React Native Navigation

... with react-navigation

tl;dr

- Navigation is giving the user access to other scenes in our app
- It is from a 3rd party but has been adopted by the community
- We declaratively describe the route organization
- You'll navigate using
  - Stack navigators
  - Tab navigators
  - Drawer navigators
- All of them are created imperatively (They're not JSX) but they create components that will be placed in other components

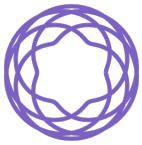
What is navigation?

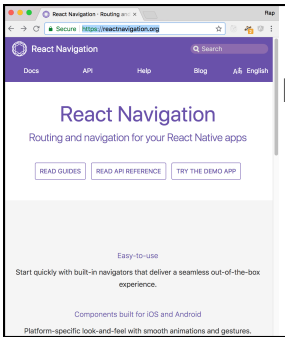
- It is the swapping out of one component for another.
- It makes the user feel like they are moving around in our app.

There is much less navigation on a device -- you don't swap views out that much. It's just the nature of devices.

Navigation

- The community solution to navigation is a standalone library that allows developers to set up the screens of an app with just a few lines of code.





It comes from ReactNavigation.org

We must install react-navigation

```
npm install react-navigation
```

All of the navigators are created imperatively

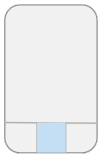
For example:

```
const dnav = createDrawerNavigator(foo, bar);
```

- They're not in JSX
- They're in JavaScript

Three types of navigators

1. StackNavigator

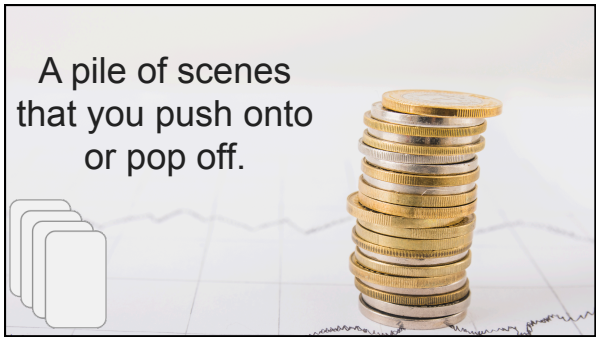


2. TabNavigator

3. DrawerNavigator

1. Stack Navigator

A pile of scenes that you push onto or pop off.



A Stack Navigator is the most fundamental navigation component

- You don't see anything different on the screen. No tabs. No menus. Whatever navigation you want done will be done manually.
- Navigation is merely ...
- Forward - Push a new screen on top of the stack (aka. the other screens).
- Back - Pop the top screen off the stack, revealing the previous one.

Make one with createStackNavigator()

This is a React Native component that you host inside another component or with React.render()

```
rc = createStackNavigator(
routingObject,
configObject)
```

Defines all the components and how we will navigate to them

Holds configuration - modifications to how navigation should work here

The Routing Object maps routes to components

The simplest routing:

```
{
 Home: Main,
 Contact: Contact,
 Buy: BuyComponent
}
```

More complete routing:

```
{
 Home: {screen:Main},
 Contact: {
  screen: Contact,
 },
 Buy: {
  screen: BuyComponent,
}
```

The Config Object

The ones you need to know:

```
{
 initialRouteName: Where we should start,
 initialRouteParams: Object for initial draw,
 ... and tons more about gestures, header
configs, and more
}
```

See the React Navigator docs at http://bit.ly/2OTSrJH for lots more options

You can create a static property called navigationOptions which will override the universal settings only when you're on that scene

What if I want different config settings per page?

```
class Foo extends Component {
  static navigationOptions = {
    header: null,
  };
  ... Rest of your component here
}
```

... or ...

```
Foo.navigationOptions = {
  title: "We are awesome",
}
```

Only navigationOptions can be overridden.

App.js

```
import { Main } from './Main';
import { Contact } from './Contact';
import { Buy } from './Buy';

const Nav = createStackNavigator({
 Home: Main,
 Contact: Contact,
 Buy: BuyComponent
});

export function App() {
 return <View>
  <StatusBar />
  <Nav />
 </View>
}
```

This ...

... can go here

The component is placed like any other

Then it's activated with props.navigation.navigate(route)

Navigable components get a ".navigation" object in their props.

