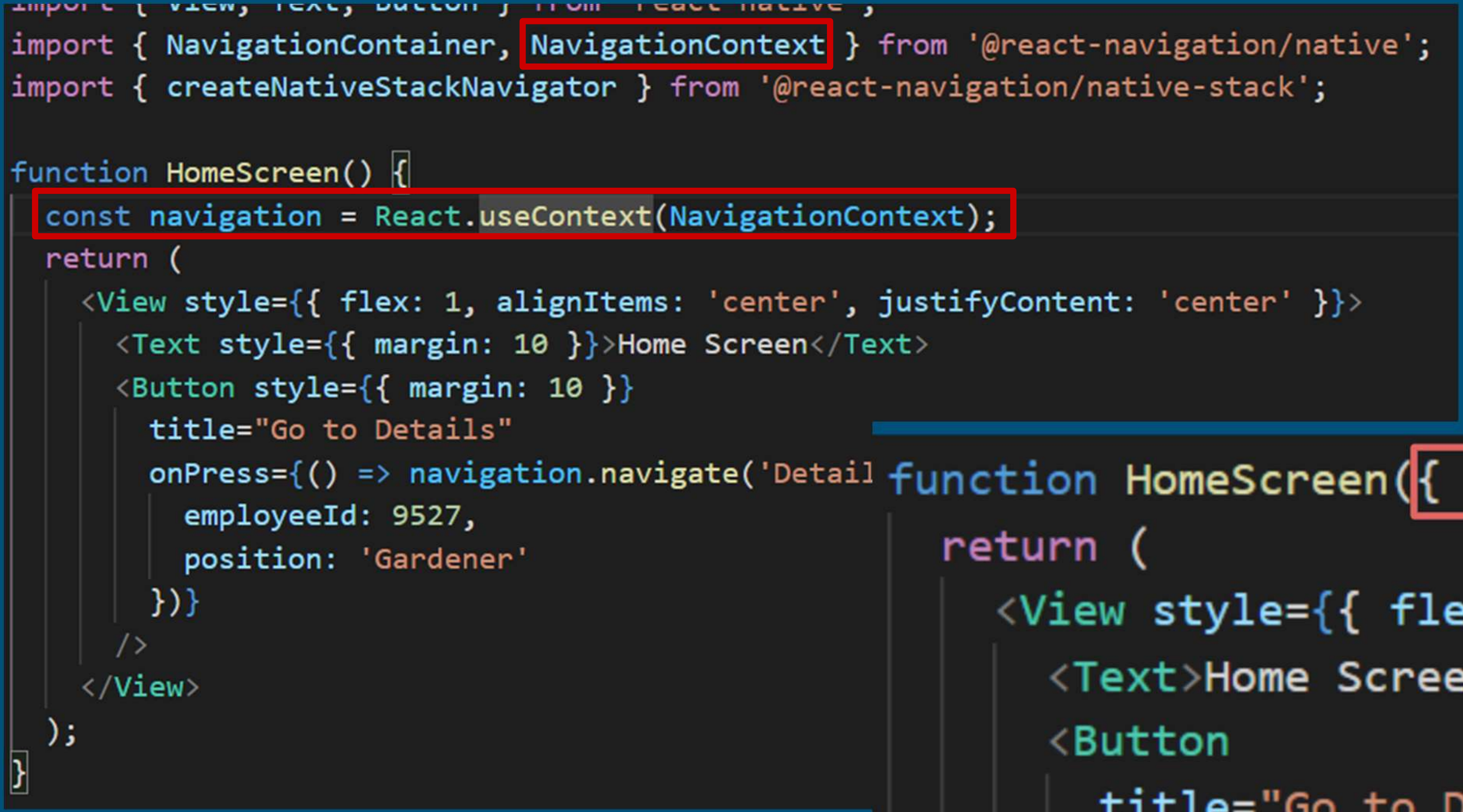
```
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Home Screen</Text>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details', {
          employeeId: 9527,
          position: 'Gardener'
        })}
      />
    </View>
  );
}
```

