



WEBINAR

# React Native Navigation

[mark@objectcomputing.com](mailto:mark@objectcomputing.com)

© 2019, Object Computing, Inc. (OCI). All rights reserved. No part of these notes may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior, written permission of Object Computing, Inc. (OCI)

[objectcomputing.com](http://objectcomputing.com)

# NAVIGATION ...

---



- Like routing in a React app
  - manages screen transitions and tracks navigation history
  - unlike web pages, React Native screens do not have URLs
- Kinds of navigation
  - **stack** - go forward and backward by pushing pages onto and popping pages from a stack
  - **switch** - vertical set of buttons that navigate to different screens when pressed
  - **tab** - set of tabs at top or bottom that navigate to different screens when pressed
  - **drawer** - side menu of options like a hamburger menu



## ... NAVIGATION

---



- Top app component can be a navigation component
- Can combine approaches within an app
  - ex. top navigation can be tab-based and some tabs can use stack-based for sub-pages



# NAVIGATION LIBRARIES

---



- None provided by React Native
- **React Native Navigation**
- **React Navigation**



## REACT NATIVE NAVIGATION

---



- Cross-platform library at <https://github.com/wix/react-native-navigation>
- Not as easy, but provides native look and feel
- Not covered here



# REACT NAVIGATION

---



- Cross-platform library at <https://facebook.github.io/react-native/docs/navigation#react-navigation>
- Can be used with React Native AND React
- Easy to use
- Our focus



## REACT NAVIGATION SETUP

---



- **npm install react-navigation**
- If using Expo CLI, no other steps needed
- If using React Native CLI
  - **react-native link react-native-gesture-handler**
- Additional steps are needed to support Android
  - see <https://reactnavigation.org/docs/en/getting-started.html>



# SCREENS



- Each “screen” in navigation is implemented by a React component
- These are passed props **navigation** and **screenProps**
- **navigation** prop has many methods used to navigate to other screens
  - **navigation.navigate(screenName, params)**
    - optional second argument is object containing data to be passed
    - destination component can access params in **navigation.state.params**
  - **navigation.goBack(fromScreenName)**
    - omit **fromScreenName** to go back one screen in history

if history contains **A**, **B**, and **C**  
where **C** is current screen,  
to go back to **A** use  
**navigation.goBack('B')**

stack navigators  
use **pop** method  
instead of **goBack**

## NAVIGATION OPTIONS ...

---



- Some options that can be specified
  - **title** - title text
  - **headerTitle** - set to component instance to use that instead of string **title** (ex. `<Icon .../>`)
  - **headerBackTitle** - defaults to screen name; define in “back” component
  - **headerStyle** - style object (ex. can set **backgroundColor**)
  - **headerTintColor** - font color for title and back button
  - **headerTitleStyle** - style object applied to header title
  - and many more; see  
<https://reactnavigation.org/docs/en/stack-navigator.html#navigationoptions-for-screens-inside-of-the-navigator>

## ... NAVIGATION OPTIONS

---



- Options on previous slide can go in two places
- 1) navigator config object passed as second argument to a **create\*Navigator** function
  - **defaultNavigationOptions** property is an object containing navigation options
- 2) static **navigationOptions** property of each screen component
- Both are used in stack navigator example next



## NAVIGATION LIFECYCLE METHODS



- Can optional implement these methods in components to execute code at before/after focus changes
- **willFocus** - called before component gains focus
- **didFocus** - called after component gains focus
- **willBlur** - called before component loses focus
- **didBlur** - called after component loses focus



# ROUTE PARAMETERS



- To pass params to a component when navigating to it

```
this.props.navigation.navigate(route-name, params-object);
```

**params-object** is an object whose properties are the params to pass

- To access params inside a component

```
this.props.navigation.getParam(param-name, default-value);
```

**default-value** is returned if **param-name** is not found

- To modify params inside a component

```
this.props.navigation.setParams({param-name: value, ...});
```