.navigate(routeName)	will push that component onto the stack and the user sees it
.goBack()	Pop off the nav stack
.push()	Navigate to a route you're already on but with different props

So navigation might look like this

```
function Contact(props) {
 const {nav} = props.navigation;
 return <View>
  <Button title="Purchase"
   onPress={()=> nav.navigate('Buy')} />
  <Button title="Back to home"
   onPress={()=> nav.goBack()} />
 </View>
}
```

Passing params when navigating

Remember, When you nest components, you pass values from one to another in props:

Nice and simple

```
function inner(props) {
  console.log(props);
  return <SomeJSX />
}
function host() {
 return <Inner foo=1 bar=2 />
}
```

But when navigating, a 2nd object sends data

To send params ...

```
function Contact(props) {
const {nav} = props.navigation;
return <View>
 <Button title="Purchase"
  onPress={()=> nav.navigate('Buy',
         {item:theProduct, price:27.50 })} />
</View>
}
```

To receive params ...

```
function BuyComponent() {
const {nav} = props.navigation.state;
const theItem = nav.item;
const price = nav.getParam("price", 5.00);
return <View>
</View>
}
```

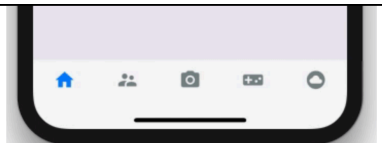
Since the two techniques are so different ...

You generally must decide beforehand if a component will be nested or navigated to

But if you must do both you can...

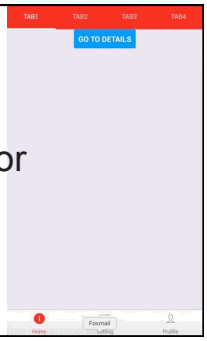
```
const {item,price} = props.navigation ?
  props.navigation.state.params : props;
```

2. Tab Navigators



BottomTabNavigator

MaterialTopTapNavigator



MaterialBottomTabNavigator

CreateBottomTabNavigator(RouteConfigs, NavConfig)
CreateMaterialBottomTabNavigator(RouteConfigs, NavConfig)
CreateMaterialTopTabNavigator(RouteConfigs, NavConfig)

```
const App = CreateBottomTabNavigator({
  Home: { screen: Home },
  Info: { screen: Info }
},{
  tabBarOptions:{
    activeTintColor: '#ff9900',
    // showIcon is only necessary for android
    showIcon: true,
  },
});
```

Adding Icons

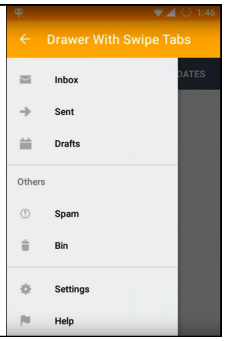
```
const Home = () => (
  <Text>Hello from Home</Text>
)

Home.navigationOptions = {
  tabBarIcon: ({ tintColor }) => (
    <Image
      source={require('./homeicon.png')}
      style={{ width: 24, height: 24, tintColor }}
    />
  ),
}
```

3. Drawer Navigator

DrawerNavigator

- The navigator is hidden, but you can swipe right to reveal it



Drawer Navigator

Step 1: import DrawerNavigator into your component

```
import {
  CreateDrawerNavigator,
} from 'react-navigation';
```

CreateDrawerNavigator

CreateDrawerNavigator(RouteConfigs, DrawerNavigatorConfig)

```
const NavComponent= DrawerNavigator({
  Home: { screen: Home },
  Info: { screen: Info }
},{
  contentOptions:{
    activeTintColor: 'pink'
  },
});
```

A simple example

```
const Home = ({ navigation }) => (
  <Text onPress={() => navigation.navigate('DrawerOpen')}
>Open Drawer</Text>
)

const Info = ({ navigation }) => (
  <Text onPress={() => navigation.navigate('DrawerOpen')}
>Open Drawer</Text>
)

const App = DrawerNavigator({
  Home: { screen: Home },
  Info: { screen: Info }
});
```

Adding Icons

```
const Home = ({ navigation }) => (
  <Text onPress={() => navigation.navigate('DrawerOpen')}>Open Drawer</Text>
)
Home.navigationOptions = {
  drawerIcon: ({ tintColor }) => (
    <Image
      source={require('./homeicon.png')}
      style={{ width: 24, height: 24, tintColor }}
    />
  ),
}
```

tl;dr

- Navigation is giving the user access to other scenes in our app
- It is from a 3rd party but has been adopted by the community
- We declaratively describe the route organization
- You'll navigate using
  - Stack navigators
  - Tab navigators
  - Drawer navigators
- All of them are created imperatively (They're not JSX) but they create components that will be placed in other components

Further Study

- React Navigation Docs
  - https://reactnavigation.org/docs/getting-started.html
- Daniel Merril's Medium series
  - http://bit.ly/react-navigation-up-and-running
  - (From where I got the navigator icons)