Passing Parameters to Routes

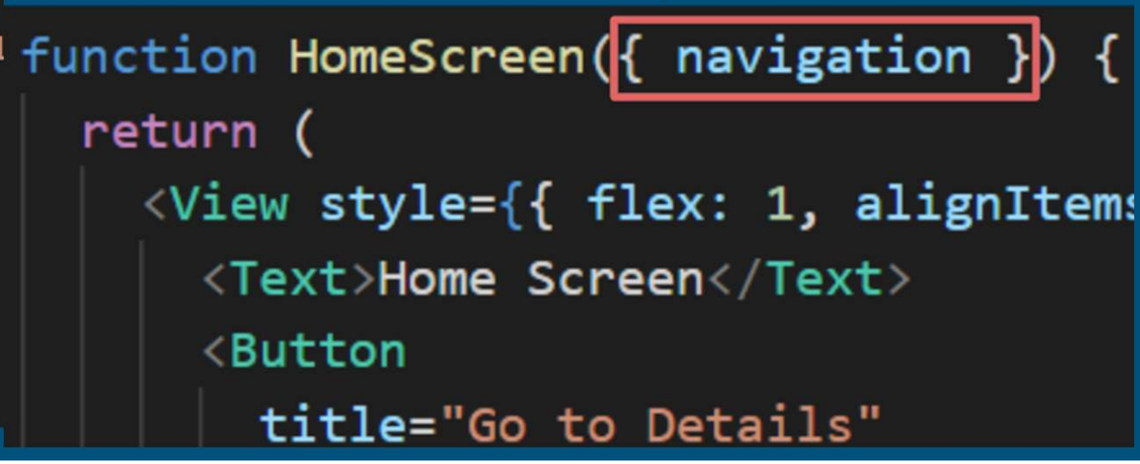


```
function DetailsScreen({ route, navigation }) {
  const {employeeId, position} = route.params;
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text>Details Screen</Text>
      <Text>Id: {JSON.stringify(employeeId)}</Text>
      <Text>Position: {JSON.stringify(position)}</Text>
      <Button
        title="Go Back"
        onPress={() => navigation.goBack()}
      />
    </View>
  );
}
```

Navigation Context



```
import { View, Text, Button } from 'react-native';
import { NavigationContainer, NavigationContext } from '@react-navigation/native';
import { createNativeStackNavigator } from '@react-navigation/native-stack';

function HomeScreen() {
  const navigation = React.useContext(NavigationContext);
  return (
    <View style={{ flex: 1, alignItems: 'center', justifyContent: 'center' }}>
      <Text style={{ margin: 10 }}>Home Screen</Text>
      <Button style={{ margin: 10 }}
        title="Go to Details"
        onPress={() => navigation.navigate('Detail
          employeeId: 9527,
          position: 'Gardener'
        })}
      />
    </View>
  );
}
```

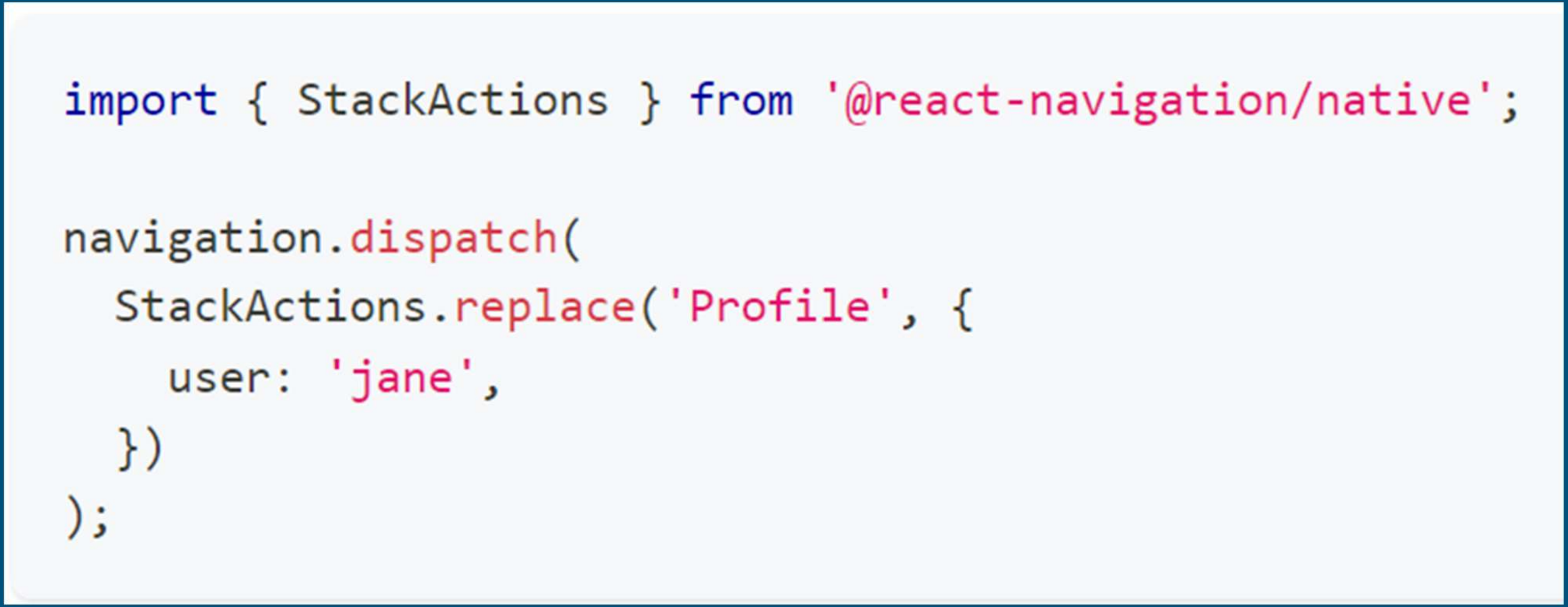


```
function HomeScreen({ navigation }) {
  return (
    <View style={{ flex: 1, alignItems
      <Text>Home Screen</Text>
      <Button
        title="Go to Details"
```