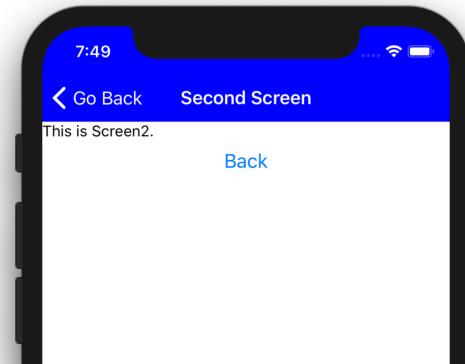
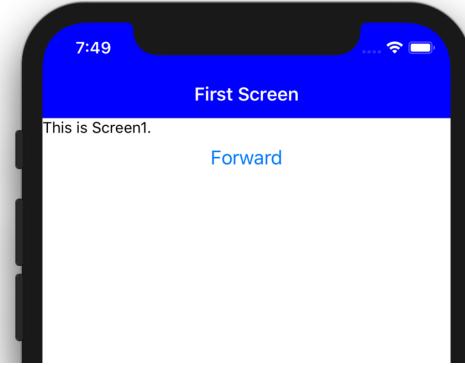
- component will re-render

passed an object whose properties are the params to be set



## STACK NAVIGATOR ...

- Push/pop screens on/off stack
- Only navigator that supports a header at top of all screens
  - to get a header in screens of other navigators, nest them inside a stack navigator or use **Header** component from React Native Elements
- This example has two pages in stack
  - **Screen1** and **Screen2**



## STACK NAVIGATOR ...

---



- **navigation** prop is passed to all stack components
- Methods specific to stack navigator
  - **push** - new route onto stack
  - **pop** - go back in stack
  - **popToTop** - go to top of stack
  - **replace** - replace current route with new one rather than add to stack
  - **reset** - wipe navigator state and replace with result of several “actions” (specified with an array)
  - **dismiss** - used in a nested stack to return to parent stack

most commonly used are  
**push**, **pop**, and **popToTop**



## ... STACK NAVIGATOR ...

```
import React, {Component} from 'react';
import {Button, StatusBar, Text, View} from 'react-native';
import {createAppContainer, createStackNavigator} from 'react-navigation';

class Screen1 extends Component {
  static navigationOptions = {
    headerBackTitle: 'Go Back',
    title: 'Screen1'
  };

  render() {
    const {navigation} = this.props;
    const params = {foo: true, bar: 1, qux: 'text'};
    return (
      <View>
        /* make time and signal indicator in status bar light */
        <StatusBar barStyle="light-content" /> ←
        without this status bar text
        will be black which is hard to
        read against a blue background

        <Text>This is Screen1.</Text>
        <Button
          onPress={() => navigation.push('Screen2', params)}
          title="Forward"
        />
      </View>
    );
  }
}
```

defining a screen component with a class

**navigationOptions** value can also be a function  
that is passed props (including **navigation**) and  
returns an object containing navigation options



## ... STACK NAVIGATOR ...



```
// params passed from Screen1 are in navigation.state.params
const Screen2 = ({navigation}) => (
  <View>
    <Text>This is Screen2.</Text>
    <Button onPress={() => navigation.pop()} title="Back" />
    /* "< screen-name" button is also provided in header */
  </View>
);
Screen2.navigationOptions = {
  title: 'Screen2'
};
```

defining a screen component with an arrow function

alternative to using a static property

## ... STACK NAVIGATOR



keys are screen names and  
values are screen components

```
const StackNav = createStackNavigator(  
{  
  Screen1, // can add options to each of these  
  Screen2: {screen: Screen2} // long form  
},  
{  
  defaultNavigationOptions: {  
    headerStyle: {  
      backgroundColor: 'blue'  
    },  
    headerTintColor: 'white'  
  },  
  mode: 'card' ←  
}  
);
```

can add `transitionConfig` property  
here to customize screen transitions

defaults for all screen components  
that do not override these

When `mode` is '`card`' (default) next  
screen transitions from right to left.  
When `mode` is '`modal`', next screen  
transitions from bottom to top.

