StackNavigator

Provides a way for your app to transition between screens where each new screen is placed on top of a stack.

By default the StackNavigator is configured to have the familiar iOS and Android look & feel: new screens slide in from the right on iOS, fade in from the bottom on Android. On iOS the StackNavigator can also be configured to a modal style where screens slide in from the bottom.

```
class MyHomeScreen extends React.Component {
  static navigationOptions = {
    title: 'Home',
  }

  render() {
    return (
      <Button
        onPress={() => this.props.navigation.navigate(
'Profile', {name: 'Lucy'})}
        title="Go to Lucy's profile"
      />
    );
  }
}

const ModalStack = StackNavigator({
  Home: {
    screen: MyHomeScreen,
  },
  Profile: {
    path: 'people/:name',
    screen: MyProfileScreen,
  },
});
```

API Definition

```
StackNavigator(RouteConfigs, StackNavigatorConfig)
```

RouteConfigs

```
StackNavigator({

  // For each screen that you can navigate to, create a new entry like this:
  Profile: {

    // `ProfileScreen` is a React component that will be the main content of the screen.
    screen: ProfileScreen,
    // When `ProfileScreen` is loaded by the StackNavigator, it will be given a `navigation` prop.

    // Optional: When deep linking or using react-navigation in a web app, this path is used:
    path: 'people/:name',
    // The action and route params are extracted from the path.

    // Optional: Override the `navigationOptions` for the screen
    navigationOptions: ({navigation}) => ({
      title: `${navigation.state.params.name}'s Profile'`,
    }),
  },

  ...MyOtherRoutes,
});
```

StackNavigatorConfig

Options for the router:

- `initialRouteName` - Sets the default screen of the stack. Must match one of the keys in route configs.
- `initialRouteParams` - The params for the initial route
- `navigationOptions` - Default navigation options to use for screens
- `paths` - A mapping of overrides for the paths set in the route configs

Visual options:

- `mode` - Defines the style for rendering and transitions:
  - `card` - Use the standard iOS and Android screen transitions. This is the default.
  - `modal` - Make the screens slide in from the bottom which is a common iOS pattern. Only

rendered.

- `float` - Render a single header that stays at the top and animates as screens are changed. This is a common pattern on iOS.
- `screen` - Each screen has a header attached to it and the header fades in and out together with the screen. This is a common pattern on Android.
- `none` - No header will be rendered.

- `cardStyle` - Use this prop to override or extend the default style for an individual card in stack.
- `transitionConfig` - Function to return an object that is merged with the default screen transitions (take a look at TransitionConfig in type definitions). Provided function will be passed the following arguments:
  - `transitionProps` - Transition props for the new screen.
  - `prevTransitionProps` - Transitions props for the old screen.
  - `isModal` - Boolean specifying if screen is modal.
- `onTransitionStart` - Function to be invoked when the card transition animation is about to start.
- `onTransitionEnd` - Function to be invoked once the card transition animation completes.

Screen Navigation Options

`title`

String that can be used as a fallback for `headerTitle`. Additionally, will be used as a fallback for `tabBarLabel` (if nested in a TabNavigator) or `drawerLabel` (if nested in a DrawerNavigator).

`header`

React Element or a function that given `HeaderProps` returns a React Element, to display as a header. Setting to `null` hides header.

Search...

`headerTitle`

String, React Element or React Component used by the header. Defaults to scene `title`. When a component is used, `headerTitleAllowFontScaling`

Whether header title font should scale to respect Text Size accessibility settings. Defaults to true.

`headerBackTitle`

Title string used by the back button on iOS, or `null` to disable label. Defaults to the previous scene's `headerTitle`.

`headerTruncatedBackTitle`

Title string used by the back button when `headerBackTitle` doesn't fit on the screen. `"Back"` by default.

`headerRight`

React Element to display on the right side of the header.

`headerLeft`

React Element or Component to display on the left side of the header. When a component is used, it receives a number of props when rendered (`onPress`, `title`, `titleStyle` and more - check `Header.js` for the complete list).

`headerStyle`

Style object for the header

`headerTitleStyle`

Style object for the title component

`headerBackTitleStyle`

Style object for the back title

`headerTintColor`

Tint color for the header

`headerPressColorAndroid`

Color for material ripple (Android >= 5.0 only)

`gesturesEnabled`

Whether you can use gestures to dismiss this screen. Defaults to true on iOS, false on Android.

of the screen to recognize gestures. It takes the following properties:

- `horizontal` - *number* - Distance for horizontal direction. Defaults to 25.
- `vertical` - *number* - Distance for vertical direction. Defaults to 135.

Navigator Props

The navigator component created by `StackNavigator(...)` takes the following props:

- `screenProps` - Pass down extra options to child screens, for example:

```
const SomeStack = StackNavigator({
  // config
});

<SomeStack
  screenProps={/* this prop will get passed to the scr
een components as this.props.screenProps */}
/>
```

Examples

See the examples SimpleStack.js and ModalStack.js which you can run locally as part of the NavigationPlayground app.

You can view these examples directly on your phone by visiting our expo demo.

Modal StackNavigator with Custom Screen Transitions

```
const ModalNavigator = StackNavigator(
  {
    Main: { screen: Main },
    Login: { screen: Login },
  },
  {
    headerMode: 'none',
    mode: 'modal',
    navigationOptions: {
      gesturesEnabled: false,
    },
    transitionConfig: () => ({
      transitionSpec: {
        duration: 300,
      screenInterpolator: sceneProps => {
        const { layout, position, scene } = sceneProps;
        const { index } = scene;

        const height = layout.initHeight;
        const translateY = position.interpolate({
          inputRange: [index - 1, index, index + 1],
          outputRange: [height, 0, 0],
        });

        const opacity = position.interpolate({
          inputRange: [index - 1, index - 0.99, index],
          outputRange: [0, 1, 1],
        });

        return { opacity, transform: [{ translateY }] }
;
      },
    }),
  }
);
```

Edit on GitHub
Previous: Intro to Navigators Next: TabNavigator