StackActions

- replace
  - name
  - params
- push
  - name
  - params
- pop – The pop action takes you back to a previous screen in the stack. It takes one optional argument (index), which allows you to specify how many screens to pop back by.
- popToTop

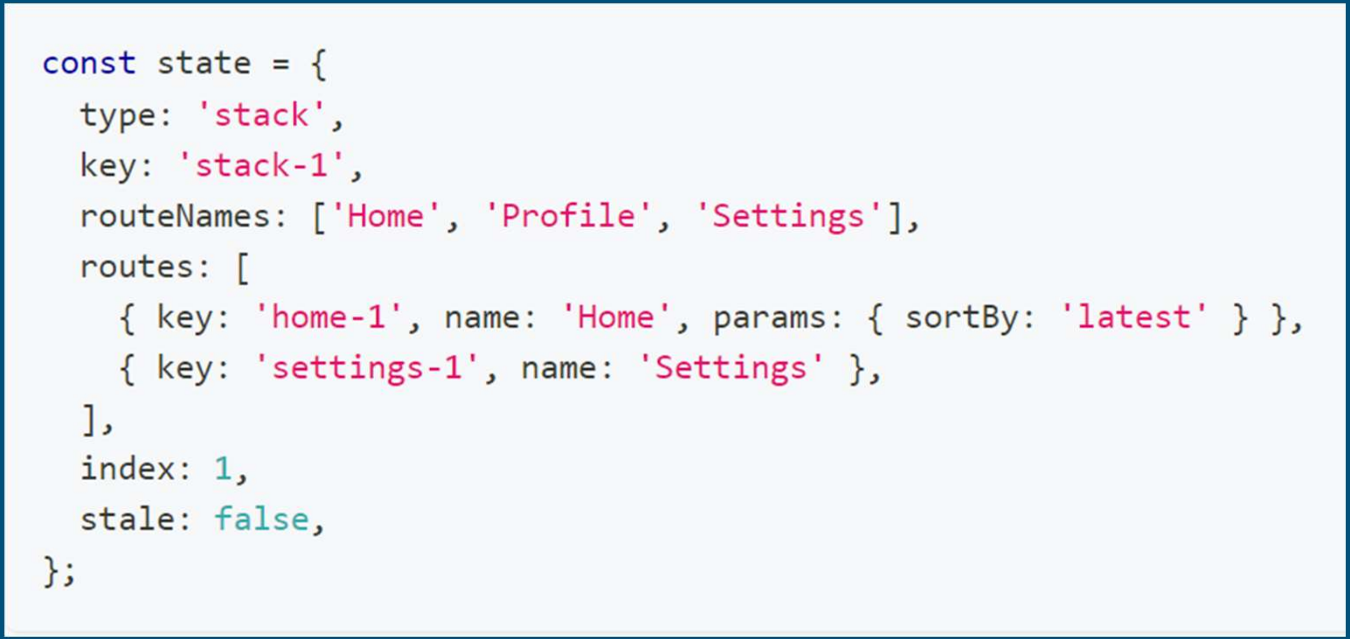
StackActions



```
import { StackActions } from '@react-navigation/native';

navigation.dispatch(
  StackActions.replace('Profile', {
    user: 'jane',
  })
);
```

