Version: 7.x

Hello React Navigation

In a web browser, you can link to different pages using an anchor (`<a>`) tag. When the user clicks on a link, the URL is pushed to the browser history stack. When the user presses the back button, the browser pops the item from the top of the history stack, so the active page is now the previously visited page. React Native doesn't have a built-in idea of a global history stack like a web browser does -- this is where React Navigation enters the story.

React Navigation's native stack navigator provides a way for your app to transition between screens and manage navigation history. If your app uses only one stack navigator then it is conceptually similar to how a web browser handles navigation state - your app pushes and pops items from the navigation stack as users interact with it, and this results in the user seeing different screens. A key difference between how this works in a web browser and in React Navigation is that React Navigation's native stack navigator provides the gestures and animations that you would expect on Android and iOS when navigating between routes in the stack.

Let's start by demonstrating the most common navigator, `createNativeStackNavigator`.

Installing the native stack navigator library

The libraries we've installed so far are the building blocks and shared foundations for navigators, and each navigator in React Navigation lives in its own library. To use the native stack navigator, we need to install `@react-navigation/native-stack` :

npm Yarn pnpm

```
yarn add @react-navigation/native-stack
```

INFO

`@react-navigation/native-stack` depends on `react-native-screens` and the other libraries that we installed in Getting started. If you haven't installed those yet, head over to that page and follow the installation instructions.

Installing the elements library

The `@react-navigation/elements` library provides a set of components that are designed to work well with React Navigation. We'll use a few of these components in this guide. So let's install it first:

npm Yarn pnpm

```
yarn add @react-navigation/elements
```

Creating a native stack navigator

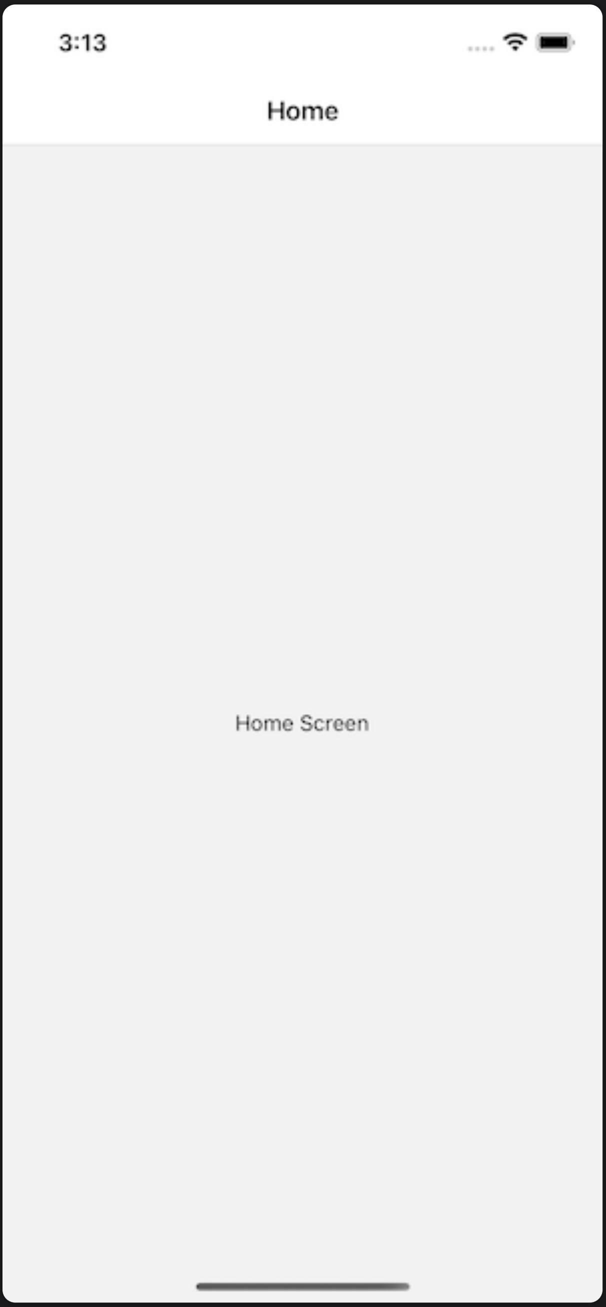
Static Dynamic

`createNativeStackNavigator` is a function that takes a configuration object containing the screens and customization options. The screens are React Components that render the content displayed by the navigator.