## ... STACK NAVIGATOR



```
const AppContainer = createStackNavigator(StackNav);

export default class App extends Component {
  // Can add use of things like Redux and Context API.
  render() {
    return <AppContainer />;
  }
}
```

## SWITCH NAVIGATOR

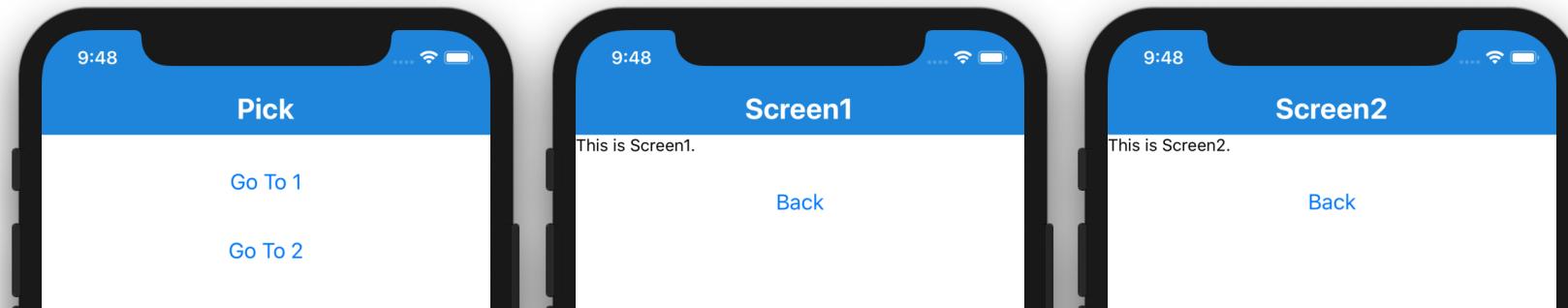
---



- Shows one screen at a time
- Does not support going back to previous screen
  - **must** use `navigator.navigate` to go to an explicit screen
- Does not provide a header
  - can use **Header** element from React Native Elements



## SWITCH NAVIGATOR EXAMPLE ...



## ... SWITCH NAVIGATOR EXAMPLE ...



```
import React from 'react';
import {Button, StatusBar, Text, View} from 'react-native';
import {Header} from 'react-native-elements';
import {createAppContainer, createSwitchNavigator} from 'react-navigation';

const getHeader = navigation => (
  <>
    <StatusBar barStyle="light-content" />
    <Header
      centerComponent={{
        text: navigation.state.routeName,
        style: {color: 'white', fontSize: 24, fontWeight: 'bold'}
      }}
    />
  </>
);

```

React  
fragment

## ... SWITCH NAVIGATOR EXAMPLE ...



```
const getNavButton = (navigation, text, screenName) => (
  <View style={{marginTop: 20}}>
    <Button onPress={() => navigation.navigate(screenName)} title={text} />
  </View>
);

const Pick = ({navigation}) => (
  <View>
    {getHeader(navigation)}
    {getNavButton(navigation, 'Go To 1', 'Screen1')}
    {getNavButton(navigation, 'Go To 2', 'Screen2')}
  </View>
);
```

## ... SWITCH NAVIGATOR EXAMPLE ...



```
const Screen1 = ({navigation}) => (
  <View>
    {getHeader(navigation)}
    <Text>This is {navigation.state.routeName}.</Text>
    {getNavButton(navigation, 'Back', 'Pick')}
  </View>
);

const Screen2 = ({navigation}) => (
  <View>
    {getHeader(navigation)}
    <Text>This is {navigation.state.routeName}.</Text>
    {getNavButton(navigation, 'Back', 'Pick')}
  </View>
);
```

## ... SWITCH NAVIGATOR EXAMPLE



```
const SwitchNav = createSwitchNavigator({Pick, Screen1, Screen2});  
  
const AppContainer = createAppContainer(SwitchNav);  
  
export default () => <AppContainer />;
```