Navigation State

The navigation state is the state where React Navigation stores the navigation structure and history of the app.



```
const state = {
  type: 'stack',
  key: 'stack-1',
  routeNames: ['Home', 'Profile', 'Settings'],
  routes: [
    { key: 'home-1', name: 'Home', params: { sortBy: 'latest' } },
    { key: 'settings-1', name: 'Settings' },
  ],
  index: 1,
  stale: false,
};
```

Navigation Lifecycles

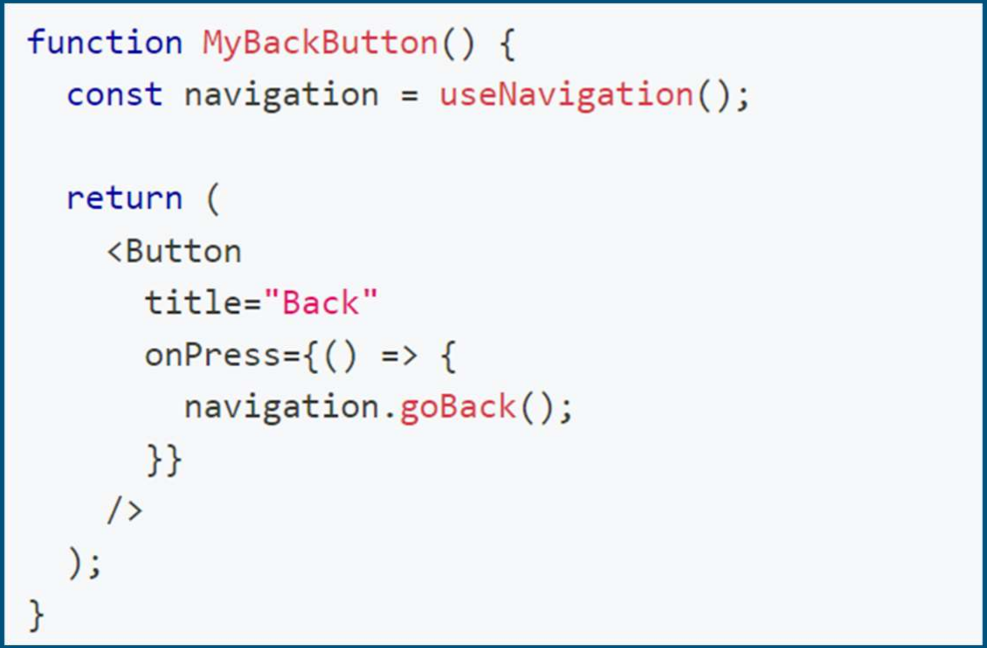
Same as React.Component

- constructor()
- componentWillMount()
- render()
- componentDidMount()
- componentWillUnmount()

Hooks

- useNavigation() – It's useful when you cannot pass the navigation prop into the component directly, or don't want to pass it in case of a deeply nested child.
- useRoute() – passing route prop into component directly.
- useTheme() – theming, color, styles.
- useNavigationState() – index, history of screens.

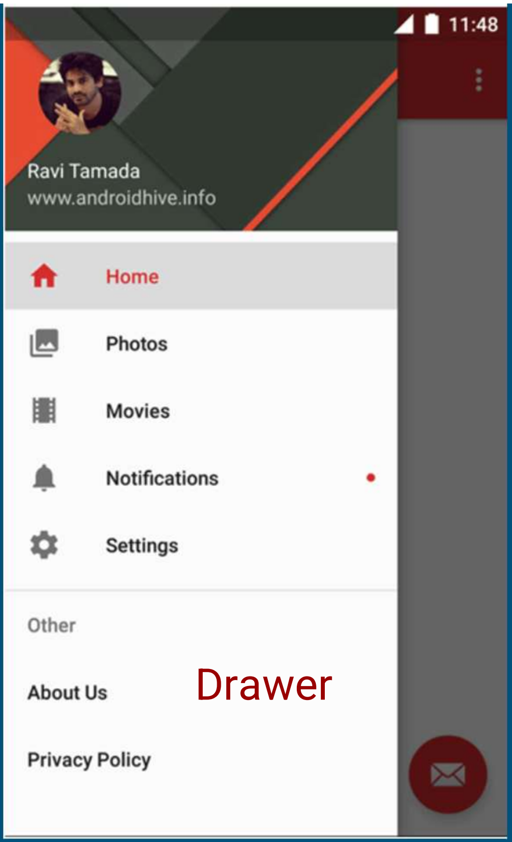
Hooks – useNavigation()