`createStaticNavigation` is a function that takes the navigator defined earlier and returns a component that can be rendered in the app. It's only called once in the app.

```
// In App.js in a new project

import * as React from 'react';
import { View, Text } from 'react-native';
import { createStaticNavigation } 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>
  );
}

const RootStack = createNativeStackNavigator({
  screens: {
    Home: HomeScreen,
  },
});

const Navigation = createStaticNavigation(RootStack);

export default function App() {
  return <Navigation />;
}
```

Try on **Snack**



If you run this code, you will see a screen with an empty navigation bar and a grey content area containing your `HomeScreen` component (shown above). The styles you see for the navigation bar and the content area are the default configuration for a stack navigator, we'll learn how to configure those later.

TIP

The casing of the route name doesn't matter -- you can use lowercase `home` or capitalized `Home`, it's up to you. We prefer capitalizing our route names.

Configuring the navigator

All of the route configuration is specified as props to our navigator. We haven't passed any props to our navigator, so it just uses the default configuration.

Let's add a second screen to our native stack navigator and configure the `Home` screen to render first:

Static Dynamic

```
const RootStack = createNativeStackNavigator({
  initialRouteName: 'Home',
  screens: {
    Home: HomeScreen,
    Details: DetailsScreen,
  },
});
```

Try on **Snack**

Now our stack has two *routes*, a `Home` route and a `Details` route. A route can be specified by under the `screens` property. The name of the property under `screens` corresponds to the name of the route we will use to navigate, and the value corresponds to the component it'll render.

Here, the `Home` route corresponds to the `HomeScreen` component, and the `Details` route corresponds to the `DetailsScreen` component. The initial route for the stack is the `Home` route. Try changing it to `Details` and reload the app (React Native's Fast Refresh won't update changes from `initialRouteName`, as you might expect), notice that you will now see the `Details` screen. Then change it back to `Home` and reload once more.

Specifying options

Each screen in the navigator can specify some options for the navigator, such as the title to render in the header.

Static Dynamic

To specify the options, we'll change how we have specified the screen component. Instead of specifying the screen component as the value, we can also specify an object with a `screen` property:

```
const RootStack = createNativeStackNavigator({
  initialRouteName: 'Home',
  screens: {
    Home: {
      screen: HomeScreen,
    },
    Details: DetailsScreen,
  },
});
```

This will let us specify additional options for the screen.

Now, we can add an `options` property:

```js
const RootStack = createNativeStackNavigator({
  initialRouteName: 'Home',
  screens: {
    Home: {
      screen: HomeScreen,
      options: {
        title: 'Overview',
      },
    },
    Details: DetailsScreen,
  },
});
```

Try on **Snack**

Sometimes we will want to specify the same options for all of the screens in the navigator. For that, we can add a `screenOptions` property to the configuration:

```js
const RootStack = createNativeStackNavigator({
  initialRouteName: 'Home',
  screenOptions: {
    headerStyle: { backgroundColor: 'tomato' },
  },
  screens: {
    Home: {
      screen: HomeScreen,
      options: {
        title: 'Overview',
      },
    },
    Details: DetailsScreen,
  },
});
```

Try on **Snack**

Passing additional props

Static   Dynamic

Passing additional props to a screen is not supported in the static API.

What's next?

The natural question at this point is: "how do I go from the `Home` route to the `Details` route?". That is covered in the next section.

▸ Using with TypeScript

Summary

Static Dynamic

- React Native doesn't have a built-in API for navigation like a web browser does. React Navigation provides this for you, along with the iOS and Android gestures and animations to transition between screens.
- `createNativeStackNavigator` is a function that takes the screens configuration and renders our content.
- Each property under screens refers to the name of the route, and the value is the component to render for the route.
- To specify what the initial route in a stack is, provide an `initialRouteName` option for the navigator.
- To specify screen-specific options, we can specify an `options` property, and for common options, we can specify `screenOptions`.

Edit this page