# TAB NAVIGATOR

---



- Top tab options
  - `createTabNavigator` - deprecated
  - `createMaterialTopTabNavigator` - uses Material Design; appropriate for Android
- Bottom tab options
  - `createBottomTabNavigator` - best option in most cases
  - `createMaterialBottomTabNavigator` - uses Material Design; appropriate for Android



## BOTTOM TAB NAVIGATOR

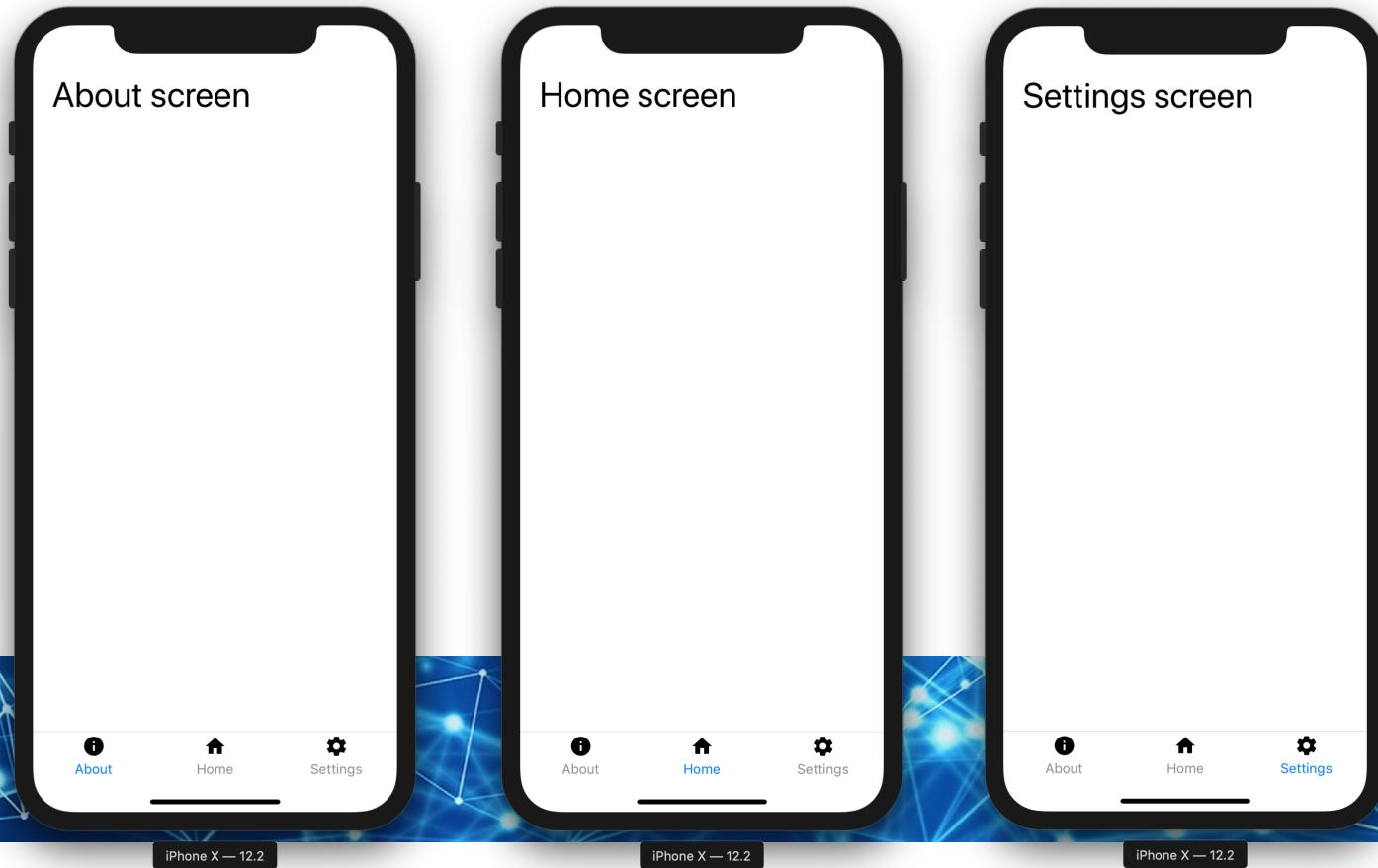
---



- Common to have white background with colored text and/or icons
  - gray for inactive
- Can use Icon element from React Native Elements



## BOTTOM TAB NAVIGATOR EXAMPLE ...



## BOTTOM TAB NAVIGATOR EXAMPLE ...



```
import React from 'react';
import {StyleSheet, Text, View} from 'react-native';
import {Icon} from 'react-native-elements';
import {createBottomTabNavigator, createAppContainer} from 'react-navigation';

const getTitle = text => <Text style={styles.title}>{text}</Text>

const About = () => (
  <View style={styles.container}>{getTitle('About screen')}</View>
