<https://youtu.be/uMbBR5nu3eg>

## Stack Navigator v1

When pressing an item in, say, an index view, we expect to go to a new screen with details on that item. React Navigation offers another navigator to do just that! With Stack Navigator, new screens are added and removed as a stack. This places screens on top of one another in a "last in, first out" manner, similar to Array's push() and pop() methods.

StackNavigator's usage is largely similar to that of TabNavigator. But rather than passing in an object of different tabs, we pass in an object of the different screens that should be available in that stack.

## Stack Navigator v2

StackNavigator has been deprecated in favor of [**createStackNavigator**](https://reactnavigation.org/docs/en/stack-navigator.html), which is functionally identical but clearly communicates that it's a function that returns a component.

According to the [**documentation**](https://reactnavigation.org/blog/), the new [**StackNavigator**](https://reactnavigation.org/docs/en/stack-navigator.html) is “less pushy”:

[**Each time you call push we add a new route to the navigation stack. When you call navigate, it first tries to find an existing route with that name, and only pushes a new route if there isn't yet one on the stack.**](https://reactnavigation.org/docs/en/navigating.html)

[**Let's suppose that we actually want to add another details screen. This is pretty common in cases where you pass in some unique data to each route (more on that later when we talk about params!). To do this, we can change navigate to push. This allows us to express the intent to add another route regardless of the existing navigation history.**](https://reactnavigation.org/docs/en/navigating.html)

Let's see how we'd use the Stack Navigator from React Navigation v2.

First, go ahead an import createStackNavigator from react-navigation.

**import** { createStackNavigator } **from** 'react-navigation';

Say we have two basic components, Home and Dashboard:

Let's see how we'd use the Stack Navigator from React Navigation v2.

First, go ahead an import createStackNavigator from react-navigation.

**import** { createStackNavigator } **from** 'react-navigation';

Say we have two basic components, Home and Dashboard:

**const** Home = ({ navigation }) => (

<View>

<Text>This is the Home view</Text>

<TouchableOpacity onPress={() => navigation.navigate('Dashboard')}>

<Text>Press here for the Dashboard</Text>

</TouchableOpacity>

</View>

);

**const** Dashboard = () => (

<View>

<Text>This is the Dashboard</Text>

</View>

);

Note that a navigation prop is passed to the stateless functional Home component, which allows navigation to another route. Once this is done, we can pass an object into createStackNavigatorsimilar to how we did for createBottomTabNavigator:

**const** Stack = createStackNavigator({

Home: {

screen: Home

},

Dashboard: {

screen: Dashboard

}

})

The return value of passing an object into createStackNavigator is a component as well, and we can render it as such!

*// App.js*

*// ...*

**export** **default** **class** **App** **extends** **React**.**Component** {

render() {

**return** (

<Stack />

);

}

}

[**Stack Navigator**](https://reactnavigation.org/docs/en/stack-navigator.html) and [**Tab Navigator**](https://reactnavigation.org/docs/en/bottom-tab-navigator.html) often go hand-in-hand. Since they each return components, you'll often see one nested within the other. Let's see this in action as we implement this into UdaciFitness!

<https://youtu.be/PkoZ__6NPE8>

<https://youtu.be/JosvkjGlt30>

<https://youtu.be/-c5FZCh5LNo>

<https://youtu.be/Hv_EbcrmbDY>

<https://youtu.be/_nRJsJ2-zgY>

## Summary

React Navigation's Stack Navigator is another customizable navigation option based on adding and removing new screens to a stack. Its API is similar to that of the Tab Navigator; it takes in an object that defines all screens, then returns a component. Since both the Stack Navigator and the Tab Navigator both return components, a common practice is to nest these navigators within one another.

In the next section, we'll take a look at the Drawer Navigator, in which screens are switched from a drawer that pops out from the side of the screen!

### Further Research

* [**Stack Navigation in React Native**](https://medium.com/@swathylenjini/stack-navigation-in-react-native-2cd00374ff3a)
* [**StackNavigator v1**](https://v1.reactnavigation.org/docs/stack-navigator.html) from the React Navigator docs
* [**StackNavigator v2**](https://reactnavigation.org/docs/en/stack-navigator.html) from the React Navigator docs

**Drawer Navigator**

<https://youtu.be/rxb47NRwix8>

## Drawer Navigator v1

React Navigation offers one more basic navigator to create custom navigation through React Native apps: the DrawerNavigator. While TabNavigator uses tabs to help users navigate to specific screens, DrawerNavigator uses a drawer-like menu that slides in from the side of the screen. While we won't be implementing this into UdaciFitness -- it's still important to know and fairly common in React Native applications!

To use DrawerNavigator, be sure to install version 1 of react-navigation and import the following from react-navigation:

**import** { DrawerNavigator } **from** 'react-navigation';

Luckily, many of the same philosophies shared by StackNavigator and TabNavigator apply here as well! Let's check out two basic components again and see how DrawerNavigator renders them:

**const** Home = ({ navigation }) => (

<View>

<Text>This is the Home view</Text>

<TouchableOpacity onPress={() => navigation.navigate('DrawerOpen')}>

<Text>Press here to open the drawer!</Text>

</TouchableOpacity>

</View>

);

**const** Dashboard = ({ navigation }) => (

<View>

<Text>This is the Dashboard view</Text>

<TouchableOpacity onPress={() => navigation.navigate('DrawerOpen')}>

<Text>Press here to open the drawer!</Text>

</TouchableOpacity>

</View>

);

Note that rather than routing to another component, each TouchableOpacity wrapper opens the drawer. Likewise, 'DrawerClose' can be used to close the drawer. To simplify things, React Navigation also offers 'DrawerToggle' to automatically select which navigation is appropriate based on the current drawer state.

Similar to TabNavigator and StackNavigator, we can then pass an object into DrawerNavigator, render the component returned, and we're all set!

**const** Drawer = DrawerNavigator({

Home: {

screen: Home

},

Dashboard: {

screen: Dashboard

}

});

*// App.js*

*// ...*

**export** **default** **class** **App** **extends** **React**.**Component** {

render() {

**return** (

<Drawer />

);

}

}

## Drawer Navigator v2

DrawerNavigator has been deprecated in favor of [**createDrawerNavigator**](https://reactnavigation.org/docs/en/drawer-navigator.html), which is functionally identical but clearly communicates that it's a function that returns a component.

According to the documentation:

[**Rather than opening a drawer with navigation.navigate(‘DrawerOpen’), you can now call navigation.openDrawer(). Other methods are closeDrawer() and toggleDrawer().**](https://reactnavigation.org/blog/)

Many native applications include a drawer menu to help users navigate through the app. The Drawer Navigator offers us a simple but powerful way to implement that in React Native apps!

What is true about the Drawer Navigator? Please select all that apply:

* Drawer Navigator's drawer generally spans the height of the screen.
* The component returned by createStackNavigator can be nested inside a Drawer Navigator.