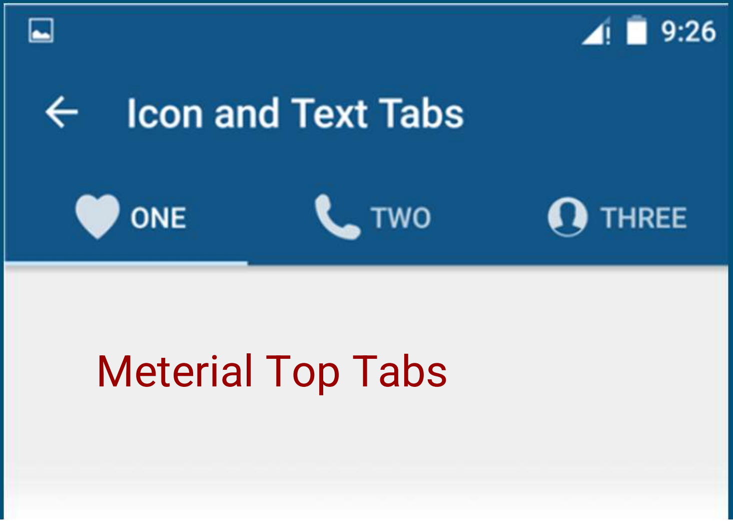


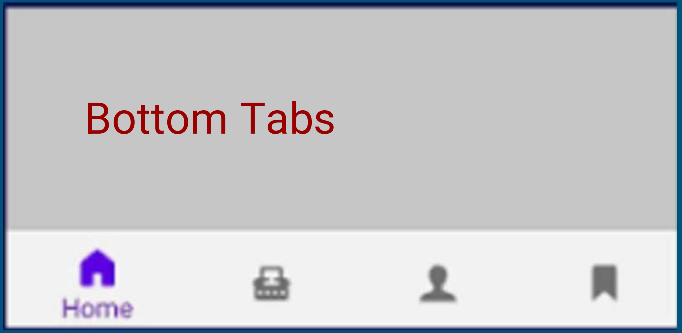
```
function MyBackButton() {
  const navigation = useNavigation();

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

Navigators

- Stack
- Drawer (Side Menu)
- Bottom Tabs (Bottom TabView)
- Meterial Top Tabs (ViewPager)

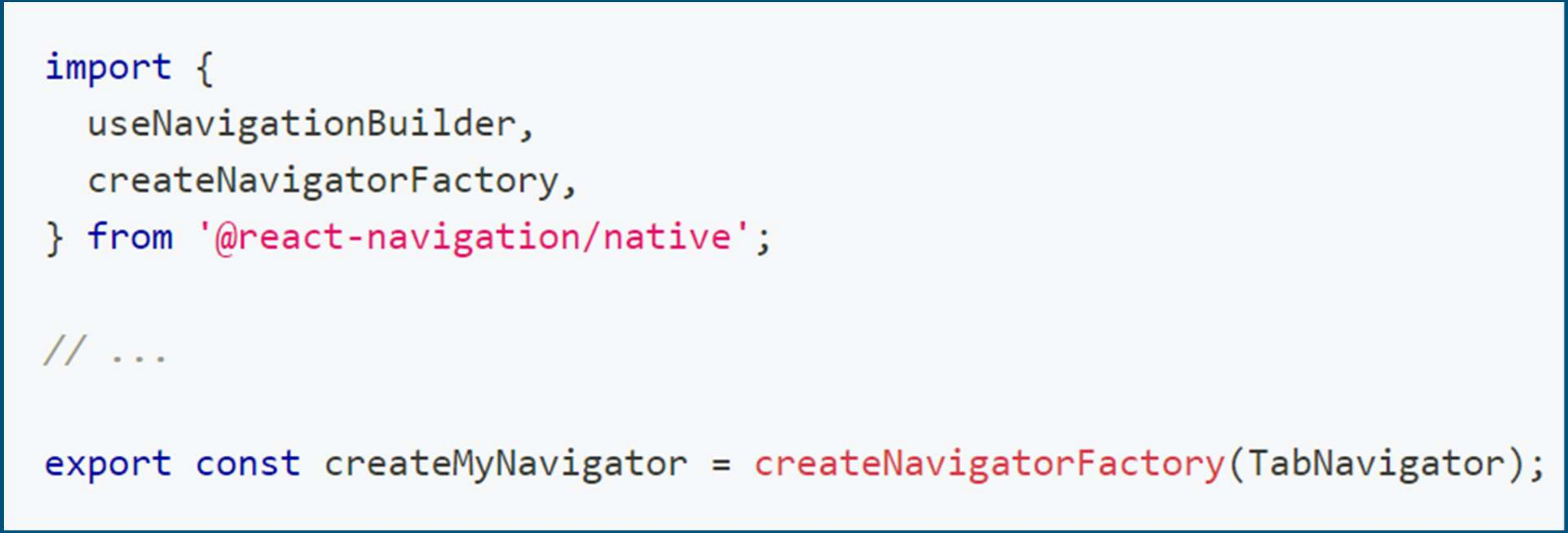






Custom Navigator

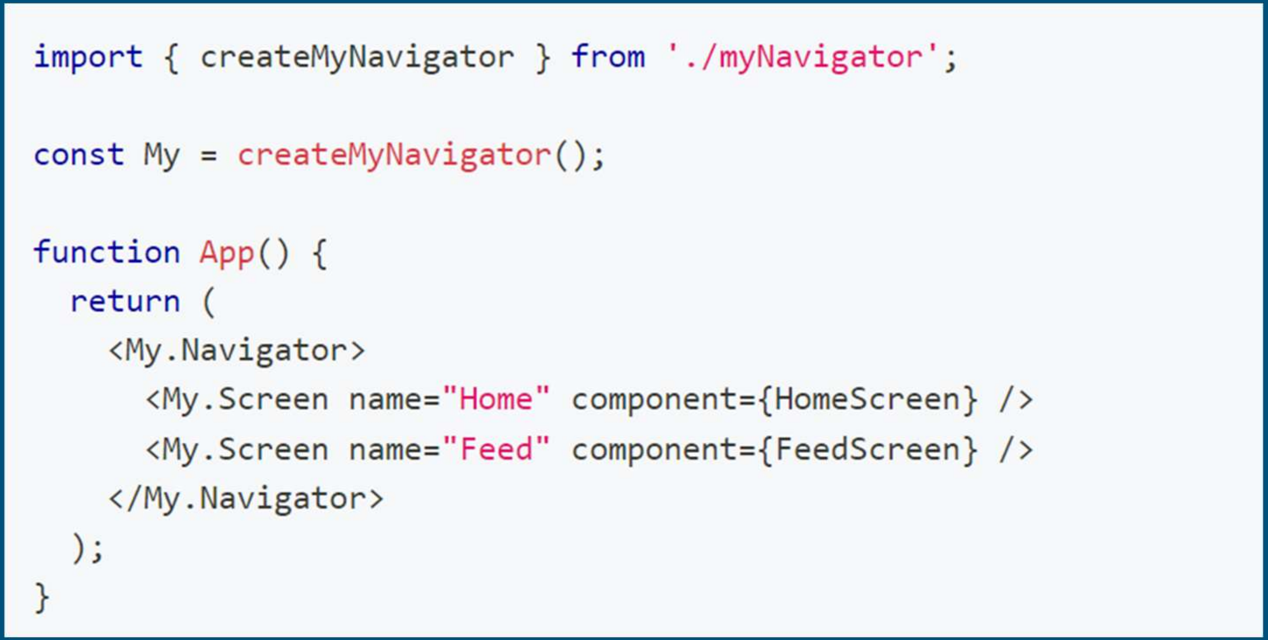
- useNavigationBuilder – This hook allows a component to hook into React Navigation.
- createNavigatorFactory – Used to create a function that will Navigator and Screen pair.



```
import {
  useNavigationBuilder,
  createNavigatorFactory,
} from '@react-navigation/native';

// ...

export const createMyNavigator = createNavigatorFactory(TabNavigator);
```

Custom Navigator

Then it can be used like this:



```
import { createMyNavigator } from './myNavigator';

const My = createMyNavigator();

function App() {
  return (
    <My.Navigator>
      <My.Screen name="Home" component={HomeScreen} />
      <My.Screen name="Feed" component={FeedScreen} />
    </My.Navigator>
  );
}
```

Thank you