);

const Home = () => (
  <View style={styles.container}>{getTitle('Home screen')}</View>
);

const Settings = () => (
  <View style={styles.container}>{getTitle('Settings screen')}</View>
);
```

## ... BOTTOM TAB NAVIGATOR EXAMPLE ...



```
const getRouteConfig = (screen, icon) => ({
  screen,
  navigationOptions: {
    tabBarIcon: <Icon name={icon}>/>
  }
});

const tabBarOptions = {
  initialRouteName: 'Home',
  tabBarOptions: {
    labelStyle: {fontSize: 14}
  }
};
```



## ... BOTTOM TAB NAVIGATOR EXAMPLE



```
const TabNav = createBottomTabNavigator(
{
  About: getRouteConfig(About, 'info'),
  Home: getRouteConfig(Home, 'home'),
  Settings: getRouteConfig(Settings, 'cog')
},
tabBarOptions
);

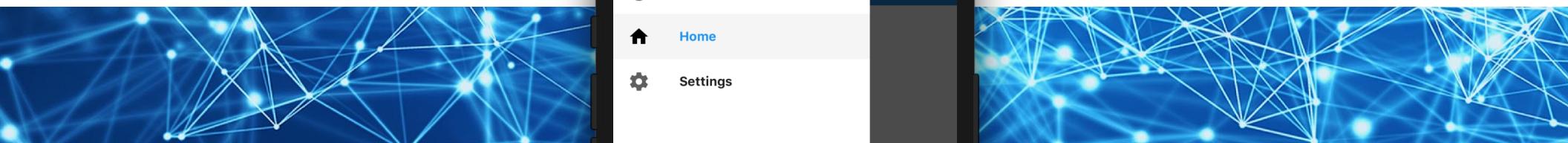
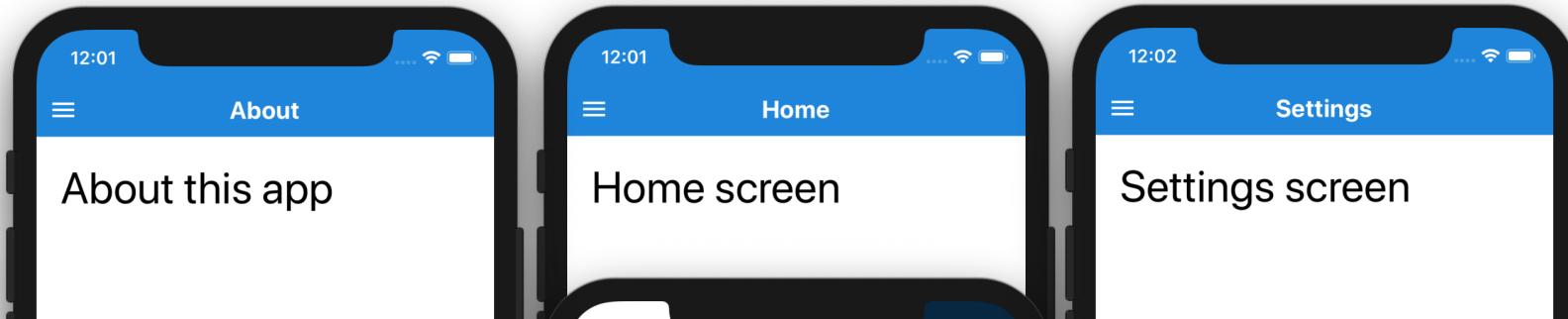
const styles = StyleSheet.create({
  container: {
    paddingHorizontal: 20,
    paddingTop: 50
  },
  title: {fontSize: 36}
});

export default createAppContainer(TabNav);
```

## DRAWER NAVIGATOR



- Provides drawer that slides out from right containing navigation buttons



## DRAWER NAVIGATOR EXAMPLE ...



```
import React from 'react';
import {Button, StyleSheet, Text, View} from 'react-native';
import {Header, Icon} from 'react-native-elements';
import {createAppContainer, createDrawerNavigator} from 'react-navigation';

const Hamburger = ({navigation}) => (
  <Icon color="white" name="menu" onPress={() => navigation.toggleDrawer()} />
);
```



## ... DRAWER NAVIGATOR EXAMPLE ...



```
const MyHeader = ({navigation, title}) => {
  return (
    <Header
      leftComponent={<Hamburger navigation={navigation} />}
      centerComponent={(
        text: title,
        style: {color: 'white', fontSize: 20, fontWeight: 'bold'}
      )}
      statusBarProps={{barStyle: 'light-content'}}
    />
  );
};

const getTitle = text => <Text style={styles.title}>{text}</Text>;
```

## ... DRAWER NAVIGATOR EXAMPLE ...



```
const About = ({navigation}) => (
  <View style={styles.container}>
    <MyHeader navigation={navigation} title="About" />
    {getTitle('About this app')}
  </View>
);
About.navigationOptions = {
  drawerIcon: <Icon name="info" />
};

const Home = ({navigation}) => (
  <View style={styles.container}>
    <MyHeader navigation={navigation} title="Home" />
    {getTitle('Home screen')}
  </View>
);
Home.navigationOptions = {
  drawerIcon: <Icon name="home" />
};
```

## ... DRAWER NAVIGATOR EXAMPLE ...



```
const Settings = ({navigation}) => (
  <View style={styles.container}>
    <MyHeader navigation={navigation} title="Settings" />
    {getTitle('Settings screen')}
  </View>
);
Settings.navigationOptions = {
  drawerIcon: <Icon name="settings" />
};

const drawerOptions = {initialRouteName: 'Home'};
const DrawerNav = createDrawerNavigator(
  {About, Home, Settings},
  drawerOptions
);
```

## ... DRAWER NAVIGATOR EXAMPLE ...



```
const styles = StyleSheet.create({
  container: {},
  title: {
    fontSize: 36,
    padding: 20
  }
});

export default createAppContainer(DrawerNav);
```



# COMBINING NAVIGATORS

TRY THIS!



```
import {createBottomTabNavigator, createStackNavigator} from 'react-navigation';
const options = {
  navigationOptions: {
    headerStyle: {
      backgroundColor: someColor
    },
    headerTintColor: someColor
  }
};
const StackNav = createStackNavigator({
  Name1: {screen: Component1},
  Name2: {screen: Component2}
}, options);
const Tabs = createBottomTabNavigator({
  Tab1Text: {screen: StackNav},
  Tab2Text: {screen: SomeComponent}
});
export default Tabs;
```

import **Tabs** in another component and render it

```
<Tabs screenProps={someObj} />
```

someObj is passed to all screens;  
can include data and callback functions

```
import {createDrawerNavigator} from 'react-navigation';
const DrawerNav = createDrawerNavigation(screens);
```